TargetLink

New Features and Migration

For TargetLink CD 1.1

May 2000
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 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 Document ................................................................. 5

New Features .............................................................................. 7
Key Features of TargetLink 1.1 ......................................................... 8
New Installation Features............................................................. 9
   Installation Manager .............................................................. 9
New TargetLink Features............................................................ 9
   Supported Targets ................................................................. 10
   Code Generator ..................................................................... 10
   Stateflow Code Generation .................................................... 12
   Block Library ........................................................................ 15
   Property Manager ............................................................... 16
   State-Space Scaling Tool ....................................................... 16
   Source File Export Utility ..................................................... 17
   Autoscaling ........................................................................... 17
   TargetLink API ...................................................................... 18
   ASAP2 File Generation ....................................................... 19
   Model Conversion Simulink <-> TargetLink............................. 20
   TargetLink Documentation Tool ........................................... 21
   Miscellaneous ....................................................................... 21

Migrating from TargetLink 1.0 to 1.1 .............................................. 23
Upgrading the Project Configuration File .................................... 24
Upgrading TargetLink Models .................................................... 25
   Upgrading a Model Automatically ....................................... 26
   Upgrading a Library Manually ............................................. 28
   Upgrading a Model Non-Interactively .................................. 29
Migration Features ..................................................................... 30
   Look-Up Tables .................................................................... 30
   TargetLink Block Properties .............................................. 30
   Volatile Attribute for Class DISP .......................................... 32
   TargetLink Info File ........................................................... 32
About This Document

This document provides you with a brief overview of the major features coming with TargetLink 1.1. Refer to:
- “Key Features of TargetLink 1.1” on page 8
In addition, you get a more detailed description of these key features as well as a summary of the most important enhancements and software changes since TargetLink 1.0 – including links to the printed and/or online documentation. Refer to:
- “New Installation Features” on page 9
- “New TargetLink Features” on page 9
Furthermore, this document provides you with information on the changes you have to perform when you switch from TargetLink 1.0 to TargetLink 1.1. Refer to:
- “Migrating from TargetLink 1.0 to 1.1” on page 23
New Features

This chapter shows you the new features, enhancements and changes that were made for TargetLink 1.1. Refer to:

- “Key Features of TargetLink 1.1” on page 8, providing an overview of TargetLink 1.1’s most important new features.
- “New Installation Features” on page 9, providing information on the new installation features of TargetLink 1.1.
- “New TargetLink Features” on page 9, listing the most important new features, enhancements and changes since TargetLink 1.0.
New Features

Key Features of TargetLink 1.1

This list shows TargetLink 1.1’s new key features.

New Installation Features

- Installation Manager

For details, see “New Installation Features” on page 9.

New TargetLink Features

- Additional supported targets
- Enhanced code generator with new variable class features
- Enhanced Stateflow code generation
- Additional and enhanced TargetLink blocks
- New Property Manager
- New State-Space Scaling Tool
- New File Export Utility
- Additional autoscaling features
- Additional TargetLink API features
- Enhanced ASAP2 file generation
- Enhanced model conversion Simulink <-> TargetLink
- Enhanced Documentation Tool

For details, see “New TargetLink Features” on page 9.
New Installation Features

The following installation features are new for TargetLink CD 1.1.

Installation Manager

To provide an easy method for working with more than one dSPACE installation on a system, dSPACE has developed a compact tool: Installation Manager. This tool allows you to switch between different versions of dSPACE software or to remove an entire dSPACE installation from your hard disk. For details, refer to Installation Manager in the TargetLink Production Code Generation Guide.

