



**Librairies  
Manual**

## **SCADE Suite Libraries Manual**



## CONTACTS

### Legal Contact

Ansys France  
15 place Georges Pompidou  
78180 Montigny-le-Bretonneux FRANCE  
Phone: +33 1 30 60 15 00  
Fax: +33 1 30 64 19 42

### Technical Support

Ansys France  
Parc Avenue, 9 rue Michel Labrousse  
31100 Toulouse FRANCE  
Phone: +33 5 34 60 90 50  
Fax: +33 5 34 60 90 41

Submit questions to Ansys SCADE Products Technical Support at [support@ansys.com](mailto:support@ansys.com).

Contact one of our Sales representatives at [scade-sales@ansys.com](mailto:scade-sales@ansys.com).

Direct general questions about SCADE products to [scade-info@ansys.com](mailto:scade-info@ansys.com).

Discover latest news on our products at [www.ansys.com/products/embedded-software](http://www.ansys.com/products/embedded-software).

## LEGAL INFORMATION

Copyrights © 2024 ANSYS, Inc. All rights reserved. Ansys, SCADE, SCADE Suite, SCADE Display, SCADE Architect, SCADE LifeCycle, SCADE Test, and Twin Builder are trademarks or registered trademarks of ANSYS, Inc. or its subsidiaries in the U.S. or other countries. SCADE Suite Libraries Manual – Published October 2024.

All other trademarks and tradenames contained herein are the property of their respective owners.

## TERMS OF USE

**Important!** Read carefully before starting this software and referring to its user documentation. This publication, as well as the software it describes, is distributed as part of the SCADE user documentation under license and may be used or copied only in accordance with the terms and conditions of the Software License Agreement (SLA) accepted during SCADE product installation. Except as permitted by the SLA, the content of this publication cannot be reproduced, stored, or transmitted in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior permission of Ansys. Any existing artwork or images that you may want to reuse in your projects may be protected under copyright law. The unauthorized use of such material into your own work may constitute a violation of Ansys copyrights. If needed, make sure you obtain the written permission from Ansys. By starting the software you expressly agree to the following conditions:

- **Property:** This software is and remains the exclusive property of Ansys. It cannot be copied or distributed without written authorization from Ansys. Ansys retains title and ownership of the software.
- **Warranty:** The software is provided, as is, without warranty of any kind. Ansys does not guarantee the use, or the results from the use, of this software. All risk associated with usage results and performance is assumed by you the user.

## DOCUMENTATION DISCLAIMER

The content of SCADE products user documentation is distributed for informational use only, is subject to change without notice, and should not be construed as a commitment by Ansys. Although every precaution has been taken to prepare this manual, Ansys assumes no responsibility or liability for any errors that may be contained in this book or any damages resulting from the use of the information contained herein.

## THIRD-PARTY LEGAL INFO

The Legal Notice (PDF) about third-party software can be found under the **help** folder of the SCADE installation.

Shipping date: 30/10/24

Revision: SC-LM-25 - DOC/rev/60204-10



# Libraries Manual Overview

This manual provides basic descriptions of all libraries delivered with your SCADE Suite installation. The content covers all libraries compatible with SCADE Suite projects.

- [“Fundamentals about Libraries”](#)
- Chapter 2: [“Library libdigital”](#)
- Chapter 3: [“Library libimpl”](#)
- Chapter 4: [“Library liblinear”](#)
- Chapter 5: [“Library libmath”](#)
- Chapter 6: [“Library libmathext”](#)
- Chapter 7: [“Library libpwnlinear”](#)
- Chapter 8: [“Library libsmk”](#)
- Chapter 9: [“Library libverif”](#)

[“Index”](#)

## RELATED DOCUMENTS

- *SCADE Suite User Manual*

## TYPOGRAPHICAL CONVENTIONS

<b>Bold</b>	GUI buttons, options, tabs, radio buttons, drop-down lists, window titles, panels.
<i>Italics</i>	Path for menu selection from main menu bar or in contextual menus, new technical concepts or words, references to product manuals.
Courier New	Code extracts, constant or variable names, file names, directory path, or typed text.
Title Case	Names of application components, menus.
UPPERCASE	File formats



# Table of Contents

---

## Libraries Manual Overview

<b>1. Fundamentals about Libraries</b>	<b>1 - 1</b>
Libraries and SCADE Suite Design Activities	1 - 2
Libraries and Model Implementation	1 - 3
Libraries and Simulink Modeling Activities	1 - 4
Libraries and Formal Verification Activities	1 - 5
<b>2. Library libdigital</b>	<b>2 - 7</b>
Digital Operators	2 - 8
Boolean Vector to Int8	2 - 8
Boolean Vector to Int16	2 - 9
Boolean Vector to Int32	2 - 9
Boolean Vector to Int64	2 - 10
Boolean Vector to UInt8	2 - 10
Boolean Vector to UInt16	2 - 11
Boolean Vector to UInt32	2 - 11
Boolean Vector to UInt64	2 - 12
Count-Down	2 - 12
Either Edge	2 - 13
EntryDetect	2 - 13
Falling Edge	2 - 14
Falling Edge (Re-Trigger)	2 - 14
Falling Edge (without Re-Trigger)	2 - 15
Flip-Flop JK	2 - 16
Flip-Flop Priority to Reset	2 - 17
Flip-Flop Priority to Set	2 - 18
Inactive Cycles	2 - 19
Inactive Time	2 - 19
Integer to Boolean Vector	2 - 20

