

TargetLink

Code Generation Reference for MATLAB[®] Code in Simulink[®] Models

For TargetLink 5.1

Release 2020-B – November 2020

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

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

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

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

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

Software Updates and Patches

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

Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not 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.

© 2013 - 2020 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

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

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

The ability of dSPACE TargetLink to generate C code from certain MATLAB code in Simulink®/Stateflow® models is provided subject to a license granted to dSPACE by The MathWorks, Inc. MATLAB, Simulink, and Stateflow are trademarks or registered trademarks of The MathWorks, Inc. in the United States of America or in other countries or both.

Contents

- About This Reference 5
- Supported MATLAB® Code Functions and Operations 9
 - Supported MATLAB® Code Function Statements and Function Operators..... 9
- API Functions for MATLAB® Code 15
 - tlCreateMATLABFunctionDDObjects..... 16
 - tlCreateMATLABFunctionDDObjects..... 16
 - tl_get_mlfcnobjects..... 18
 - tl_get_mlfcnobjects..... 18
- Index 19

About This Reference

Content

Provides information on how to add MATLAB code functions to Simulink® models for a script-based implementation of function algorithms.

Note


The ability of dSPACE TargetLink to generate C code from certain MATLAB code in Simulink®/Stateflow® models is provided subject to a license granted to dSPACE by The MathWorks, Inc.

Target audience

This information is targeted at function developers and software specialists who want to integrate MATLAB functions into Simulink® models.









Required knowledge

It is assumed that you are familiar with MATLAB, the MATLAB language, Simulink®, and Stateflow®.

Orientation and Overview Guide For an introduction to the use cases and the TargetLink features that are related to them, refer to the  [TargetLink Orientation and Overview Guide](#).

Symbols

dSPACE user documentation uses the following symbols:

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

Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

Special folders

Some software products use the following special folders:

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files


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

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

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

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

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

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

Supported MATLAB® Code Functions and Operations

Supported MATLAB® Code Function Statements and Function Operators

Supported functions and operators

Not all MATLAB functions are supported by TargetLink. The following tables list all functions that are fully or partially supported. MATLAB functions not listed here are not supported.

Note

If the Code Generator finds an unsupported operation or function in a MATLAB function for which code has to be generated, TargetLink aborts code generation and displays an error message.

Supported functions and operators

Operation / Function	Description
for	For loop statement <pre>for index=loopexpression loopbody end</pre> <p>Inside the <i>For loop</i> expression, all kinds of ranges, such as <code>2:in</code>, are supported.</p>
if	If statement
switch	Switch statement
while	While statement
end	End of for , while , switch and if Also: highest index of a dimension, e.g., <code>A(2:end)</code> Corresponds to <code>A(2:length(A))</code>
+ <code>plus()</code>	Addition (element wise)
- <code>minus()</code>	Subtraction (element wise)
* <code>mtimes()</code>	Multiplication (matrix)





Operation / Function	Description
<code>.* times()</code>	Multiplication (element wise)
<code>./ rdivide()</code>	Division (element wise)
<code>.\ ldivide()</code>	Array left division (element wise)
<code>.^ power()</code>	Raising to a scalar power (element wise)
<code>[]</code>	Multi-assignment <code>[a b] = f(in1, in2)</code>
<code>' ctranspose()</code>	Complex conjugate transpose
<code>.' transpose()</code>	Transpose
<code>.</code>	Decimal-dot
<code>,</code>	Comma used as separator for: <ul style="list-style-type: none"> ▪ Row element separator ▪ Array index separator ▪ Function input and output separator ▪ & command or statement separator
<code>...</code>	Ellipsis: Line continuation including comments in statements Comments after the ellipsis are displayed in the generated code.
<code>%</code>	Percent: Comment Comments are shown in the generated code.
<code>%{</code> <code>%}</code>	Percent-Brace: Block-Comment Comments are shown in the generated code.
<code>< lt</code>	Less than (element wise)
<code><= le</code>	Less than or equal to (element wise)
<code>>= ge</code>	Greater than or equal to (element wise)
<code>> gt</code>	Greater than (element wise)
<code>== eq</code>	Equal (element wise)
<code>~= ne</code>	Not equal (element wise)
<code>& and</code>	Logical AND
<code> or</code>	Logical OR
<code>~ not</code>	Logical NOT
<code>- uminus</code>	Unary minus
<code>+ uplus</code>	Unary plus
<code>&&</code>	Logical AND (short-circuiting)
<code> </code>	Logical OR (short-circuiting)
<code>xor</code>	Logical XOR
<code>=</code>	Assignment
<code>abs</code>	Abs
<code>acos</code>	Inverse cosine in radians
<code>acosd</code>	Inverse cosine in degrees