New TargetLink Features

This section describes the new features, enhancements and changes that were made for TargetLink 1.1:

- “Supported Targets” on page 10
- “Code Generator” on page 10
- “Stateflow Code Generation” on page 12
- “Block Library” on page 15
- “Property Manager” on page 16
- “State-Space Scaling Tool” on page 16
- “Source File Export Utility” on page 17
- “Autoscaling” on page 17
- “TargetLink API” on page 18
- “ASAP2 File Generation” on page 19
- “Model Conversion Simulink <-> TargetLink” on page 20
- “TargetLink Documentation Tool” on page 21
- “Miscellaneous” on page 21
Supported Targets

In addition to the previously supported targets, the following targets are now also supported by TargetLink 1.1:

- Hitachi H8S (TSM only)
- Motorola MPC555 (TSM only)
- Infineon TriCore (TSM and TOM)

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>Hitachi H8S</td>
<td>yes</td>
<td>+/-</td>
<td>for Hitachi compiler</td>
<td>Hitachi SH2 in the TargetLink</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>for Diab Data compiler</td>
<td>Reference</td>
</tr>
<tr>
<td>Motorola MPC555</td>
<td>yes</td>
<td>+/-</td>
<td>for Tasking compiler</td>
<td>MotorolaMPC5xx in the TargetLink</td>
</tr>
<tr>
<td>Infineon TriCore</td>
<td>yes</td>
<td></td>
<td>for Tasking compiler</td>
<td>Reference</td>
</tr>
</tbody>
</table>

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

Code Generator

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

New features

**Bool datatype**  The datatype ‘Bool’ has been introduced for Boolean variables. Boolean variables can now be implemented as single bits if this is supported by the target processor and if a target optimization module is available.

**Variable class properties**  The following variable class properties were added for TargetLink 1.1:

- preprocessor macro: If the variable class property ‘macro’ is enabled, the code generator creates preprocessor macros to access the corresponding variables. Macros with the storage class ‘extern’ can be redefined by the user. See Specifying a Parameter as a
Preprocessor Macro in the TargetLink Production Code Generation Guide and macro Field in the TargetLink Reference.

- **section name**: This assigns all variables belonging to a class to the selected linker section. TargetLink generates the desired compiler-specific statements automatically. This feature requires a Target Optimization Module (TOM). See Allocating Memory Sections in the TargetLink Production Code Generation Guide and sectionName Field (class Struct) in the TargetLink Reference.

- **pre-/post declaration statements (#pragma directives)**: You can insert statements before and after variable declarations, such as comments, #pragma / #ifdef directives and assembler statements. See class Struct in the TargetLink Reference.

- **configuration options for variable initialization**: You can now select between two options to initialize variables: You can disable the assignment of initial values to variables at declaration time. Or you can specify the name of the initialization function. The generation of an initialization function can be suppressed if the name is left empty. See Initializing Variables in the TargetLink Production Code Generation Guide.

- **type prefix**: This is inserted as a string in front of variable declarations, for example, 'near', 'far', or 'register'. See typePrefix Field in the TargetLink Reference.

- **variable as function return value**: TargetLink outports can be configured to become the function return value. See argClass Field in the TargetLink Reference.

- **Default variable classes and function classes**: These let you select classes for variables and functions that are created implicitly by the code generator (class ‘default’). For example, you can specify a variable class for auxiliary variables, or select a function class for Stateflow entry and exit functions. See Defining Default Classes for Implicit Variables in the TargetLink Production Code Generation Guide and defaultClass Struct or defaultFunctionClass Struct in the TargetLink Reference.

- **Functions with global variables as input/output signals**: These can be configured in the project configuration file by setting the default variable class fcnInput and fcnOutput to GLOBAL. See
New Features

Defining Default Classes for Implicit Variables in the TargetLink Production Code Generation Guide and defaultClass Struct in the TargetLink Reference.

**Function classes** Function classes were introduced to help you configure properties of functions such as the scope (static/global) or the code section. You can even generate inlined functions and interrupt functions. See Function Classes in the TargetLink Production Code Generation Guide and functionClass Struct in the TargetLink Reference.

**Enhancements and changes**

- **Full support for re-used functions** With re-used functions the code size can be reduced significantly. There is no longer the restriction that re-used functions can not contain state variables. Parameters and state variables of re-used functions are stored in instance-specific data structures. Systems that contain state charts can not be re-used.

