TargetLink

New Features and Migration

TargetLink 2.1.5 – November 2005

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Technologiepark 25 33100 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 66529
E-mail:	info@dspace.de
Web:	http://www.dspace.com
General Technical Support:	support@dspace.de +49 5251 1638-941 http://www.dspace.com/goto?support
TargetLink Support:	support.tl@dspace.de +49 5251 1638-700

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/goto?support for software updates and patches.

Important Notice

Contents

About This Document

This document provides you with a brief overview of the major new features of TargetLink 2.1 since TargetLink 2.0.

New features and enhancements

For a description of the key features, and a summary of the major enhancements made since TargetLink 2.0, refer to *Key Features of TargetLink 2.1* on page 7.

Migration

For information on the changes you have to perform when you migrate from previous releases to TargetLink 2.1, refer to *Migrating to TargetLink 2.1* on page 35.

Last-minute information

For information on last-minute changes of TargetLink Release 2.1, refer to *Last-Minute Information* on page 41.

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 2.1

TargetLink 2.1 has the following key features, enhancements and changes.

New TargetLink features

■ *Miscellaneous Features* on page 31

Supported MATLAB Platforms

TargetLink 2.1 supports the following MATLAB releases:

MATLAB Release ...	With Simulink Version ...	And Stateflow Version ...
MATLAB R14SP2+	Simulink 6.2.1	Stateflow 6.2.1
MATLAB R14SP2	Simulink 6.2	Stateflow 6.2
MATLAB R14SP1	Simulink 6.1	Stateflow 6.1
MATLAB R13SP2	Simulink 5.2	Stateflow 5.1.2
MATLAB R13SP1+	Simulink 5.1.1	Stateflow 5.1.2
MATLAB R13SP1	Simulink 5.1	Stateflow 5.1.1
MATLAB R13.0.1	Simulink 5.0.2	Stateflow 5.1

The Java Virtual Machine (JVM) 1.5.0 originally installed with MATLAB R14SP2 causes errors that particularly disturb the work with TargetLink, although the same errors also occur without TargetLink.

Thus TargetLink 2.1 has been released explicitly for the Java Virtual Machine 1.4.2_08. It is strongly recommended to use this JVM in conjunction with TargetLink 2.1.

You can install the JVM 1.4.2_08 from `Tools/Java` in the root folder of the CD.

Limitations As of MATLAB R14SP2, the MathWorks Installer has two new features on Windows systems:

- The Installer now allows a folder name with spaces in the installation path.
- By default, the Installer selects the Windows default installation folder, which is **Program Files** on most machines.

dSPACE software does **not** support MATLAB installation folder names with spaces in the path. The dSPACE Setup issues a warning if you try to install the dSPACE software for a MATLAB installation that has spaces in its installation path.

You also cannot install dSPACE software in paths that contain spaces.

Installing MATLAB in a network dSPACE Setup does not check whether MATLAB is installed in a network or locally on your host PC. If MATLAB is installed in a network, it depends on your modify rights how dSPACE Setup modifies the `matlabrc.m` file:

- If you do not have modify rights, dSPACE Setup issues a message and lets you modify `matlabrc.m` manually or specify another MATLAB folder.
- If you have modify rights, dSPACE Setup modifies `matlabrc.m` automatically.

For information on how to install TargetLink and a detailed description of its requirements, refer to *Installation Overview* in the *TargetLink Installation and Configuration Guide*.

Combination of dSPACE Products

To install a combined installation of dSPACE software (dSPACE Solutions for Control, TargetLink, CalDesk) in a folder where dSPACE software is already installed, you must install the dSPACE software items of the combined installation in a specified order.

Combinations that do not comply with these installation orders will not work properly.

The table below shows you which installation orders will work:

Combination is Possible When Installing dSPACE Software ...	After dSPACE Software ...											
	dSPACE Solutions for Control Release				TargetLink				CalDesk			
	4.0.x	4.1	4.2	5.0	2.0	2.0.6/ 2.0.7	2.1	2.1 with Master Setup	1.0	1.1	1.2	1.2.2
RLS[1] 4.0.x	–				No	No	No	No	Yes	No	No	No
RLS 4.1					No	No	No	No	Yes	No	No	No
RLS 4.2					Yes	Yes	No	No	No	Yes[4]	No	No
RLS 5.0					Yes	Yes	Yes	No	No	No	Yes	No
TL[2] 2.0	Yes	Yes	No	No	–				Yes[5]	No	No	No
TL 2.0.6/2.0.7	Yes	Yes	No	No					Yes[6]	Yes	No	No
TL 2.1	Yes	Yes	Yes	No					No	Yes	Yes	No
TL 2.1 with Master Setup	Yes	Yes	Yes	Yes					No	Yes	Yes	No
CAL[3] 1.0	Yes	No	No	No	No	No	No	No	–			
CAL 1.1	Yes	Yes	No	No	Yes[7]	No	No	No				
CAL 1.2[8]	Yes	Yes	Yes[9]	No	Yes	Yes	No	No				
CAL 1.2.2	Yes	Yes	Yes[9]	Yes	Yes	Yes	Yes	Yes				

1) dSPACE Solutions for Control Release

2) TargetLink

3) CalDesk

4) Only if you have installed the Unicode version of CAL 1.1.1. To install the Unicode version of CAL 1.1.1, start the `CDsetup.exe` program from the CD with the command line parameter `/UNICODE`, for example, `e:\CDsetup.exe /UNICODE`.

5) Only if you have installed CAL 1.0.2.

6) Not for CAL 1.0.2

7) For combining the Unicode version of CalDesk 1.1.1 with TargetLink 2.0, see below.

8) By default, the Unicode version of CAL 1.2 is installed. If a dSPACE installation already exists in the setup folder, CalDesk's setup installs CAL 1.2 in the same version (Unicode or non-Unicode) as the existing installation.