# Table of Contents

---

Relay	2 - 20
Rising Edge	2 - 21
Rising Edge (Re-Trigger)	2 - 22
Rising Edge (without Re-Trigger)	2 - 22
Toggle	2 - 23
Trigger (Either)	2 - 23
Trigger (Fall)	2 - 24
Trigger (Rise)	2 - 24
<b>Truth Table Operators</b>	2 - 25
Truth Table	2 - 25
Truth Table (Exhaustive)	2 - 26
Truth Table (Index)	2 - 26
<b>3. Library libimpl</b>	3 - 27
<b>Arithmetic Operators</b>	3 - 29
Integer Division (Ceiling Rounding)	3 - 30
Integer Division (Fix Rounding)	3 - 30
Integer Division (Floor Rounding)	3 - 31
Integer Division (Saturation)	3 - 31
Integer Multiplication	3 - 32
Integer Multiplication (Int16)	3 - 32
Integer Multiplication (UInt16)	3 - 33
Integer Multiplication (Int32)	3 - 33
Integer Multiplication (UInt32)	3 - 33
Integer Multiplication (Saturation)	3 - 34
Integer Negation (Saturation)	3 - 34
Integer Subtraction (Saturation)	3 - 35
Integer Sum (Saturation)	3 - 35
<b>Comparison Operators</b>	3 - 36
Strictly Less Than	3 - 36



# Table of Contents

---

Less Than or Equal	3 - 37
Equal	3 - 37
Different	3 - 38
Strictly Greater Than	3 - 38
Greater Than or Equal	3 - 39
<b>Bitshift and Bitwise Operators</b>	3 - 40
Bit Clear	3 - 40
Bit Set	3 - 41
Decrease LSB	3 - 41
Extract Bit Range	3 - 42
Increase LSB to Ceiling	3 - 42
Increase LSB to Fix	3 - 43
Increase LSB to Floor	3 - 43
Shift Left	3 - 44
Shift Right	3 - 44
<b>4. Library liblinear</b>	4 - 45
<b>Filter and Transfer Function Operators</b>	4 - 46
Discrete Filter (Numerator scalar, Denominator deg. 1)	4 - 47
Discrete Filter (Numerator scalar, Denominator deg. 1 normalized)	4 - 47
Discrete Filter (Numerator scalar, Denominator deg. 2)	4 - 48
Discrete Filter (Numerator scalar, Denominator deg. 2 normalized)	4 - 48
Discrete Filter (Numerator scalar, Denominator deg. Ds)	4 - 49
Discrete Filter (Numerator scalar, Denominator deg. Ds normalized)	4 - 50
Discrete Filter (Numerator deg. 1, Denominator deg. 1)	4 - 51
Discrete Filter (Numerator deg. 1, Denominator deg. 1 normalized)	4 - 51
Discrete Filter (Numerator deg. 1, Denominator deg. 2)	4 - 52
Discrete Filter (Numerator deg. 1, Denominator deg. 2 normalized)	4 - 52
Discrete Filter (Numerator deg. Ns, Denominator deg. Ds)	4 - 53
Discrete Filter (Numerator deg. Ns, Denominator deg. Ds normalized)	4 - 54

# Table of Contents

---

Discrete Filter (Numerator deg. $N_s$ , Denominator deg. $N_s$ )	4 - 55
Discrete Filter (Numerator deg. $N_s$ , Denominator deg. $N_s$ normalized)	4 - 56
Transfer Function (Numerator scalar, Denominator deg. 1)	4 - 56
Transfer Function (Numerator scalar, Denominator deg. 2)	4 - 57
Transfer Function (Numerator scalar, Denominator deg. $D_s$ )	4 - 57
Transfer Function (Numerator deg. 1, Denominator deg. 2)	4 - 58
Transfer Function (Numerator deg. $N_s$ , Denominator deg. $D_s$ )	4 - 58
<b>Linear Function Operators</b>	4 - 59
Backlash	4 - 59
Derivative	4 - 60
Detect (Change)	4 - 60
Detect (Decrease)	4 - 61
Detect (Fall Negative)	4 - 61
Detect (Fall Non-Positive)	4 - 62
Detect (Increase)	4 - 62
Detect (Rise Non-Negative)	4 - 63
Detect (Rise Positive)	4 - 63
Filter (First Order Loop)	4 - 64
Gain	4 - 64
Hit Crossing (Either Direction)	4 - 65
Hit Crossing (Falling Direction)	4 - 65
Hit Crossing (Rising Direction)	4 - 66
Integrator (Backward)	4 - 66
Integrator (Forward)	4 - 67
Integrator (Trapezoidal)	4 - 68
Mean Cycle (2 values)	4 - 69
Mean Cycle (3 values)	4 - 69
Memory	4 - 70
Memory (Basic)	4 - 70

# Table of Contents

---

<b>5. Library libmath</b>	5 - 71
<b>Arithmetic Operators</b>	5 - 73
Absolute Value	5 - 73
Maximum (2 inputs)	5 - 74
Maximum (3 inputs)	5 - 74
Mean (2 inputs)	5 - 75
Mean (3 inputs)	5 - 75
Minimum (2 inputs)	5 - 76
Minimum (3 inputs)	5 - 76
Mod	5 - 77
Polynomial	5 - 77
Sign	5 - 78
<b>Conversion Operators</b>	5 - 79
Boolean to Float32	5 - 79
Boolean to Float64	5 - 80
Boolean to Int8	5 - 80
Boolean to Int16	5 - 81
Boolean to Int32	5 - 81
Boolean to Int64	5 - 82
Boolean to UInt8	5 - 82
Boolean to UInt16	5 - 83
Boolean to UInt32	5 - 83
Boolean to UInt64	5 - 84
Integer to Boolean	5 - 84
Octet to UInt8	5 - 85
Real to Boolean	5 - 85
Rounding Function	5 - 86
Rounding to Ceiling Function	5 - 86
Rounding to Floor Function	5 - 87