- **Control flow optimization** The Advanced page of the TargetLink Main Dialog (see Advanced Page in the TargetLink Reference) now has the option "Always optimize control flow." When this option is selected the control flow is also optimized if a block contains a user-defined class, such as DISP. Nested if-statements are generated for sequences of switch statements. See Optimizing the Control Flow in the TargetLink Production Code Generation Guide.

- **Additional optimizations** The code generator’s capability to eliminate superfluous variables and optimize the control flow has been improved. See Optimizing the Production Code in the TargetLink Production Code Generation Guide.

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**
  - C-like bit operations: TargetLink 1.1 now provides the "Enable C-like bit operations" option on the Advanced page of the TargetLink Main Dialog (see Advanced Page in the TargetLink Reference). This is similar to the option in the Stateflow
Simulation/RTW Coder Options dialog. If you select this option, the operators '&', '|', '^' and '~' are interpreted as bit-wise operators.

- **Math functions**: The functions $\min$, $\max$ and $\abs$ are now supported in floating-point and fixed-point arithmetic. All functions defined in the C library `<math.h>` are supported in floating-point arithmetic.

- **Comments**: The contents of the description fields of charts, states and data are placed as comments in the production code. Comments in state and transition annotations are also assumed in the production code. See Stateflow Object Properties in the TargetLink Reference.

**TargetLink production code**

- **TargetLink properties for Stateflow data variables**: TargetLink 1.1 supplies the full range of production code properties for all Stateflow data variables. For each Stateflow data variable you can specify the variable name, datatype, LSB, offset, variable class (code section, storage class, declaration prefix, ...) and address. See Using the TargetLink Property Manager in the TargetLink Production Code Generation Guide.

- **Switch transformation**: TargetLink 1.1 supports the modeling of switch statements. TargetLink 1.0 could not detect flowcharts that represented a switch construct and always converted them to less efficient nested if-then-else statements. See Switch Statements in the TargetLink Production Code Generation Guide.

- **Stateflow subsystem**: In TargetLink 1.1 each chart can become a separate function. You can specify the function name and the code file it will be written into. In addition, you can specify a function class that contains information about the storage class, declaration prefix, linker section, etc. See Stateflow Subsystems in the TargetLink Production Code Generation Guide.

**Stateflow modeling**

- **Arrays**: The limitation in TargetLink 1.0 that the first index of array data must be zero is no longer valid for TargetLink 1.1. Any integer value can be chosen as the first index.
TargetLink production code

- Customization of implicitly generated variables: In addition to the data variables specified in the Stateflow diagram, the code generator has to introduce implicit variables and functions to implement the chart. Via a configuration file you can customize the storage class, linker section and other properties of the generated variables and functions. See Customization of Generated Variables and Functions in the TargetLink Production Code Generation Guide.

Code efficiency

- Single-bit state representation: TargetLink 1.1 implements the state of a state machine by using a single bit for each state. This bit represents the activity of the state. Especially if a hierarchical state only has a few children, which is very often the case, this implementation leads to smaller code size in comparison to the substate representation.

- Transition actions: The coding of transition actions has been improved. The use of flag variables that represent the activity of a switching transition is minimized.

- Common 'entry' and 'during' functions: The MathWorks' recommended style for modeling common 'entry' and 'during' functions is to create a default transition and an inner transition that share the same destination junction. This kind of flowchart is detected by TargetLink 1.1 and the code is generated only once.

- State entry function parameters: The number and types of state entry function parameters are optimized. Default flags and parameters representing the current event are generated only if they are necessary.
Block Library

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

New features

**FIR Filter block**  The new FIR Filter block enables the highly efficient implementation of FIR filters. Special optimizations are generated for processors with DSP instructions (Infineon TriCore, Hitachi SH2). See FIR Filter Block in the TargetLink Reference for details. See FIR Filter block in the TargetLink Production Code Generation Guide to see how to use this block and for background information on FIR filters.