Operation / Function	Description
<code>asin</code>	Inverse sine in radians
<code>asind</code>	Inverse sine in degrees
<code>atan</code>	Inverse tangent in radians
<code>atand</code>	Inverse tangent in degrees
<code>atan2</code>	Four quadrant inverse tangent
<code>bitand</code>	Bitwise AND
<code>bitcmp</code>	Bitwise complement
<code>bitor</code>	Bitwise OR
<code>ceil</code>	Round towards plus infinity
<code>cos</code>	Cosine in radians
<code>cosd</code>	Cosine in degrees
<code>cosh</code>	Hyperbolic cosine
<code>double</code>	Convert to double precision
<code>exp</code>	Exponential
<code>floor</code>	Round towards minus infinity
<code>int8, int16, int32</code>	Casting
<code>isequal</code>	True if arrays are numerically equal
<code>log</code>	Natural logarithm
<code>logical</code>	Converts numerical values to logical values
<code>mod</code>	Modulus after division
<code>persistent</code>	<p>Makes an implicit MATLAB script variable persistent. Only supported with the following fixed code pattern:</p> <pre> persistent PROD_X if isempty(PROD_X) PROD_X = <constant value expression>; end </pre> <p>The pattern has to be on the top level of the function, i.e., it must not be inside <code>if/else</code> branches or loops. Only constant value expressions (Glossary) are supported for initialization.</p>
<code>pi</code>	3.1415926535897
<code>sign</code>	Signum function
<code>sin</code>	Sine in radians
<code>sind</code>	Sine in degrees
<code>sinh</code>	Hyperbolic sine
<code>single</code>	Convert to single precision
<code>sqrt</code>	Square root
<code>tan</code>	Tangent in radians
<code>tand</code>	Tangent in degrees

Operation / Function	Description
<code>tanh</code>	Hyperbolic tangent
<code>uint8</code> , <code>uint16</code> , <code>uint32</code>	Casting

Partially supported functions and operators

Operation / Function	Description	Comments and Details
<code>/</code> <code>mrdivide()</code>	Division (slash or matrix right)	Supported only for <i>scalar*scalar</i> , <i>scalar*vector</i> , and <i>scalar*matrix</i> .
<code>\</code> <code>mldivide()</code>	Division (backslash or matrix left)	Supported only for <i>scalar*scalar</i> , <i>scalar*vector</i> , and <i>scalar*matrix</i> .
<code>^</code> <code>mpower()</code>	Matrix power	Supported only for scalar operands.
<code>[]</code>	Concatenation of matrices	<p>The level of support depends on the environment in which the concatenation is located.</p> <p><i>Assignments and Function Arguments</i></p> <pre>a = [...] myFcn(..., [...], ...)</pre> <p>Fully supported in assignments and as function arguments.</p> <p>Not supported for:</p> <ul style="list-style-type: none"> Variables with exchangeable widths inside concatenations <pre>a = [myVar 5.1*graphicalFcn(in); -[7; 8]'] myFunction(-[4 myArray(1)])</pre> <p><i>Elsewhere (persistent initializations, complex expressions, conditions, ...)</i></p> <p>In all other environments, such as complex expressions or conditions, concatenation is supported only if it is a constant value expression (Glossary).</p>