# Table of Contents

---

<b>Interval Operators</b>	5 - 88
In Range (In-In)	5 - 88
In Range (In-Out)	5 - 89
In Range (Out-In)	5 - 89
In Range (Out-Out)	5 - 90
<b>Vector and Matrix Operators</b>	5 - 91
BuildArray	5 - 91
Determinant of Square Matrix (2x2)	5 - 92
Determinant of Square Matrix (3x3)	5 - 92
Determinant of Square Matrix (4x4)	5 - 93
Inverse of Square Matrix (2x2)	5 - 93
Inverse of Square Matrix (3x3)	5 - 94
Inverse of Square Matrix (4x4)	5 - 94
Matrix Addition	5 - 95
Matrix Difference	5 - 95
Matrix Product	5 - 96
Matrix Product by Vector	5 - 96
Scalar Product	5 - 97
Selector	5 - 97
Vector Addition	5 - 98
Vector Difference	5 - 98
Vector Product by Matrix	5 - 99
 <b>6. Library libmathtext</b>	 6 - 101
<b>Trigonometric and Power-Based Functions</b>	6 - 102
<b>Advanced Mathematical Operators</b>	6 - 103
Integer Power for Integers	6 - 103
Integer Power for Reals	6 - 104
Inverse	6 - 104
Real Power for Reals	6 - 105

# Table of Contents

---

Square	6 - 105
<b>Conversion Operators</b>	6 - 106
Cartesian to Polar	6 - 106
Cartesian to Spherical	6 - 107
Celsius to Fahrenheit	6 - 107
Degrees to Radians	6 - 108
Fahrenheit to Celsius	6 - 108
Polar to Cartesian	6 - 109
Radians to Degrees	6 - 109
Spherical to Cartesian	6 - 110
<b>7. Library libpwnlinear</b>	7 - 111
<b>Piecewise Linear Functions</b>	7 - 112
CheckSlope	7 - 112
Clock Counter	7 - 113
Counter	7 - 113
Dead Band (Asymmetrical)	7 - 114
Dead Band (Symmetrical)	7 - 114
Hysteresis (Falling)	7 - 115
Hysteresis (Rising)	7 - 115
Limiter (Asymmetrical)	7 - 116
Limiter (Symmetrical)	7 - 116
MaxReset	7 - 117
MinReset	7 - 117
Pre-load (Asymmetrical)	7 - 118
Pre-load (Symmetrical)	7 - 118
Quantizer	7 - 119
Rate Limiter	7 - 119
<b>Look-Up Table Operators</b>	7 - 120
Interpolation 1D	7 - 120

# Table of Contents

---

Interpolation 1D (Floor)	7 - 121
Interpolation 2D	7 - 121
Interpolation 2D (Floor)	7 - 122
LUT 1D	7 - 122
LUT 1D (Value above)	7 - 123
LUT 1D (Value below)	7 - 124
LUT 1D (Nearest value)	7 - 125
LUT 2D	7 - 126
LUT 2D (Value above)	7 - 127
LUT 2D (Value below)	7 - 128
LUT 2D (Nearest value)	7 - 129
LUT 3D	7 - 130
LUT 3D (Nearest value)	7 - 131
Pre-LUT	7 - 132
Pre-LUT (Direct)	7 - 132
<b>8. Library libsmk</b>	<b>8 - 133</b>
<b>Discrete Filter Operators</b>	<b>8 - 135</b>
Discrete Filter	8 - 135
Discrete Filter (Normalized)	8 - 135
<b>Discrete-Time Integrator Operators</b>	<b>8 - 136</b>
Discrete-Time Integrator (Backward)	8 - 136
Discrete-Time Integrator (Forward)	8 - 137
Discrete-Time Integrator (Trapezoidal)	8 - 138
Transfer Function (First Order)	8 - 139
Transfer Function (Lead or Lag)	8 - 139
Transfer Function (Real Zero)	8 - 140
<b>Arithmetic Operators</b>	<b>8 - 141</b>
Bias	8 - 141
Inverse (int/int32)	8 - 142

# Table of Contents

---

Maximum (Resettable)	8 - 142
Minimum (Resettable)	8 - 143
Modulo (int/int32)	8 - 143
Modulo (real/float)	8 - 144
Polynomial	8 - 144
Remainder (real/float)	8 - 145
Sign (int/int32)	8 - 145
<b>Time Operators</b>	8 - 146
Decrement Time to Zero (int/int32)	8 - 146
Decrement Time to Zero (real/float)	8 - 147
Decrement to Zero	8 - 147
Sample Time (all math operations)	8 - 148
Unit Delay Enabled	8 - 149
Unit Delay Enabled (Init)	8 - 149
Unit Delay Enabled Resettable	8 - 150
Unit Delay Enabled Resettable (Init)	8 - 150
Unit Delay (Init)	8 - 151
Unit Delay Resettable	8 - 151
Unit Delay Resettable (Init)	8 - 152
<b>Bitwise Logical Operators</b>	8 - 153
Bit Clear	8 - 153
Bit Set	8 - 154
Bitwise AND	8 - 154
Bitwise Nand	8 - 155
Bitwise Nor	8 - 155
Bitwise NOT	8 - 156
Bitwise OR	8 - 156
Bitwise Exclusive OR	8 - 157
Bitwise Shift	8 - 157
Bitwise Shift Left	8 - 158