**Look-Up Table blocks**  There are two new features here:

- There is now a separate index search and interpolation for Look-Up Table blocks. This optimizes the calculation of multiple look-up tables with common input vectors and saves execution time. See Using a Separate Search Function in the TargetLink Production Code Generation Guide.

- TargetLink’s look-up table functions can now be replaced by custom look-up functions. The replacement is described by a MATLAB M-script. Any look-up table structures may be defined. See Using Custom Look-Up Table Functions in the TargetLink Production Code Generation Guide.

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

**Math block**  The Math block now supports all math functions for floating-point code generation. The functions square, sqrt, reciprocal, rem and mod are also supported for fixed-point datatypes. See Math Block in the TargetLink Reference for details.

**Relay block**  The Relay block is now fully supported. See Relay Block in the TargetLink Reference for details.

Enhancements and changes

**Custom Code block**  In TargetLink 1.0, the custom code was enclosed by a couple of #define statements to establish the interface to the surrounding C code. This is avoided in TargetLink 1.1, which improves the readability of the code. See Custom Code Block in the TargetLink Reference for details.
Property Manager

TargetLink 1.1 now provides the TargetLink Property Manager, which enables you to quickly handle models with numerous blocks. It is also a convenient interface to attach TargetLink properties to state machine (Stateflow) variables. With the Property Manager you can:

- Handle large models via a graphical frontend
- Navigate through the model via the Model Browser
- Search for certain blocks, e.g., all Look-Up Table blocks in a model
- Modify multiple blocks at a time, e.g., change the datatype from UInt8 to UInt16
- Autoscale selected subsystems
- Set TargetLink properties for Stateflow objects

For instructions on how to use the Property Manager, see Using the TargetLink Property Manager in the TargetLink Production Code Generation Guide. For a detailed description see TargetLink Property Manager in the TargetLink Reference.

State-Space Scaling Tool

TargetLink 1.1 now provides the State-Space Scaling Tool, which simplifies the complicated task of scaling state-space systems. This tool computes worst case signal ranges according to the L1 algorithm and scales the Discrete State-Space block accordingly.

For background information and instructions on how to operate the State-Space Scaling Tool, refer to Scaling State-Space Outputs and States via the State-Space Scaling Tool in the TargetLink Production Code Generation Guide. For a detailed description see State-Space Scaling Tool in the TargetLink Reference.
New Features

Source File Export Utility

TargetLink 1.1 now provides the File Export Utility, which is called up via `tl_export_files` in the MATLAB Command Window or from the Copy files button on the TargetLink Main Dialog's Tools page.

This utility carries out the following tasks:

- It copies all generated files and required standard header files/object libraries to a selected directory. It also analyzes all dependencies and collects all files to build a stand-alone application that does not require a TargetLink installation.
- It creates utility M-files to postprocess the generated code. For example, there is the `make_ASAP2` utility to create an ASAP2 file after the final target application has been built.

For more information, see Exporting Generated Files in the TargetLink Production Code Generation Guide and TargetLink Source File Export Utility in the TargetLink Reference.

Autoscaling

This section shows the new features that were added to TargetLink’s Autoscaling Tool. For more information, see Preparing and Performing Autoscaling in the TargetLink Production Code Generation Guide and Scaling Page (for the TargetLink Main Dialog) in the TargetLink Reference.

Autoscaling of parameters

In TargetLink 1.0, only block outputs could be autoscaled. Now, in TargetLink 1.1, the parameters of all TargetLink blocks can be autoscaled too, including Look-Up Table blocks and FIR Filter blocks.

Selection of optimized datatype and word width for parameters

If this option is used, the Autoscaling Tool selects an unsigned datatype for positive variables to reduce the quantization error. If a parameter can be realized as an 8-bit variable without loss of precision, the smaller datatype is selected.

Execution

The Autoscaling Tool can also be executed from the context menu of the Property Manager's Model Browser, or via the
**New Features**

Direct API function call `tl_autoscaling`. You can apply autoscaling to individual blocks or subsystems instead of just the whole model.