9) If RTI Bypass Blockset 2.0 is installed (included in RLS 4.2), an update to RTI Bypass Blockset 2.0.1 will be installed automatically for compatibility reasons.

For the latest information on combining dSPACE software, visit http://www.dspace.com/goto?ds_sw_combi.

The following example will help you to interpret the table.

If you first install dSPACE Release 4.2, then CalDesk 1.2, and thirdly TargetLink 2.1, the installation will work.

It is not possible to install CalDesk 1.2 in an existing TargetLink 2.1 installation because CalDesk 1.2 was released before TargetLink 2.1.

The installations are listed in the dSPACE Installation Manager and the **Add/Remove Programs** list of the Windows Control Panel. Since installations are listed alphabetically and not chronologically, do not forget to make a note of the order in which you combined the dSPACE software in. Otherwise, it will be difficult to reconstruct the installation order later on.

TargetLink 2.1 Compatibility

The table below shows you which installation orders (scenarios) will work and which restrictions apply:

	MATLAB Version	dSPACE Software	TargetLink 2.1 installation Possible[1]	Note / Limitation
Product dependencies (with regard to the active installation):				
		TargetLink 1.3	NO	-
		TargetLink 2.0.x (including TargetLink Blockset 2.0.x)	NO	-
		TargetLink Blockset 2.1(stand-alone)	NO	-
		dSPACE Data Dictionary Manager 1.3.1	YES	-
		CalDesk 1.0.2	NO	-
		CalDesk 1.1.x	YES	CalDesk 1.1.x and TargetLink 2.1 use different Data Dictionary versions. So Data Dictionary files generated with TargetLink 2.1 cannot be opened in CalDesk 1.1.x.
		CalDesk 1.2	YES	-
No active dSPACE installation				
	< R12.1		NO	-
	< = R14SP2		YES	-
	others (including R14SP2+)		YES	For compatibility information, refer to http://www.dspace.com/goto?TL21
Solutions for Control (SFC) and MATLAB dependencies:				
		Release < 3.4	NO	-
	R12.1	Release 3.4.x	NO	-
	R13	Release 3.4 + MLCU[2] for R13	YES	Only MATLAB R13 is supported. This combination has not been qualified but will not be blocked.
		Release 3.5	YES	This combination has not been qualified but will not be blocked.
		Release 4.0	YES	This combination has not been qualified but will not be blocked.
		Release 4.0.1	YES	This combination has not been qualified but will not be blocked.
		Release 4.1	YES	MTest 1.2 projects cannot access TargetLink 2.1
		Release 4.2	NO	
	R13.0.1	Release 3.4 + MLCU[2] for R13	YES	This combination has not been qualified but will not be blocked.
		Release 3.5	YES	This combination has not been qualified but will not be blocked.
		Release 4.0	YES	This combination has not been qualified but will not be blocked.
		Release 4.0.1	YES	This combination has not been qualified but will not be blocked.
		Release 4.1	YES	
		Release 4.2	YES	

	MATLAB Version	dSPACE Software	TargetLink 2.1 installation Possible[1]	Note / Limitation
	R13SP1	Release 4.0 + MLCU[2] for R13SP1	YES	Only MATLAB R13SP1 is supported. This combination has not been qualified but will not be blocked.
		Release 4.0.1	YES	This combination has not been qualified but will not be blocked.
		Release 4.1	YES	
		Release 4.2	YES	
	R13SP1+	Release 4.1	YES	
	R13SP2	Release 4.1 + MLCU[2] for R13SP2	YES	Only MATLAB R13SP2 is supported
		Release 4.2	YES	
	R14	Release 4.1 + MLCU[2] for R14	NO	Only MATLAB R14 is supported
		Release 4.2	NO	
	R14SP1	Release 4.1 + MLCU[2] for R14SP1	YES	Only MATLAB R14SP1 is supported
		Release 4.2	YES	
	R14SP2	Release 4.2	YES	
	R14SP2+	Release 4.2	YES	For compatibility information, refer to http://www.dspace.com/goto?TL21

1) The same applies to the TargetLink Blockset 2.1 (stand-alone)

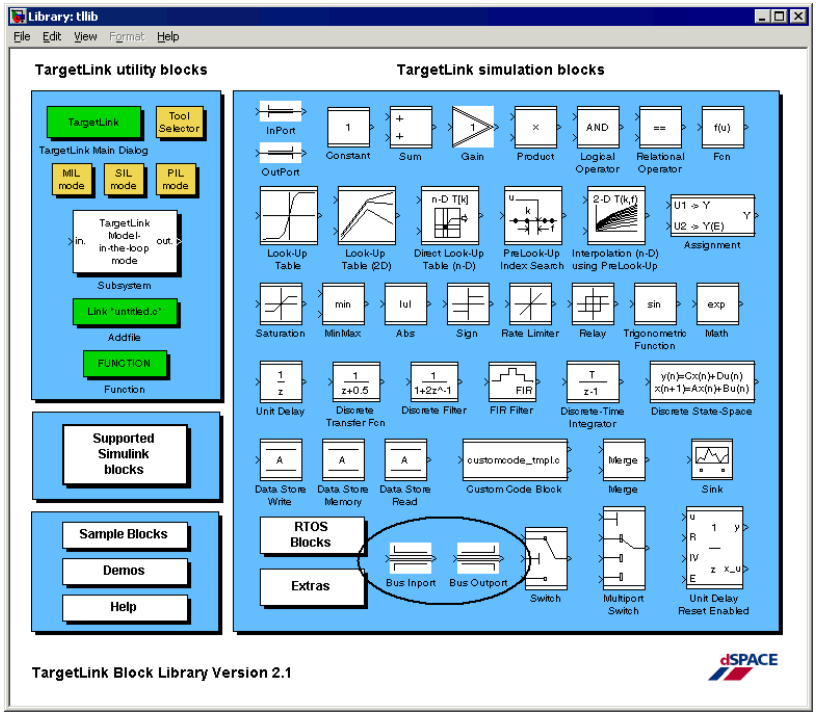
2) MLCU stands for MATLAB Compatibility Update

For combinations that are not mentioned explicitly in the table, the installation of TargetLink 2.1 is not possible.