# Table of Contents

---

Bitwise Shift Right	8 - 158
Extract Bit Range	8 - 159
<b>Conversion and Transformation Operators</b>	8 - 160
Boolean to Boolean	8 - 160
Cartesian to Polar	8 - 161
Cartesian to Spherical	8 - 161
Celsius to Fahrenheit	8 - 162
Degrees to Radians	8 - 162
Enumeration to Integer	8 - 163
Fahrenheit to Celsius	8 - 163
Integer to Boolean	8 - 164
Integer to Enumeration	8 - 164
Polar to Cartesian	8 - 165
Radians to Degrees	8 - 165
Real to Boolean	8 - 166
Scalar to Boolean	8 - 166
Spherical to Cartesian	8 - 167
<b>Flip-Flop Operators</b>	8 - 168
Clock	8 - 168
DLatch	8 - 168
Flip-Flop D	8 - 169
Flip-Flop JK	8 - 169
Flip-Flop SR	8 - 170
<b>Logical Operators</b>	8 - 171
AND	8 - 171
Compare (all operations)	8 - 172
Compare Enumeration	8 - 173
Detect Change	8 - 174
Detect Decrease	8 - 174
Detect Falling Edge (negative)	8 - 175



# Table of Contents

---

Detect Falling Edge (non-positive)	8 - 175
Detect Increase	8 - 176
Detect Rising Edge (non-negative)	8 - 176
Detect Rising Edge (positive)	8 - 177
In Range (In-In)	8 - 177
In Range (In-Out)	8 - 178
In Range (Out-In)	8 - 178
In Range (Out-Out)	8 - 179
NOT	8 - 179
OR	8 - 180
<b>Model Verification Operators</b>	8 - 181
Assertion	8 - 181
Assertion (Inverse)	8 - 182
Check Bounds	8 - 182
Check Bounds (Extended)	8 - 183
Check Gap	8 - 183
Check Gap (Extended)	8 - 184
Check Gradient	8 - 184
Check Gradient (Extended)	8 - 185
Check Lower Bound	8 - 185
Check Lower Bound (Extended)	8 - 186
Check Upper Bound	8 - 186
Check Upper Bound (Extended)	8 - 187
<b>Signal Routing Operators</b>	8 - 188
If Then Else	8 - 188
Ignore First Input	8 - 189
Merge (2 inputs)	8 - 189
Merge (3 inputs)	8 - 190
Switch	8 - 190
Switch (GTT)	8 - 191

# Table of Contents

---

Switch (NEZ)	8 - 191
Switch Enumeration	8 - 192
Switch Enumeration (GTT)	8 - 192
<b>Subsystem Operators</b>	8 - 193
Trigger Either Edge	8 - 193
Trigger Either Edge (Extended)	8 - 194
Trigger Falling Edge	8 - 194
Trigger Falling Edge (Extended)	8 - 195
Trigger Rising Edge	8 - 195
Trigger Rising Edge (Extended)	8 - 196
<b>Signal Generator Operators</b>	8 - 197
Clock Generator	8 - 197
Ground (Boolean)	8 - 197
Ground (int/int32)	8 - 198
Ground (real/float)	8 - 198
<b>Discontinuity Operators</b>	8 - 199
Backlash	8 - 199
Dead Zone Dynamic	8 - 200
Rate Limiter	8 - 200
Rate Limiter Dynamic	8 - 201
Relay	8 - 201
Saturate Dynamic	8 - 202
Wrap To Zero	8 - 202
<b>LUT, Pre-LUT and Interpolation Operators</b>	8 - 203
LUT 1D	8 - 203
Pre-LUT2	8 - 204
Pre-LUT2 (Direct)	8 - 204
Interpolation1D2	8 - 205
Interpolation1D (Floor)	8 - 205

# Table of Contents

---

Interpolation2D2	8 - 206
Interpolation2D (Floor)	8 - 206
<b>Miscellaneous Operators</b>	8 - 207
Border Crossing Detection	8 - 207
Counter (limited)	8 - 207
Entry Detection	8 - 208
Identity	8 - 208
Inactive Cycles	8 - 209
Inactive Time	8 - 209
Quantizer	8 - 210
Unit Delay (Helper)	8 - 210
Width (Matrix)	8 - 211
Width (Scalar)	8 - 211
Width (Vector)	8 - 212
<b>9. Library libverif</b>	9 - 213
After Nth Tick	9 - 213
Always After First Condition	9 - 214
At Least N Ticks	9 - 215
Has Never Been True	9 - 216
Implies	9 - 217
Implies Within N Ticks	9 - 218
<b>INDEX</b>	219



# 1 / Fundamentals about Libraries

This manual provides an exhaustive description of each SCADE Suite library with detailed operator description and important information. The most widely used functions or filters necessary when modeling applications are provided in SCADE Suite libraries.