**Advanced L1 scaling method** TargetLink 1.1 has an advanced L1-scaling method for the Discrete State-Space block and the FIR Filter block. This computes worst case output signal ranges depending on the range of the input signals. See Implementation of the L1-Scaling Method in the TargetLink Production Code Generation Guide and State-Space Scaling Tool in the TargetLink Reference.

**TargetLink API**

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

**New features**

**New commands to automate code generation** The following commands were added to automate code generation:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tl_generate_code</code></td>
<td>to generate code for a model</td>
</tr>
<tr>
<td><code>tl_compile_host</code></td>
<td>to compile the host S-function</td>
</tr>
<tr>
<td><code>tl_compile_target</code></td>
<td>to compile the target application</td>
</tr>
<tr>
<td><code>tl_build_host</code></td>
<td>to build S-functions for production code host simulation</td>
</tr>
<tr>
<td><code>tl_build_target</code></td>
<td>to build production code target applications</td>
</tr>
<tr>
<td><code>tl_download</code></td>
<td>to download to the target evaluation board</td>
</tr>
</tbody>
</table>

See Code Generation Related Utilities in the TargetLink Reference for more details.
**hook functions** In TargetLink 1.1 there are now hook functions available at various points during code generation:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl_pre_codegen_hook,</td>
<td>to customize the code generation phase</td>
</tr>
<tr>
<td>tl_post_codegen_hook,</td>
<td></td>
</tr>
<tr>
<td>tl_post_framegen_hook</td>
<td></td>
</tr>
<tr>
<td>tl_pre_compile_host_hook,</td>
<td>to customize the compilation phase for production code host S-functions</td>
</tr>
<tr>
<td>tl_post_compile_host_hook</td>
<td></td>
</tr>
<tr>
<td>tl_pre_compile_target_hook,</td>
<td>to customize the compilation phase for production code target applications</td>
</tr>
<tr>
<td>tl_post_compile_target_hook</td>
<td></td>
</tr>
<tr>
<td>tl_pre_download_hook,</td>
<td>to customize the download phase for production code target applications</td>
</tr>
<tr>
<td>tl_post_download_hook</td>
<td></td>
</tr>
</tbody>
</table>

See Code Generation Related Utilities in the TargetLink Reference for more details.

**Error handling functions** There is now a common interface to check for and handle errors in TargetLink API commands. See Error Handling Utilities in the TargetLink Reference for more details.

**Stateflow properties** Stateflow properties can now be accessed via the `tl_get` and `tl_set` utilities. See `tl_get` and `tl_set` in the TargetLink Reference for more details.

### ASAP2 File Generation

This section shows the enhancements and changes that were made in TargetLink's ASAP2 file generation.

**ASAP2 generation for Stateflow objects** TargetLink 1.1 now supports ASAP2 generation for Stateflow objects. Stateflow objects are now listed in the ASAP2 file. Bit variables denoting states are not supported yet.

**Support of re-used functions** Data structures with instance-specific data of re-used functions are now fully supported.

**Verbal conversion tables** A configuration file now lets you specify verbal tables, for example, to map logical values 1/0 to the strings

**Look-up tables with equidistant axis**   The limitation in TargetLink 1.0 that the parameters of equidistant table axes could not be calibrated has been removed for TargetLink 1.1.

**New ASAP2 template files**   There are now ASAP2 template files to generate an ASAP2 file without address information, for example. In this case it is not necessary to provide the final target application.

For more information, see Using the ASAP2 File Generator in the TargetLink Production Code Generation Guide.

### Model Conversion Simulink <-> TargetLink

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

**Enhancements and changes**

**Mapping of blocks**   In TargetLink 1.1, the mapping of blocks can be made in dependence on current block parameters. For example, a Function block with the expression \( u(1) \times u(2) \) can be mapped to a Product block, while \( u(1) \& u(2) \) is mapped to a Simulink AND block.