Operation / Function	Description	Comments and Details
(r, c)	Selection / indexing: The selection might have several arguments. The arguments can be scalar, vector, or matrix.	Supported if the arguments of the selection are scalar and do not have logical Simulink data type. If one argument is not scalar, this argument has to be one of the following: <ul style="list-style-type: none"> ▪ A supported range or a full range (without start/end) ' '. Refer to the : <i>Range</i> part. ▪ A constant concatenation. Refer to the Concatenation of matrices – Elsewhere (complex expressions, conditions, ...) part. ▪ A scalar variable added to a supported concatenation, e.g., <code>scalarVar+[4 1 3:C]</code> Not supported in the following cases: <ul style="list-style-type: none"> ▪ An indexed vector/matrix is not a variable, e.g., function return, <code>f(in)(3)</code>. ▪ Logical Simulink data types except for: <ul style="list-style-type: none"> ▪ Argument that are literals such as <code>true</code> or <code>[false true true]</code> ▪ No element is selected. This results in an empty matrix, which is not supported by TargetLink.
:	Range Start:End Start:Step:End	For ranges in For-loops. For more information, refer to the for operation in <i>Supported functions and operators</i> . Ranges in concatenations and in initializations of persistent variables are also restricted. For more information, refer to Concatenation of matrices. In all other environments, only the following cases are supported: <ul style="list-style-type: none"> ▪ Start/Step/End are supported for scalar  constant value expression ( Glossary). ▪ Identical scalar variables in numerical ranges (Start:Step:End) together with an addition or subtraction of  constant value expression ( Glossary), for example: <code>x:(x+7)</code> <code>y:-2:(y-length(a))</code> <code>z-5:z</code> Not supported for: <ul style="list-style-type: none"> ▪ Empty ranges (resulting from no elements), for example: <code>6:5</code>
.	Dot: Structure component access (cannot be used as Stateflow path access)	Access to structure components Not supported: <ul style="list-style-type: none"> ▪ Simulink.Bus objects for local MATLAB variables ▪ Struct creation using the dot notation
all	True if all elements of a vector are nonzero.	All or vectordim as second argument is not supported.

Operation / Function	Description	Comments and Details
any	True if any element of a vector is a nonzero number.	All or vectordim as second argument is not supported.
bitshift	Value of A shifted by K bits	K has to be a scalar constant value expression (Glossary). Not supported: <ul style="list-style-type: none"> ▪ Bitshift with 3 arguments
cast	Cast a variable to a different data type	Casts are only supported for Simulink types that match a TargetLink base type. Not supported: <ul style="list-style-type: none"> ▪ Cast with argument like
false	False array	Supported only if arguments are constant value expressions (Glossary).
isempty	True for empty array	Supported only for the initialization of persistent variables with the following fixed code pattern: <pre>if isempty(PROD_X) PROD_X = 1; end</pre> The pattern has to be at the top level of the function, i.e., it must not be in <i>if/else</i> branches or <i>loops</i> . For initialization, only numerical constants are supported (1, 2:5, [1 3 5:9], ...).
length	Length of vector	Not supported: <ul style="list-style-type: none"> ▪ Variable vector width
max	Largest component	Supported only for two scalar or two vectorial parameters (NaN behavior options are ignored because they are limited).
min	Smallest component	Supported only for two scalar or two vectorial parameters (NaN behavior options are ignored because they are limited).
ones	Ones array	Supported only if arguments are constant value expressions (Glossary).
size	Size of array	Not supported: <ul style="list-style-type: none"> ▪ If size is called with two arguments and the second argument is no constant value expression (Glossary). ▪ If the first argument is configured for variable vector width.
true	True array	Supported only if arguments are constant value expressions (Glossary).
zeros	Zeros array	Supported only if arguments are constant value expressions (Glossary).