Representation of Busses in the Production Code

Bus port blocks for busses

TargetLink supports busses at the boundaries of the root level of a TargetLink subsystem. Busses combine multiple separate signals (bus elements) and can themselves contain busses (subbusses). In TargetLink, you can work with busses using the Bus Inport and Bus Outport blocks (called bus port blocks below) of the TargetLink block library.



With the bus port blocks, you can:

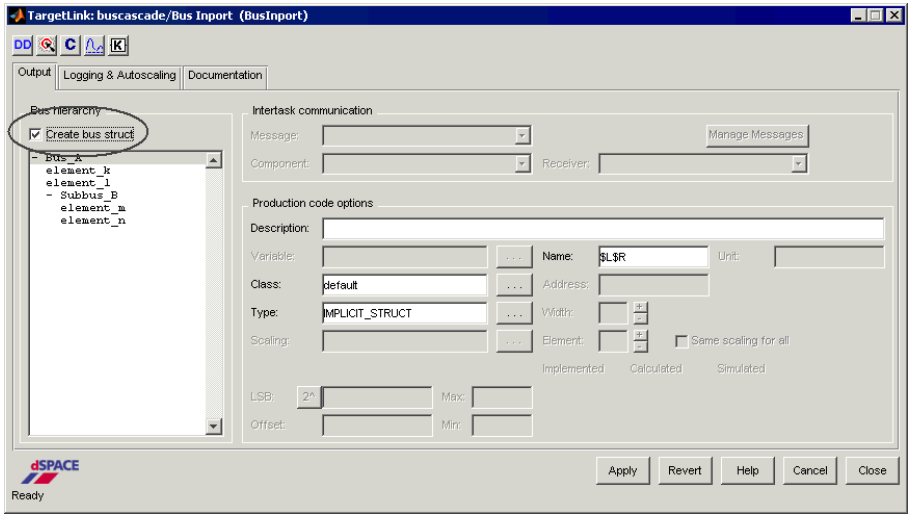
- Map bus elements to separate variables
- Generate struct variables for busses
- Map bus elements to predefined struct variables in hand-written code

Mapping bus elements to separate variables

TargetLink's bus port blocks provide a way of handling the elements of a bus as separate inports. You can use this option if you want TargetLink to generate a separate variable for each bus element, for example, if a bus is pending at an external system that has a predefined function interface.

Generating struct variables for bus signals

TargetLinks bus port blocks provide the **Create bus struct** property.



If you select this checkbox, TargetLink maps the relevant busses to structure variables (structs) in the production code.

Mapping bus elements to predefined structs

Suppose you want to map a bus to a predefined struct. For example, a struct exists in custom code, and you want to assign the busses to the struct components. TargetLink lets you create the predefined struct in the dSPACE Data Dictionary (DD), and assign the bus elements to the struct components created in the DD.

For details, refer to *Defining the Representation of Busses in the Production Code* in the *TargetLink Advanced Practices Guide*.

Conditional Control Flow via Preprocessor Directives

TargetLink 2.1 offers a new method of adapting the generated production code and its parameters to different types of vehicles without changing the model: conditional code compilation. This is in addition to the variant coding of variable properties in the dSPACE Data Dictionary.

Conditional code compilation

You can select the subsystems to be included at code compilation time by means of preprocessor directives (function variants), that is, you can use preprocessor directives to specify the control flow that is used for code compilation. This is useful since it makes programs easier to develop, and easier to compile in different execution environments. For details, refer to *Specifying Conditional Control Flows* in the *TargetLink Advanced Practices Guide*.

External macros

If you do not want the Code Generator to define the macros for generating conditional control flows, you can use your own macros. For details, refer to *How to Include External Macros for Generating Conditional Control Flows* in the *TargetLink Advanced Practices Guide*.

Changes in Multirate Modeling

The following topics were changed or added in the *TargetLink Multirate Modeling Guide* since TargetLink 2.0.

First-time TargetLink buyers receive the latest print edition of the *TargetLink Multirate Modeling Guide*. Customers who have a software maintenance service (SMS) contract are requested to use the current version of the *TargetLink Multirate Modeling Guide* in the TargetLink HelpDesk.

Intertask communication

Access function TargetLink actually assigns all signals that are exchanged between root step functions to messages. This is also the case if a root step function receives signals from, or sends signals to, outside the TargetLink subsystem. Thus you can prevent TargetLink from assigning signals to messages at the boundaries of a TargetLink root system by assigning variable classes that reference an access function to the interface variables concerned. For details, refer to *Message Basics* in the *TargetLink Multirate Modeling Guide*.

Bus Port blocks TargetLink supports intertask communication for busses. You must specify the properties for the bus elements separately. For details, refer to *Message Basics* in the *TargetLink Multirate Modeling Guide*.

CounterTrigger port of the CounterAlarm block

If the counter is time-synchronous, you should connect the CounterTrigger inport to a function-call trigger source which resides outside the TargetLink subsystem. The sample time of this trigger source must be equal to the Tick duration in simulation value specified in the CounterAlarm block dialog. Since the trigger must not be activated at time 0, it is recommended to use a statechart to model the counter trigger. If you have not purchased Stateflow, you must model the trigger using a Simulink Function-Call Generator block, which must not be executed at simulation time 0. For details, refer to *Connecting Inports and Outports of the CounterAlarm Block* in the *TargetLink Multirate Modeling Guide*.