**Demo example**   TargetLink now includes a demo example, located in the directory `<DSPACE_ROOT>\demos\tl\conversion`, that shows how to configure the library mapping. This demo example contains various implementations of the Simulink Discrete Integrator block.
**TargetLink Documentation Tool**

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

<table>
<thead>
<tr>
<th>Enhancements and changes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stateflow objects</strong></td>
<td>The Documentation Tool now supports Stateflow objects.</td>
</tr>
<tr>
<td><strong>Re-used functions</strong></td>
<td>The Documentation Tool now supports re-used functions.</td>
</tr>
<tr>
<td><strong>Output directory</strong></td>
<td>You can now specify the directory where the documentation files are placed. For more information, see Automatic Document Generation in the TargetLink Production Code Generation Guide.</td>
</tr>
</tbody>
</table>

**Miscellaneous**

TargetLink 1.1 also has the following new feature.

<table>
<thead>
<tr>
<th>New feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-defined code editor</strong></td>
<td>You can now select your own preferred code editor. See Configuring a User-Defined Code Editor in the TargetLink Production Code Generation Guide.</td>
</tr>
</tbody>
</table>
Migrating from
TargetLink 1.0 to 1.1

After you install TargetLink 1.1 it is necessary to carry out the following steps:

- “Upgrading the Project Configuration File” on page 24
- “Upgrading TargetLink Models” on page 25

In addition, you should be aware of the following changes since TargetLink 1.0:

- “Migration Features” on page 30

Please refer to the TargetLink ReadMe file in the directory <DSPACE_ROOT>\ReadMe for important last-minute information.
Upgrading the Project Configuration File

Since many fields in the project configuration file (by default default_cfg.m in the directory <DSPACE_ROOT>/matlab/tl/config) have changed, project configuration files written for TargetLink 1.0 can not be used with TargetLink 1.1. To upgrade your configuration files use the upgrade tool tl_upgrade_projectfile to convert your old project configuration file to TargetLink 1.1:

tl_upgrade_projectfile('OldFile', <file_Vs1_0>, 'NewFile', ...<file_Vs1_1>)

with

<file_Vs1_0> (name of the Vs. 1.0 project configuration file)
<file_Vs1_1> (name of the Vs. 1.1 project configuration file).

You can use filenames with or without absolute or relative paths. If you invoke the tool with identical filenames, a backup copy <file_Vs1_0>.bak will be generated.

Suppose your old project configuration file is myproj_cfg.m and you want to convert it to the new file myproj_1_1_cfg.

tl_upgrade_projectfile('OldFile', 'myproj_cfg', 'NewFile', ...'myproj_1_1_cfg')

will process myproj_cfg.m (which must reside on the MATLAB search path or in the current working directory), and generate a new Vs. 1.1 project configuration file myproj_1_1_cfg.m in your current working directory.

tl_upgrade_projectfile('OldFile', '.\myconfigs\myproj_cfg',... 'NewFile', 'f:\projects\configs\myproj_1_1_cfg')

will process myproj_cfg.m in the subdirectory myconfigs, and generate a new Vs. 1.1 project configuration file myproj_1_1_cfg.m in f:\projects\configs.
In TargetLink 1.1 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.

### Upgrading TargetLink Models

Property names of TargetLink blocks have changed from TargetLink 1.0 to TargetLink 1.1 and some new properties have been introduced. In addition, TargetLink data of Stateflow objects are now attached in a different format. Therefore, all Simulink models that contain TargetLink blocks or Stateflow objects with TargetLink data and that were created with TargetLink 1.0 need to be upgraded if they should be processed with TargetLink 1.1.

- If you skip upgrading, you will encounter numerous errors when you try to work with your non-upgraded model. This applies, for example, if you try to open a TargetLink block dialog, start code generation, or invoke any of the TargetLink tools.
- You need a valid TargetLink 1.1-compatible project configuration file before any upgrading is performed. Refer to “Upgrading the Project Configuration File” on page 24.