- [“Libraries and SCADE Suite Design Activities”](#)
- [“Libraries and Model Implementation”](#)
- [“Libraries and Simulink Modeling Activities”](#)
- [“Libraries and Formal Verification Activities”](#)

The content of this manual is divided into chapters describing each library:

- [libdigital](#)
- [liblinear](#)
- [libmathext](#)
- [libsmk](#)
- [libimpl](#)
- [libmath](#)
- [libpwlinear](#)
- [libverif](#)

For each library operator, any differences related to the library version in use in SCADE Suite projects are reported. For basic instructions or tips about SCADE Suite library management, please consult [“Managing Libraries”](#) on page 115 in *SCADE Suite User Manual*.

# Libraries and SCADE Suite Design Activities

The libraries listed below are intended to facilitate design work. All the libraries distributed with SCADE Suite provide a set of operators which are commonly used in model design.

## WHAT IS THE GENERIC FUNCTION OF OPERATORS IN EACH LIBRARY?

- **libdigital** contains digital operators that handle Boolean data (*e.g.*, falling and rising edges, flip-flops, truth tables). Read details in [“Library libdigital”](#).
- **liblinear** contains operators that compute linear functions or filters like derivative, gain, memories, or integrators. Read details in [“Library liblinear”](#).
- **libmath** contains basic mathematical operators (*e.g.*, absolute value, rounding, mean, maximum) and operators for vectors and matrices. Read details in [“Library libmath”](#).
- **libmathext** contains operators that compute trigonometric functions and power-based functions. Read details in [“Library libmathext”](#).
- **libpwlinear** contains operators that compute piecewise linear functions or filters (clock counter, counter, dead bands, hysteresis, rate limiter, etc.) and look-up tables. Read details in [“Library libpwlinear”](#).

## WHAT IS THE GENERAL PURPOSE OF SCADE SUITE LIBRARIES?

SCADE Suite includes an extensive set of libraries with functions commonly used in software development. All SCADE Suite libraries can be modified and enriched. Users can also create, share, and reuse their own libraries between projects.

---

### Note

Library functions are expressed in Scade language when it makes sense. Some are in C code, meaning they may be platform-dependent as libimpl and libmathext.

---

## WHAT CAN I USE THEM FOR?

Most libraries contain mathematical functions and are highly generic. They can be useful in any application developed with SCADE Suite. Other libraries support more specific activities in SCADE Suite model-based design environment. For instance, `libverif` is needed to perform formal verification on SCADE Suite models.

## WHERE CAN I FIND THEM IN MY DISTRIBUTION?

The set of libraries distributed with SCADE Suite is located in the following directory: `<SCADESuiteInstallDirectory>\libraries`

## DISCLAIMER

Although every precaution has been taken when developing and testing the libraries, Ansys assumes no responsibility or liability for any errors that may be contained in the models or any damages resulting from the use of the libraries. If the libraries are used in a qualified / certified context, all documentation, verification, and validation activities requested by the Safety standard must be performed by the end user.

# Libraries and Model Implementation

SCADE Suite models are independent from the target. Indeed, the implementation choice for Scade data types is made only when code is being built. If data types are different and known at model level, the implementation size of integers is needed; the `libimpl` library provides many possible implementations for integer types and functions.

## WHAT IS THE GENERIC FUNCTION OF OPERATORS IN THIS LIBRARY?

- **libimpl** contains all standard integer operators (arithmetic, comparison, logical (bitwise, left/right shifting), time operators, type conversion, etc.) and types (unsigned and signed 8, 16, and 32 bits integers). For details, see [“Library libimpl”](#).
- 

### Note

This library can easily be customized for specific target: The types are defined in the following header files: `libimpl.h` and `macro_libimpl.h`. The definitions of macro functions are contained in the `macro_libimpl.h` file. For SCADE Suite Simulator, conversion functions are defined in the `libimpl.c` file.

---

## Libraries and Simulink Modeling Activities

When importing Simulink models into SCADE Suite, Simulink blocks are mapped, depending on their purpose, to Scade nodes using `libsmk`, `libmath`, `libdigital`, `liblinear`, `libpwlinear` or `libmathext` libraries.

The *libsmk* library is dedicated to support the correct translation of Simulink models into SCADE Suite models.

---

### Note

The `libsmk` library is available only if SCADE Suite Simulink® Importer is installed.

---

## WHAT IS THE GENERIC FUNCTION OF OPERATORS IN THIS LIBRARY?

- **libsmk** contains operators designed to reproduce the behavior of Simulink blocks with respect to SCADE Suite formalism. For details, see [“Library libsmk”](#).



# Libraries and Formal Verification Activities

Formal verification allows designers to check the correctness of their design. In SCADE Suite, Design Verifier enables extended debugging, non-regression proof and property verification without having to write verification tests.

The *libverif* library is dedicated to the formal verification of SCADE Suite models. It provides you with a set of library operators that are useful for expressing the properties to be verified in your models.

---

## Note

The libverif library is available only if Design Verifier is installed.

---

## WHAT IS THE GENERIC FUNCTION OF OPERATORS IN THIS LIBRARY?

- **libverif** contains verification operators designed to help you express properties (implies, etc.). Such operators are expressed as regular Scade nodes. For details, see ["Library libverif"](#).