Name macros

You can define the names of tasks, ISRs, etc. manually or you can let TargetLink generate them automatically by using name macros. The `$(Time)`, `$(Period)` and `$(Offset)` name macros expand to the sample time and the offset during code generation. For details, refer to *Configuring Names in Multirate Models* in the TargetLink Multirate Modeling Guide.

Multirate limitations

Interrupted signal flow in specific subsystems If awkward subsystem sample times are chosen, the data flow between the subsytems may be interrupted and ignored by TargetLink.

Unit delay block for sample rate transitions If a Unit Delay block is used for sampling rate transition, you cannot switch logging on for it. Additionally, you must not specify saturation and the **Cast output signal to TargetLink type** property.

For details on multirate limitations, refer to *TargetLink Limitations of Multirate Modeling* in the *TargetLink Multirate Modeling Guide*.

Signals at TargetLink Subsystem Boundaries

The interface between a TargetLink subsystem and the remaining Simulink model is specified by TargetLink InPort/Bus Inport and OutPort/Bus Outport blocks. Each incoming and outgoing signal of a TargetLink subsystem must pass one of these TargetLink blocks first. These blocks do not need to reside on the topmost level of the TargetLink subsystem, they can reside in a subsystem on a lower level. Some preconditions must be fulfilled for this. For details, refer to *Interface Between a Simulink Model and a TargetLink Subsystem* in the *TargetLink Production Code Generation Guide*.

Look-Up Tables

Look-up function name macro

You can reuse a look-up table function for several TargetLink subsystems, via a template of the dSPACE Data Dictionary. The `$(FunctionName)` look-up function name macro lets you conveniently specify the template, for example, the name of the code file. For details, refer to *Example of a Table Function Reuse via Template* in the *TargetLink Advanced Practices Guide*.

Adaptation of table data

The table data vector (1-D table) or matrix (2-D table, Interpolation (n-D) using PreLook-Up (1-D or 2-D)) is quantized according to the associated scaling parameters if the table's data type is an integer. Elements that exceed the associated scaling range are saturated.

By default, the quantization effects of the table argument vectors are also taken into consideration, and the table data vector is adapted. This minimizes the deviation of the implemented table data from the original table data.

However, if a table data vector was used not only by its associated look-up table block, but also, for example, by a gain block, this correction would lead to an error. You can therefore disable the adaptation of the table data vector via the **Adapt table values** option on the Table page of the Look-Up Table 1D and 2D blocks. For details, refer to *Look-Up Table Block* and *Look-Up Table 2D Block* in the *TargetLink Block and Object Reference*.

Conforming Freestanding Implementation

The ISO C standard defines two classes of conforming implementation:

- A conforming hosted implementation supports the whole standard including all the library facilities.
- A conforming freestanding implementation is only required to provide certain library facilities: those in <float.h>, <limits.h>, <stdarg.h>, <stddef.h>, <iso646.h>, <stdbool.h> and <stdint.h>.

The conforming freestanding implementation does not include <math.h>, which contains the fabs() function. If you have to meet the requirements of conforming freestanding implementation, TargetLink provides a special implementation of the fabs() function in the tl_fabs.h file. For details, refer to *SrcFiles* in the *TargetLink File and Message Reference*.

Block Library and Block Properties

New simulation blocks

Bus port blocks TargetLink supports the Bus Inport and Bus Outport blocks, which let you specify the representation of busses in the production code. For details, refer to the following topics:

- *Representation of Busses in the Production Code* on page 16
- *Defining the Representation of Busses in the Production Code* in the *TargetLink Advanced Practices Guide*

Enhanced block properties

Addfile block In TargetLink's Addfile block, you can specify the absolute or relative path and the name of the file to be added to the production code. A complete description of a code file consists of <path><basename>.<extension>. The permitted path separators are '/' or '\'. The file name can contain letters, digits, underscores, and hyphens, and begin with any character. For details, refer to *Addfile Block* in the *TargetLink Block and Object Reference*.

Constant block The Constant block lets you specify the MATLAB expression that returns the current sample time, and the Cast output signal to TargetLink type property. For details, refer to *Constant Block* in the *TargetLink Block and Object Reference*.

Math block TargetLink 2.1 supports arbitrary scaling of rem and mod functions. It converts arbitrary scalings into integer representations. Arbitrary and integer representations can slightly differ. For the rem and mod functions of the Math block, however, even a small difference can lead to invalid results. If no exact integer representation of an arbitrary LSB value can be found, a warning is given by TargetLink's Code Generator, and you should either apply power-of-two scaling, or modify the arbitrary LSB value. For details, refer to *Math Block* in the *TargetLink Block and Object Reference*.

Changes in Stateflow

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

Events on state machine level

TargetLink now supports events on state machine level. There are limitations concerning the propagation of events. Refer to *Limitations in Creating a TargetLink Subsystem* in the *TargetLink Production Code Generation Guide*.

Bind actions

TargetLink now supports bind actions, introduced in Stateflow 6.0. There are limitations concerning state and output reset that apply to function-call triggered subsystems called by bound events. Refer to *Limitations in Creating a TargetLink Subsystem* in the *TargetLink Production Code Generation Guide*.

Target Boards

dSPACE offers the following new target boards:

Freescale S12X

Freescale S12X, based on the Freescale S12X microcontroller family member MC9S12XDP512. For details, refer to *Freescale S12X* in the *TargetLink TargetReference*.

Infineon TriCore (TBTC1796)

Infineon TriCore (TBTC1796), based on the Infineon TriCore microcontroller family member TC1796. For details, refer to *Infineon TriCore (TBTC1796)* in the *TargetLink TargetReference*.

STMicroelectronics ST10

STMicroelectronics ST10, based on the STMicroelectronics microcontroller family member ST10F276. For details, refer to *STMicroelectronics ST10* in the *TargetLink TargetReference*.