This chapter contains the following sections:

- “Upgrading a Model Automatically” on page 26 (the recommended procedure)
- “Upgrading a Library Manually” on page 28 (if you want to manually upgrade the block library)
- “Upgrading a Model Non-Interactively” on page 29 (useful for upgrading several models at the same time)
Upgrading a Model Automatically

TargetLink 1.1 provides the upgrading tool `tl_upgrade` to upgrade your model automatically. This tool (see below) 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. This automatically starts `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 1.1 if it is not upgraded. During the upgrade, messages about which block is currently being processed are pasted into 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 (see below).

3 For TargetLink Custom Code blocks, special upgrading rules apply. You will be 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.
Upgrading a Library Manually

If your model contains TargetLink blocks with library links these blocks can not 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, `tl_upgrade` tells you that the libraries need still to be upgraded (see below).

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:
   ```
   tl_upgrade <library>
   ```
   in the MATLAB Command Window, where `<library>` is the library’s name. The library will now be 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 1.1.
Upgrading 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:
   \texttt{tl\_error\_set batchmode on}
   in the MATLAB Command Window.
2. Start the upgrading tool by invoking:
   \texttt{tl\_upgrade <mymodel>}
   in the MATLAB Command Window,
   \texttt{tl\_upgrade}
   will run through the upgrading process without having you answer message boxes.
Migration Features

This chapter describes changes that you should be aware of when migrating from TargetLink 1.0 to TargetLink 1.1:

- “Look-Up Tables” on page 30
- “TargetLink Block Properties” on page 30
- “Volatile Attribute for Class DISP” on page 32
- “TargetLink Info File” on page 32

Look-Up Tables

The parameters of 1D look-up tables are now stored in a data structure that is passed as the first input argument to the look-up function. This is the same approach as for 2D look-up tables. In TargetLink 1.1, 1D look-up functions are always called according to the syntax

\[
y = \text{tablidxxx}(&\text{strct}, x); \quad \text{(TL 1.1)}
\]

This saves some bytes on the stack compared to the previous syntax where all parameters were passed as single arguments:

\[
y = \text{tablidxxx}(x\_table, y\_table, n, x); \quad \text{(TL 1.0)}
\]

For 2D look-up tables, the applied data structures and the syntax of look-up functions have not changed.

TargetLink Block Properties

When you work with the TargetLink API keep in mind that some block library properties have been changed for TargetLink 1.1. Refer to TargetLink Block Properties in the TargetLink Reference for detailed tables of the properties of each TargetLink blocktype.
New property names

The following property names have been changed:

<table>
<thead>
<tr>
<th>Previous Name in Data Struct</th>
<th>Previous Name in TargetLink API</th>
<th>New Name in Data Struct and API</th>
</tr>
</thead>
<tbody>
<tr>
<td>arb</td>
<td>arbitrary</td>
<td>arb</td>
</tr>
<tr>
<td>arbndist</td>
<td>arbdist</td>
<td>arbdistance</td>
</tr>
<tr>
<td>autosc</td>
<td>autosc</td>
<td>autoscalingmode</td>
</tr>
<tr>
<td>data_to_log</td>
<td>data_to_log</td>
<td>loggingmode</td>
</tr>
<tr>
<td>dec_const</td>
<td>decimals</td>
<td>decimals</td>
</tr>
<tr>
<td>deviation</td>
<td>deviation</td>
<td>tolerance</td>
</tr>
<tr>
<td>dolog</td>
<td>dolog</td>
<td>globalloggingmode</td>
</tr>
<tr>
<td>extrapol</td>
<td>extrapolate</td>
<td>extrapolate</td>
</tr>
<tr>
<td>fthrough</td>
<td>feedthrough</td>
<td>feedthrough</td>
</tr>
<tr>
<td>fxp_code</td>
<td>fxp_code</td>
<td>prodcode</td>
</tr>
<tr>
<td>hroomunit</td>
<td>headroomunit</td>
<td>headroomunit</td>
</tr>
<tr>
<td>inlinemode</td>
<td>inlinemode</td>
<td>inline</td>
</tr>
<tr>
<td>inputs</td>
<td>inputs</td>
<td>ninputs</td>
</tr>
<tr>
<td>interp</td>
<td>interpolate</td>
<td>interpolate</td>
</tr>
<tr>
<td>lthrough</td>
<td>lower_headroom</td>
<td>lowerheadroom</td>
</tr>
<tr>
<td>link</td>
<td>link</td>
<td>addfilemode</td>
</tr>
<tr>
<td>lsb</td>
<td>LSB</td>
<td>lsb</td>
</tr>
<tr>
<td>max_num_simulations</td>
<td>max_num_simulations</td>
<td>maxnumsimulations</td>
</tr>
<tr>
<td>sfcn</td>
<td>sfcn</td>
<td>codefile</td>
</tr>
<tr>
<td>signalname</td>
<td>signalname</td>
<td>dsmname</td>
</tr>
<tr>
<td>target_timeout</td>
<td>target_timeout</td>
<td>targettimeout</td>
</tr>
<tr>
<td>uhroom</td>
<td>upper_headroom</td>
<td>upperheadroom</td>
</tr>
<tr>
<td>$x_0$</td>
<td>$x_0$</td>
<td>$x_0$</td>
</tr>
<tr>
<td>$x_n$</td>
<td>$x_n$</td>
<td>$x_n$</td>
</tr>
</tbody>
</table>