# 2 / Library libdigital

This library contains digital operators that handle Boolean data and a count-down counter. You can read the technical description of the library components by package:

- ["Digital Operators"](#) in **digital** package
- ["Truth Table Operators"](#) in **truthtables** package

Access the description of library operators in alphabetical order:

<a href="#">"Boolean Vector to Int8"</a>	<a href="#">"Flip-Flop Priority to Set"</a>
<a href="#">"Boolean Vector to Int16"</a>	<a href="#">"Inactive Cycles"</a>
<a href="#">"Boolean Vector to Int32"</a>	<a href="#">"Inactive Time"</a>
<a href="#">"Boolean Vector to Int64"</a>	<a href="#">"Integer to Boolean Vector"</a>
<a href="#">"Boolean Vector to UInt8"</a>	<a href="#">"Relay"</a>
<a href="#">"Boolean Vector to UInt16"</a>	<a href="#">"Rising Edge"</a>
<a href="#">"Boolean Vector to UInt32"</a>	<a href="#">"Rising Edge (Re-Trigger)"</a>
<a href="#">"Boolean Vector to UInt64"</a>	<a href="#">"Rising Edge (without Re-Trigger)"</a>
<a href="#">"Count-Down"</a>	<a href="#">"Toggle"</a>
<a href="#">"Either Edge"</a>	<a href="#">"Trigger (Either)"</a>
<a href="#">"EntryDetect"</a>	<a href="#">"Trigger (Fall)"</a>
<a href="#">"Falling Edge"</a>	<a href="#">"Trigger (Rise)"</a>
<a href="#">"Falling Edge (Re-Trigger)"</a>	<a href="#">"Truth Table"</a>
<a href="#">"Falling Edge (without Re-Trigger)"</a>	<a href="#">"Truth Table (Exhaustive)"</a>
<a href="#">"Flip-Flop JK"</a>	<a href="#">"Truth Table (Index)"</a>
<a href="#">"Flip-Flop Priority to Reset"</a>	

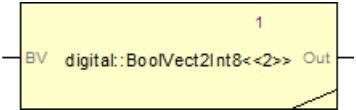
# Digital Operators

Read the description of the following operators:

- ["Boolean Vector to Int8"](#)  
["Boolean Vector to Int16"](#)  
["Boolean Vector to Int32"](#)  
["Boolean Vector to Int64"](#)  
["Boolean Vector to UInt8"](#)  
["Boolean Vector to UInt16"](#)  
["Boolean Vector to UInt32"](#)  
["Boolean Vector to UInt64"](#)  
["Count-Down"](#)  
["Either Edge"](#)  
["EntryDetect"](#)  
["Falling Edge"](#)  
["Falling Edge \(Re-Trigger\)"](#)  
["Falling Edge \(without Re-Trigger\)"](#)
- ["Flip-Flop JK"](#)  
["Flip-Flop Priority to Reset"](#)  
["Flip-Flop Priority to Set"](#)  
["Inactive Cycles"](#)  
["Inactive Time"](#)  
["Integer to Boolean Vector"](#)  
["Relay"](#)  
["Rising Edge"](#)  
["Rising Edge \(Re-Trigger\)"](#)  
["Rising Edge \(without Re-Trigger\)"](#)  
["Toggle"](#)  
["Trigger \(Either\)"](#)  
["Trigger \(Fall\)"](#)  
["Trigger \(Rise\)"](#)

## Boolean Vector to Int8

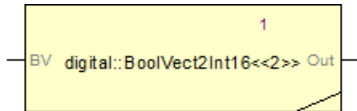
Symbol 2.1:



SCADE name	BoolVect2Int8
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: int8
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Int16

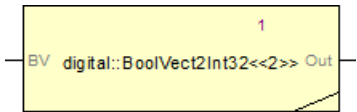
Symbol 2.2:



SCADE name	BoolVect2Int16
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: int16
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Int32

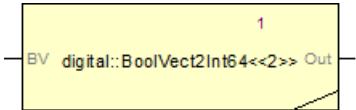
Symbol 2.3:



SCADE name	BoolVect2Int32
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: int32
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Int64

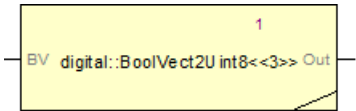
Symbol 2.4:



SCADE name	BoolVect2Int64
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: int64
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Uint8

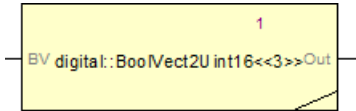
Symbol 2.5:



SCADE name	BoolVect2Uint8
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: uint8
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Uint16

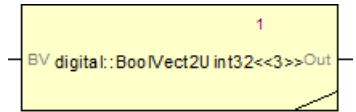
Symbol 2.6:



SCADE name	BoolVect2Uint16
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: uint16
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Uint32

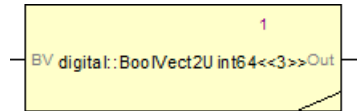
Symbol 2.7:



SCADE name	BoolVect2Uint32
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: uint32
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Boolean Vector to Uint64

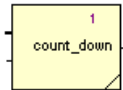
Symbol 2.8:



SCADE name	BoolVect2Uint64
Package	digital
Function	Outputs the integer value computed from a vector of booleans BV interpreted as its Boolean representation.
Size parameters	Nbool
Inputs	BV: bool ^Nbool
Outputs	Out: uint64
Comment	BV[0] is the highest bit, <i>i.e.</i> , (BoolVect2Int([true,false])=2.

# Count-Down

Symbol 2.9:

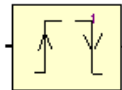


SCADE name	count_down
Package	digital
Function	Returns input N when Reset is true, otherwise, decrements the output by one at each cycle.
Inputs	Reset: bool and N: 'T
Outputs	cpt: 'T where 'T integer
Comment	The count_down function does not "stop" at value 0; it decrements the count at each step, until Reset is true, even if the count is negative.