NEC V850ES

NEC V850ES, based on the NEC V850ES microcontroller family member V850ES/FJ2 UPD70F3239. For details, refer to *NEC V850ES* in the *TargetLink TargetReference*.

Freescale PowerPC MPC5500

Freescale PowerPC MPC5500, based on the Freescale PowerPC MPC5500 (MPC55xx) microcontroller family member MPC5554. For details, refer to *Freescale PowerPC MPC5500* in the *TargetLink TargetReference*.

Overview of Microcontrollers, Boards, and Compilers

The following table shows the combinations of microcontrollers, boards, and compilers as supported by TargetLink 2.1 (TargetLink abbreviations). For details, refer to *Supported Targets* in the *TargetLInk Target Reference*.

Microcontroller Family	Board	Compiler
C16x	Promo167	Task51 Task60 Task75 Task80 Task85
H8S/26xx	EVB2633F	Hit30 Hit60
	EVB2655	Hit30
HC12	HC12EVB	Cosmic42 Cosmic44 Cosmic45
HCS12	HCS12TBoard HCS12EVB	Cosmic44 Cosmic45 Cosmic46 Met12 Met20 Met31
S12X	S12XEVB	Cosmic46
M32R	MSA2114	Mcc32r20 Mcc32r43 Gaio913
MPC5xx	CME555	Diab43 Diab44 Diab50 Diab52 GHS30 GHS35 GHS36 GHS40 Met60 Met81
	CMD565	Diab50 Diab52
MPC55xx	MPC5554DEMO	Diab52 GHS40
NEC V850ES	DI_V850F3239	GHS35 GHS40
SH2	SH2eEVB EVB7058	Hit41 Hit50 Hit51 Hit60 Hit70 Hit80

Microcontroller Family	Board	Compiler
ST10F276	START276	Task75 Task80 Task85
TI TMS470R1x	EVB470R1	TIccs13 TIccs21 TIccs221
TriCore112	TBTC1775	Task11 Task13 Task14 Task15 Task22
TriCore113	TBTC1796	Task22

New and Enhanced API Commands

In TargetLink 2.1, the following API commands are new or have been enhanced.

New API commands

tl_code_cov_preferences_hook You can change the code coverage HTML output to your preferences using the `tl_code_cov_preferences_hook` API command. For details, refer to *tl_code_cov_preferences_hook* in the *TargetLink API Reference*.

Enhanced API commands

tl_code_coverage You can remove code coverage marks from the production code using the `RemoveAllCCMarks` option of the `tl_code_coverage` API command. For details, refer to *tl_code_coverage* in the *TargetLink API Reference*.

sl2tl The `sl2tl` API command offers the following new options:

- `InsertPorts` inserts TargetLink ports at the root level of the subsystem.
- `InsertPortsOnly` inserts TargetLink ports at root level and leaves all other blocks untouched.
- `ConvertLibBlocks` allows a subsystem which is to be converted to be a library block whose link is deactivated.
- `SetCastOutputFlag` sets the output.castoutput property of the replacing TargetLink block to 'on' if the block supports this property and if the output of a Simulink block which was replaced is an unscaled integer type.

For details, refer to *sl2tl* in the *TargetLink API Reference*.

sllib2tllib The `sllib2tllib` API command offers the ConvertLibBlocks option, which allows subsystems which are to be converted to be library blocks (in another library) whose link is deactivated. For details, refer to *sllib2tllib* in the *TargetLink API Reference*.

tl_build_host You can generate production code for all specified TargetLink subsystems, and optionally for all their nested subsystems configured for incremental code generation by activating the GenerateAll property. For details, refer to *tl_build_host* in the *TargetLink API Reference*.

tl_build_standalone You can generate production code for all specified TargetLink subsystems, and optionally for all their nested subsystems configured for incremental code generation by activating the GenerateAll property. For details, refer to *tl_build_standalone* in the *TargetLink API Reference*.

tl_build_target You can generate production code for all specified TargetLink subsystems, and optionally for all their nested subsystems configured for incremental code generation by activating the GenerateAll property. For details, refer to *tl_build_target* in the *TargetLink API Reference*.

tl_generate_code You can generate production code for all specified TargetLink subsystems, and optionally for all their nested subsystems configured for incremental code generation by activating the GenerateAll property. For details, refer to *tl_generate_code* in the *TargetLink API Reference*.

tl_pre_compile_host_hook In TargetLink 2.1 this command is now similar to tl_pre_compile_target_hook, except that the MEX compiler is set instead of the simConfig option, which is not available in SIL mode. For details, refer to *tl_pre_compile_host_hook* in the *TargetLink API Reference*.

tl_post_compile_host_hook In TargetLink 2.1 this command is now similar to tl_post_compile_target_hook, except that the MEX compiler is set instead of the simConfig option, which is not available in SIL mode. For details, refer to *tl_post_compile_host_hook* in the *TargetLink API Reference*.

Miscellaneous Features

File name conventions

C file names can contain letters, digits, underscores, and hyphens, and begin with any character. You can specify C file names, for example, in the Function or Addfile block, or via the Module property of a DD object. For details, refer to *Configuring Names* in the TargetLink Advanced Practices Guide.

Property Manager

Block Explorer The following details of the Block Explorer of TargetLink's Property Manager are modified:

- In the Block Explorer, the rows and columns are seperated by lines.
- If the Block Explorer is scrolled horizontally, the name column does not change its position.
- Gray fields in the Block Explorer show that the property is not editable for the specific block.
- You can edit vectorized block variables. For example, this applies to Custom Code blocks with more than one output, state variables, and to bus port blocks.

DD objects You can now create DD Variable, Scaling, and Message objects via Property Manager.

ModuleTemplate

You can use the ModuleTemplate to:

- Specify the names of modules which are user-defined via the Function block or the Module property of DD objects
- Specify if a module is included via system or user-defined include
- Specify the file name extensions of source and header files
- Suppress the generation of the module

For details, refer to *ModuleTemplate* in the *dSPACE Data Dictionary Manager Reference*.