API Functions for MATLAB® Code

Where to go from here

Information in this section

tlCreateMATLABFunctionDDObjects.....	16
tl_get_mlfcnobjects.....	18

tlCreateMATLABFunctionDDObjects

tlCreateMATLABFunctionDDObjects

Purpose

Generates DD objects for specifying internal MATLAB code variables and MATLAB sub-functions.

Description

This function automatically creates all objects in the Data Dictionary that are required to specify internal MATLAB code variables and MATLAB sub-functions. It connects MATLAB Function blocks and Stateflow MATLAB functions in the model to their corresponding DD objects via the matlabfunction property. All actions that modified the Data Dictionary during one execution of the function are logged to a file.

Syntax

```
msgStruct = tlCreateMATLABFunctionDDObjects(propertyName, propertyValue, ...)
```

Property value pairs

Additional function parameters are expected as propertyName/propertyValue pairs, refer to the following table:

Property	Description
XmlFiles	List of XML files containing information to create the appropriate DD objects. These XML files are generated by the Code Generator if the OutputMATLABCodeInfo Code Generator option is active. The Code Generator uses the following pattern for the XML file names: MATLABCodeInfo_<subsystem ID>_<primary function name>.xml.
Model	Model name or handle. If no model is specified, the current model is used.
AutoSave	Specifies whether to automatically save modified models. This also applies to referenced models. The following values are possible: <ul style="list-style-type: none"> 'on' <ul style="list-style-type: none"> - Automatically saves modified models. 'off' <ul style="list-style-type: none"> - Modified models are not saved automatically. Default: 'off'

Property	Description
AutoDelete	<p>Specifies whether to automatically delete DD objects that are no longer used for the execution of this function. The following values are possible:</p> <ul style="list-style-type: none"> ▪ 'on' <ul style="list-style-type: none"> - Automatically deletes obsolete DD objects. ▪ 'off' <ul style="list-style-type: none"> - Generates a log file entry stating that obsolete objects exist. This option affects only DD objects in the following DD trees: <ul style="list-style-type: none"> ▪ /Pool/Variables/MATLAB_LocalVariables ▪ /Pool/Functions/MATLAB_Subfunctions <p>Default: 'off'</p>
LogFile	<p>Name of the log file that contains actions performed by this function. Default: 'ddlookup.log'</p>

Output parameters

The following output parameters are available:

Parameter	Description
msgStruct	Structure with messages.

tl_get_mlfcnobjects

tl_get_mlfcnobjects

Purpose Compiles a list of MATLAB code model elements of a specified class.

Syntax

```
[hMLFcnObjList, modelElementTypes] = tl_get_mlfcnobjects(systemList, objClass)
```

Input parameters The following input parameters are available:

Parameter	Description
systemList	Simulink systems to be searched for MATLAB code model elements.
objClass	Either a list of model element types or the string 'all' as a shortcut for all MATLAB code model element types known by TargetLink. Default: 'all'

Output parameters The following output parameters are available:

Parameter	Description
hMLFcnObjList	Column vector with handles of the found MATLAB code model elements or an empty matrix if no objects were found.
modelElementTypes	Cell array containing the model element types.

Example

```
% get all MATLAB code model elements located in the block diagrams model1 and model2.
h = tl_get_mlfcnobjects({'model1', 'model2'});

% get inputs and outputs of MATLAB function blocks in myModel.
hMLFcnBlocks = tl_get_blocks('myModel', 'Stateflow')
[h, meTypes] = tl_get_mlfcnobjects(hMLFcnBlocks, {'MLFcnInput', 'MLFcnOutput'})
```

A

API

tl_get_mlfcnobjects 18

tlCreateMATLABFunctionDDObjects 16

C

Common Program Data folder 6

CommonProgramDataFolder 6

D

Documents folder 6

DocumentsFolder 6

L

Local Program Data folder 6

LocalProgramDataFolder 6

T

tl_get_mlfcnobjects 18

tlCreateMATLABFunctionDDObjects 16