The dolog flag is replaced by two separate properties in the front end: globalloggingmode and cleancode.
Volatile Attribute for Class DISP

When using TargetLink 1.1, keep this change in mind: The variable class DISP is defined with the property class(i).volatile = 0;. Variables belonging to the predefined class DISP and the related classes STATIC_DISP, EXTERN_DISP, STATIC_LOCAL_DISP and EXTERN_LOCAL_DISP are declared without the volatile attribute. This enables the compiler to apply more optimizations to DISP variables.

For example, if the class DISP was set for the variable disp_var, this leads to the declaration

```c
Int16 disp_var;  // (TL 1.1)
```

instead of

```c
volatile Int16 disp_var;  // (TL 1.0)
```

TargetLink Info File

The structure of the Info file was changed significantly for TargetLink 1.1.

**Old structure**

In TargetLink 1.0 the information about blocks and their corresponding variables was generated into each single function description (Info file property FCN_xxx). For example, a Gain block belonging to the function controller was generated into the block list of FCN_controller.

**New structure**

TargetLink 1.1 generates the block data and variable data into separate lists, which are referenced via indices by the function description. With this approach the processing of Info files can be accelerated significantly.
New lists in the Info file

The following new lists are contained the Info file:

<table>
<thead>
<tr>
<th>List</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeInfo</td>
<td>list of all datatype definitions, variable classes and function classes applied in the generated C file(s)</td>
</tr>
<tr>
<td>StateFlow</td>
<td>list of all Stateflow nodes and variables referenced from the Stateflow machine</td>
</tr>
<tr>
<td>Variables</td>
<td>list of all variables applied in the generated C file(s)</td>
</tr>
<tr>
<td>BlockData</td>
<td>list of all blocks that contain variables which appear in the C code</td>
</tr>
</tbody>
</table>

If you wrote any utilities to process the Info file, you must adapt them to the new TargetLink 1.1 Info file format, which relies on indexed references into the lists described above.

For details please refer to TargetLink Info File in the TargetLink Reference and Obtaining Information from the Info File in the TargetLink Production Code Generation Guide.

Example M-script

A sample M-script (info_demo.m) is located in the directory \$DSPACE_ROOT\demos\tl\poscontrol. This file shows how to process an Info file in the TargetLink 1.1 format. Use the file as a template for your own scripts to:

- walk through the hierarchy of generated functions
- get a list of all global variables
- get a list of all calibratable variables
- get a list of all blocks belonging to a TargetLink subsystem
- get a list of all variables belonging to a given function
- get instance-specific information about re-used functions