MACRO variable class for vectors

As of TargetLink 2.1, vector variables can be defined with the MACRO variable class. The name for an n-dimensional vector is expanded to `<name>_1 .... <name>_n` if it is used as a parameter. A separate macro symbol is created for each element of the vector.

Plot windows

If available, the units of stack size, execution time, and variables are displayed in plot windows.

Code coverage level names

The names of code coverage levels have changed.

Code coverage level	Name
Statement coverage	C0
Decision coverage	C1

Model checksum

The TargetLink Main Dialog provides the **Add checksum to code** property on its Options page. If this checkbox is selected, a checksum is placed in the file header of the generated code. It lets you check if a model was changed after the last code generation.

Unicode character sets

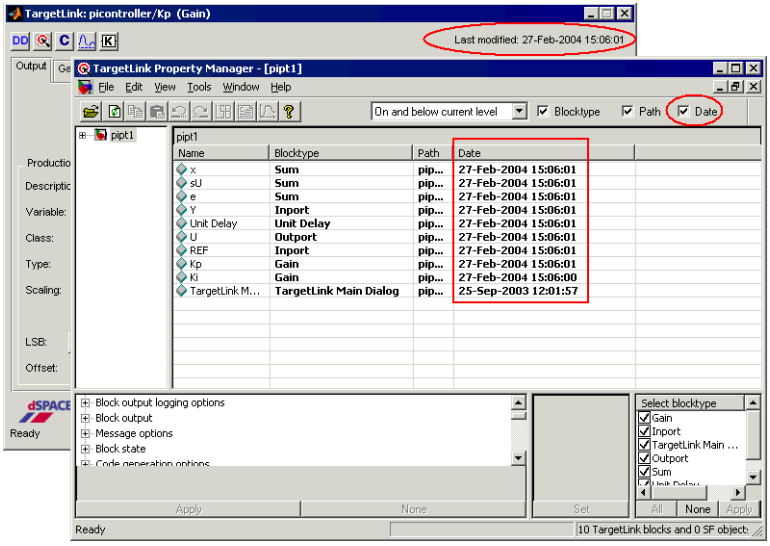
TargetLink does not support the complete Unicode Standard, but specific parts of it, to display characters, which are encoded by 1 or 2 bytes. Since TargetLink is closely coupled to MATLAB, it follows MATLAB's encoding of character sets. TargetLink stores strings as a sequence of 16-bit integer values, each of which is the code of one character. This allows characters to be encoded according to the following schemes:

- ASCII strings
- Double-byte character sets (DBCS)
- Unicode character sets that do not use 4-byte integers
- The Unicode Character Code Charts Hiragana and Katakana

TargetLink uses the character set of the locally installed operating system. For details, refer to *Using Different Character Sets* in the TargetLink Advanced Practices Guide.

Apply button in TargetLink block dialogs

The Apply button in TargetLink block dialogs writes any changed block settings to the block. If you have not changed any settings, no data is written to the block. Thus, you can sort the TargetLink blocks by date in the Property Manager to track the blocks whose settings you changed recently.



Refer to *Property Manager* in the *TargetLink Tools and Utility Reference*.

Suppression of shift operations

By default, TargetLink tries to implement shift operations instead of multiplications and divisions. This makes the generated production code more efficient. However, sometimes you might want to suppress shift operations, for example, to comply with MISRA rule #37 (MISRA-C:1998 Standard), which does not allow bit operations with signed data. You can use the `tl_pre_codegen_hook.m` file to force the Code Generator to suppress right and/or left shifts. For details, refer to *Suppression of Shift Operations* in the *TargetLink Production Code Generation Guide*.

Zero-based indexing

TargetLink supports the use of index 0 for the first input of the following blocks:

- Assignment
- Multiport Switch
- For Iterator
- Selector

For example, refer to *Assignment Block* in the *TargetLink Block and Object Reference*.

Autoscaling behavior

For autoscaling (use of simulation data), all inports and outports must be nonvirtual, as otherwise no simulation data is available for autoscaling.

Importing graphics in generated documents

TargetLink 2.1 supports a new option for customizing your documentation via the AutoDoc Customization block. You can add graphics (as supported by your browser), by using the **`$insertimage:<filepath>`** command. For details, refer to *How to Include User-Defined Information in the Documentation* in the *TargetLink Production Code Generation Guide*.

Migrating to TargetLink 2.1

When migrating from TargetLink Release 2.0 to TargetLink 2.1, you should be aware of the following aspects:

Migrating from TargetLink 2.0 or Earlier

When migrating from former TargetLink releases, for example, from TargetLink Release 1.3 to 2.1, you have to migrate step by step via the intervening TargetLink versions. Consequently, it is recommended to read the relevant New Features And Migration Guide(s) beforehand. These guides contain information that is essential for migration.

If you want to migrate from TargetLink version 1.2 to TargetLink version 2.0, you have to follow the migration steps given in:

1. New Features and Migration of TargetLink version 1.3
2. New Features and Migration of TargetLink version 2.0
3. Finally, the migration steps described in this document.

Accessing new features and migration documents

You can find the PDF files of the New Features and Migration documents for previous releases in the \Doc\Print folder on the dSPACE CD or download them from http://www.dspace.de/goto?migration_tl.

The PDF files are named NewFeaturesAndMigrationxx.pdf, where xx stands for the version or release number.

Upgrading Data Dictionary Project Files

Due to enhancements in the data model of dSPACE Data Dictionary version 1.3, you have to upgrade DD project files you created with dSPACE Data Dictionary version 1.1. For detailed information, refer to *Migrating from dSPACE Data Dictionary 1.1* in dSPACE Data Dictionary New Features and Migration.