# Either Edge

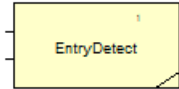
Symbol 2.10:



SCADE name	EitherEdge
Package	digital
Function	Detects all edges, rising or falling (i.e., true to false, and false to true transitions).
Inputs	EE_Input: bool
Outputs	EE_Output: bool
Comment	Output is true during transition clock cycle. Output is initialized to false.

# EntryDetect

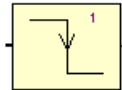
Symbol 2.11:



SCADE name	EntryDetect
Package	digital
Function	Detects entry into a range based on two Boolean inputs: one input indicates whether the signal is outside the border and the other one whether the signal is at the border.
Inputs	Outside: bool and OnTheBorder: bool
Outputs	Entry: bool

# Falling Edge

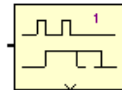
Symbol 2.12:



SCADE name	FallingEdge
Package	digital
Function	Detects a falling edge (i.e., true to false transition of Boolean input).
Inputs	FE_Input: bool
Outputs	FE_Output: bool
Comment	Output is true during transition clock cycle. Output is initialized to false.

# Falling Edge (Re-Trigger)

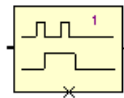
Symbol 2.13:



SCADE name	FallingEdgeRetrigger
Package	digital
Function	Detects a falling edge (i.e., true to false transition of Boolean input).
Inputs	FER_Input: bool
Hidden inputs	NumberOfCycle: int
Outputs	FER_Output: bool
Comment	The output becomes true as soon as the transition is detected and remains unchanged during NumberOfCycle. Output is initialized to false. If another falling edge occurs while the output is true, it is detected.

# Falling Edge (without Re-Trigger)

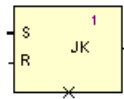
Symbol 2.14:



SCADE name	FallingEdgeNoRetrigger
Package	digital
Function	Detects a falling edge (i.e., <code>true</code> to <code>false</code> transition of Boolean input).
Inputs	FENR_Input: bool
Hidden inputs	NumberOfCycle: 'T
Outputs	FENR_Output: bool
Comment	The output becomes <code>true</code> as soon as the transition is detected and remains unchanged during <code>NumberOfCycle</code> . Output is initialized to <code>false</code> . If another falling edge occurs while the output is <code>true</code> , it is not detected.

# Flip-Flop JK

Symbol 2.15:



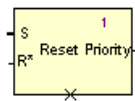
SCADE name	FlipFlopJK
Package	digital
Function	Basic JK flip-flop of Boolean values. See <a href="#">Table 2.1</a> below.
Inputs	Set and Reset: bool
Hidden inputs	Init: bool
Outputs	FFJK_Output: bool
Comment	At the first step, the Output takes the same value as Init, regardless of the values of Set and Reset.

Table 2.1: FlipFlopJK functional description

Set	Reset	FFJK_Output
false	false	FFJK_Output -1 (memorization)
false	true	false
true	false	true
true	true	not FFJK_Output -1 (memorization)

# Flip-Flop Priority to Reset

Symbol 2.16:



SCADE name	FlipFlopReset
Package	digital
Function	Basic flip-flop of Boolean values with priority for reset. See <a href="#">Table 2.2</a> below.
Inputs	Set and Reset: bool
Hidden inputs	Init: bool
Outputs	FFR_Output: bool
KCG pragma	Operator expansion
Comment	At the first step, the Output takes the same value as Init, regardless of the values of Set and Reset.

Table 2.2: FlipFlopReset functional description

Set	Reset	FFR_Output
false	false	FFR_Output -1 (memorization)
false	true	false
true	false	true
true	true	false

# Flip-Flop Priority to Set

Symbol 2.17:



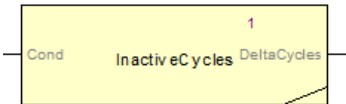
SCADE name	FlipFlopSet
Package	digital
Function	Basic flip-flop of Boolean values with priority for set. See <a href="#">Table 2.3</a> below.
Inputs	Set and Reset: bool
Hidden inputs	Init: bool
Outputs	FFS_Output: bool
KCG pragma	Operator expansion
Comment	At the first step, the Output takes the same value as Init, regardless of the values of Set and Reset.

Table 2.3: FlipFlopSet functional description

Set	Reset	FFS_Output
false	false	FFS_Output -1 (memorization)
false	true	false
true	false	true
true	true	true

# Inactive Cycles

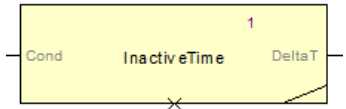
Symbol 2.18:



SCADE name	InactiveCycles
Package	digital
Function	Gives the number of cycles since last time Cond was true. Each time Cond is true, the counter is reset to 0.
Inputs	Cond: bool
Outputs	DeltaCycles: int32

# Inactive Time

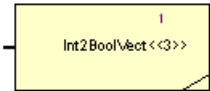
Symbol 2.19:



SCADE name	InactiveTime
Package	digital
Function	Gives the time since last time Cond was true. Each time Cond is true, the counter is reset to 0.0. The TimeCycle input gives the length of a cycle.
Inputs	Cond: bool
Hidden inputs	TimeCycle: 'T where 'T is numeric
Outputs	DeltaT: 'T where 'T is numeric

# Integer to Boolean Vector

Symbol 2.20:



SCADE name	Int2BoolVect
Package	digital
Function	Outputs a vector of booleans BV that is the boolean representation of the integer input, modulo $2^{N_{bool}}$ .
Size parameters	Nbool
Inputs	In : 'T where 'T is integer
Outputs	BV : bool ^Nbool
Comment	BV[0] is the highest bit, e.g., (Int2BoolVect<<2>>)([true,false])=2.