Exporting TargetLink Block Variables to the dSPACE Data Dictionary

If you migrate from TargetLink version 1.3 or earlier to TargetLink version 2.1, you have to export block data from TargetLink models created with TargetLink version 1.3 to the dSPACE Data Dictionary. You can use an M script for these purposes. For a detailed description, open %DSPACE_ROOT%/doc/print/AppNote_Mod2DD.pdf.

Migration Features

Optimization of function classes

The default value of the Optimization property of function classes is now empty. This might be important if you have function classes created automatically.

Last-Minute Information

This chapter provides information on changes and enhancements that were made after the TargetLink documentation was completed.

TargetLink Production Code Generation Guide

For information on last-minute changes concerning production code generation, refer to *Changes on the TargetLink Production Code Generation Guide* on page 42.

TargetLink API Reference

For information on last-minute changes concerning the TargetLink API, refer to *Changes to the TargetLink API Reference* on page 43.

Changes on the TargetLink Production Code Generation Guide

This chapter contains information on changes that were made after the TargetLink Production Code Generation Guide was completed.

Simulated ranges

Autoscaling based on simulated ranges and a netlist uses not only the simulated ranges to determine the block scaling, but also propagation from the surrounding blocks. This can result in a scaling that does not cover the simulated range, i.e., at switch input, if the input signal does not pass the switch during simulation. You should check the model structure in this case.

Access functions

Suppose a variable is assigned the EXTERN_MACRO variable class and you want to access the variable via access macro. If you have defined a specific body (implementation) for the access macro, TargetLink cannot build a simulation frame for SIL or PIL simulation. A warning message is displayed.

Changes to the TargetLink API Reference

This chapter contains information on changes that were made after the TargetLink API Reference was completed.

tl_get_checksum

You can use the API command to maintain consistency between your model and the generated code by comparing the checksum of the model with the checksum written to the code by the Model checksum feature (refer to *Miscellaneous Features* on page 31).

tl_get_checksum

To calculate the checksum of a Simulink model file or an arbitrary ASCII file.

Syntax

```
[checksum,errorflag,msg] = tl_get_checksum(objectKind,fileName)
```

Input parameters

The following input parameters are available:

Parameter	Description
objectKind	Kind of object whose whose checksum maust be calculated.
	Model The checksum is calculated from a Simulink Model.
	txt The checksum is calculated from an arbitrary ASCII file.
fileName	Name of the file whose checksum is calculated.

Output parameters

The following output parameters are available:

Parameter	Description
checksum	Checksum of the selected file
errorflag	Flag indicating success or failure.
	0 no error
	1 invalid object kind
	2 file name is no string or an empty string
	3 File is not accessable
	4 Error executing tl_get_checksum.exe
msg	Error or warning message (empty on success)

```
chkSum = tl_get_checksum('Model','MyModel.mdl');
chkSum = tl_get_checksum('txt','tl_pre_codegen_hook.m');
```

Key Features of the MATLAB R14SP3 Compatibility Update for TargetLink 2.1

Where to go from here

Information in this section

General Enhancements and Changes

Introduction

The MATLAB R14SP3 Compatibility Update for TargetLink 2.1 enables you to use TargetLink 2.1 together with MATLAB® and Simulink® on the basis of The MathWorks *Release 14 with Service Pack 3*. For simplicity, the term *MATLAB R14SP3* is used to refer to The MathWorks Release 14 with Service Pack 3 throughout the rest of this document.

MATLAB R14SP3 provides several new features. This document describes which features are supported when you work with TargetLink. Note that the MATLAB R14SP3 Compatibility Update for TargetLink 2.1 does not support all the new MATLAB R14SP3 features and that its primary aim is compatibility. For details on the limitations that apply when you work with the MATLAB R14SP3 Compatibility Update for TargetLink 2.1, refer to *Limitations of the MATLAB R14SP3 Support in TargetLink 2.1* on page 51.

When you work with the MATLAB R14SP3 Compatibility Update for TargetLink 2.1, it is assumed that you have knowledge in handling TargetLink, and that you know the TargetLink Production Code Generation Guide and the TargetLink Advanced Practices Guide. It is also assumed that you know the Release Notes for Release 14 with Service Pack 3 for MATLAB, Simulink, and Stateflow.

Compatibility of TargetLink models

TargetLink models created with TargetLink 2.1 provide full functionality when you work with the MATLAB R14SP3 Compatibility Update for TargetLink 2.1. Nevertheless, you should be aware of the limitations described in *Limitations of the MATLAB R14SP3 Support in TargetLink 2.1* on page 51.

Models you saved with TargetLink 2.1.5 can still be opened with TargetLink 2.1 if saved with the proper Simulink model file format. If you saved your model with Simulink 6.3 and open it with Simulink 6.2 later on, an error message indicates that unknown Simulink properties were used.

Related topics

Basics	*Limitations of the MATLAB R14SP3 Support in TargetLink 2.1* on page 51
HowTos	–
Examples	–
References	–

Supported MATLAB Features

New MATLAB preferences directory

With MATLAB R14SP3, a new preferences directory named R14SP3 is available. The preferences directory MATLAB uses depends on the release. Targetlink preferences, for example information on the dSPACE Data Dictionary in use, are also stored in the directory.

The procedure used from R13 through R14SP2 was changed for R14SP3 to address downward compatibility problems. You can obtain the name of the current preferences directory from MATLAB by running the `prefdir` function in the MATLAB Command Window.

The differences are relevant primarily if you run multiple versions of MATLAB. If you run only R14SP3, or run R14SP3 with R13 or R12 releases, you will not be affected:

- When you install R14SP3, MATLAB migrates files from your existing R14 preferences directory to the new directory, R14SP3. Changes made to files in this directory are not used if you run previous R14 releases. As an example, commands you run in R14SP3 will not appear in the Command History when you run R14SP2, and vice versa.
- For all other releases from R13 through R14SP2, the associated preferences directory is always used. For example, R13 and R13SP1 share the R13 preferences directory. However, any changes will not be used when you run R14 releases because R14 releases do not use the R13 preferences directory. Similarly, any changes made to files in the preferences directory while R14 releases are running are not used when R13 is run.

Related topics

Basics	*Limitations of the MATLAB R14SP3 Support in TargetLink 2.1* on page 51
HowTos	–
Examples	–
References	–

Supported Simulink Features

Input Port latching enhancements

MATLAB R14SP3 includes the following enhancements to the signal latching capabilities of the Inport block:

- Label clarified for triggered subsystem latch option (see below)
- Latch option for function-call subsystems

  This option is not supported by TargetLink. For details, refer to *Unsupported Simulink Features* on page 52.

Label clarified for triggered subsystem latch option The dialog box for an Inport block contains a checkbox to latch the signal connected to the system via the port. This checkbox applies only to triggered subsystems and hence is enabled only when the Inport block resides in a triggered subsystem. In this release, the label for the check box that selects this option has changed from *Latch (buffer) input* to *Latch input by delaying outside signal*. This change is intended to make it clear what the option does, i.e., cause the subsystem to see the input signal's value at the previous time step when the subsystem executes at the current time step (equivalent to inserting a Memory block at the input outside the subsystem). The Inport block's icon displays **<Lo>** to indicate that this option is selected.

Related topics

Basics	*Unsupported Simulink Features* on page 52 *Unsupported New Simulink Blocks* on page 54
HowTos	–
Examples	–
References	–

Supported Stateflow Features

Specifying execution order of parallel states explicitly

MATLAB R14SP3 gives you the ability to specify the execution order of parallel states explicitly in Stateflow charts. Previously, the execution order of parallel states was determined solely by implicit rules, based on geometry. A disadvantage of implicit ordering is that it creates a dependency between design layout and execution priority. When you rearrange parallel states in your diagram, you may inadvertently change the order of execution and affect the simulation results. Explicit ordering gives you more control over your designs.

See *Execution Order for Parallel States* in the Stateflow Semantics (Using Stateflow) chapter in the Stateflow documentation.

Related topics

Basics	*Unsupported Stateflow Features* on page 55
HowTos	–
Examples	–
References	–

Limitations of the MATLAB R14SP3 Support in TargetLink 2.1

Where to go from here

Information in this section

Unsupported Simulink Features

Latch option for function-call subsystems

MATLAB R14SP3 provides a checkbox labeled *Latch input by copying inside signal* in the Inport block's dialog box. This option applies only to function-call subsystems and hence is enabled only if the *Inport* block resides in a function-call subsystem. Selecting this option causes Simulink to copy the signal output by the block to a buffer before executing the contents of the subsystem and to use the copy as the block's output during execution of the subsystem. This ensures that the subsystem's inputs, including those generated by the subsystem's context, will not change during execution of the subsystem. The Inport block's icon displays **<Li>** to indicate that this option is selected.

This option is not supported by TargetLink. Selecting it will cause TargetLink to display the following error message:

E20100 Currently *Latch input by copying inside signal* is not supported by TargetLink.

Additional reset trigger for Discrete-Time Integrator block

MATLAB R14SP3 provides a *sampled level* trigger option which resets the output of the Discrete-Time Integrator to the initial condition when the reset signal is nonzero. The new reset trigger is more efficient than the *level reset* option, but may introduces a discontinuity when integration resumes.

Neither the *sampled level* trigger option nor the *level trigger* option is supported by TargetLink. If this Simulink option is selected before you start a Simulink to TargetLink conversion, the option is set to *either* by the Model Converter automatically. This is indicated by a corresponding message.

Custom signal viewers and generators

MATLAB R14SP3 allows you to add custom signal viewers and generators so that you can manage them in the Signal & Scope Manager.

TargetLink does not support the Signal & Scope Manager, so the Code Generator ignores signal viewers and generators. Inputs and Outputs that are connected which such global objects are treated as if they were not connected.

Related topics

Basics	*General Enhancements and Changes* on page 46 *Supported Simulink Features* on page 49
HowTos	–
Examples	–
References	–

Unsupported New Simulink Blocks

Truth Table block

A Truth Table block is now available as a Stateflow element in the Simulink library. You can call a truth table function directly from your Simulink model with this new block.

TargetLink does not support the Truth Table block. If you want to use truth table functions with TargetLink, include a Stateflow chart with a truth table instead. If a Truth Table block is identified in your model, the following error message will be displayed.

E20937 Truth table charts are currently not supported.

Related topics

Basics	*General Enhancements and Changes* on page 46 *Supported Simulink Features* on page 49
HowTos	–
Examples	–
References	–

Unsupported Stateflow Features

Sharing global data between Simulink and Stateflow

MATLAB R14SP3 provides an interface that gives Stateflow charts access to global variables in Simulink models. Simulink implements global variables as data stores, created either as data store memory blocks or as instances of *Simulink.Signal* objects. Now Stateflow charts can share global data with Simulink by reading and writing data store memory symbolically using the Stateflow action language.

This feature is not supported by TargetLink. If you try to access global data, for example specified in *Data Store Memory* blocks, using the Stateflow action language from within TargetLink, the following error message will be displayed:

E20941 Stateflow data with scope data store memory is currently not supported.

Using embedded MATLAB action language in truth tables

MATLAB R14SP3 provides an option that allows you to use the embedded MATLAB action language in Stateflow truth tables. Previously, you were restricted to the Stateflow action language.

This option is not supported by TargetLink. If you use the MATLAB action language anyway, the following error message will be displayed:

E20885 Evaluating MATLAB expressions from within Stateflow diagrams is not supported.

Related topics

Basics	*General Enhancements and Changes* on page 46 *Supported Stateflow Features* on page 50
HowTos	–
Examples	–
References	–