## Relay

Symbol 2.21:

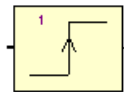


SCADE name	Relay
Package	digital
Function	Basic relay that generates discontinuous output values based on input.
Inputs	Input1 : 'T where 'T is numeric
Hidden inputs	UpperLimit_On: 'T and LowerLimit_Off: 'T where 'T is numeric OutputWhenOn: 'T and OutputWhenOff: 'T where 'T is numeric
Outputs	Output1 : 'T where 'T is numeric
Comment	Initially, the relay is off. UpperLimit_On >= LowerLimit_Off. <ul style="list-style-type: none"><li>when off (<i>i.e.</i>, output = OutputWhenOff), relay remains off unless the input reaches or exceeds the UpperLimit_On (<i>i.e.</i>, input &gt;= UpperLimit_On)</li><li>when on (<i>i.e.</i>, output = OutputWhenOn), relay remains on unless the input reaches or drops below LowerLimit_Off (<i>i.e.</i>, input &lt;= LowerLimit_Off).</li></ul>



# Rising Edge

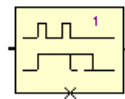
Symbol 2.22:



SCADE name	RisingEdge
Package	digital
Function	Detects a rising edge (i.e., <code>false</code> to <code>true</code> transition of Boolean input).
Inputs	RE_Input: bool
Outputs	RE_Output: bool
Comment	The output is true during the transition clock cycle. Output is initialized to <code>false</code> .

# Rising Edge (Re-Trigger)

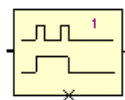
Symbol 2.23:



SCADE name	RisingEdgeRetrigger
Package	digital
Function	Detects a rising edge (i.e., <code>false</code> to <code>true</code> transition of Boolean input).
Inputs	RER_Input: bool
Hidden inputs	NumberOfCycle: 'T where 'T is integer
Outputs	RER_Output: bool
Comment	The output becomes <code>true</code> as soon as transition is detected and remains unchanged during NumberOfCycle. If another rising edge occurs while the output is <code>true</code> , it is detected. Output is initialized to <code>false</code> .

# Rising Edge (without Re-Trigger)

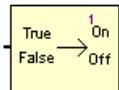
Symbol 2.24:



SCADE name	RisingEdgeNoRetrigger
Package	digital
Function	Detects a rising edge (i.e., <code>false</code> to <code>true</code> transition of Boolean input).
Inputs	RENr_Input: bool
Hidden inputs	NumberOfCycle: 'T where 'T is integer
Outputs	RENr_Output: bool
Comment	The output is set to <code>true</code> as soon as transition is detected and remains unchanged during NumberOfCycle. If another rising edge occurs while the output is <code>true</code> , it is not detected. Output is initialized to <code>false</code> .

# Toggle

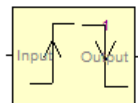
Symbol 2.25:



SCADE name	Toggle
Package	digital
Function	Stores the input value into On and Off states. When the input is <code>true</code> , On is <code>true</code> and Off is <code>false</code> . When the input is <code>false</code> , On is <code>false</code> and Off is <code>true</code> .
Inputs	T_Input: bool
Outputs	T_On and T_Off: bool
KCG pragma	Operator expansion
Comment	T_On always equals NOT T_Off.

# Trigger (Either)

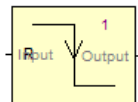
Symbol 2.26:



SCADE name	TriggerEither
Package	digital
Function	Output is true if any of the following conditions is true: <ul style="list-style-type: none"><li>• Input is 0 and was greater than 0 at previous cycle</li><li>• Input lower than 0 and was greater than 0 at previous cycle</li><li>• Input is lower than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle</li><li>• Input is 0 and was lower than 0 at previous cycle</li><li>• Input greater than 0 and was lower than 0 at previous cycle</li><li>• Input is greater than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle</li></ul>
Inputs	Input: 'T where 'T is numeric
Outputs	Output: bool

# Trigger (Fall)

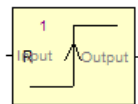
Symbol 2.27:



SCADE name	TriggerFall
Package	digital
Function	<div>Output is true if any of the following conditions is true:</div> <ul style="list-style-type: none"><li>• Input is 0 and was greater than 0 at previous cycle</li><li>• Input lower than 0 and was greater than 0 at previous cycle</li><li>• Input is lower than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle</li></ul>
Inputs	Input: 'T where 'T is numeric
Outputs	Output: bool

# Trigger (Rise)

Symbol 2.28:



SCADE name	TriggerRise
Package	digital
Function	<div>Output is true if any of the following conditions is true:</div> <ul style="list-style-type: none"><li>• Input is 0 and was lower than 0 at previous cycle</li><li>• Input greater than 0 and was lower than 0 at previous cycle</li><li>• Input is greater than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle</li></ul>
Inputs	Input: 'T where 'T is numeric
Outputs	Output: bool

# Truth Table Operators

Read the description of the following operators:

["Truth Table"](#)

["Truth Table \(Exhaustive\)"](#)

["Truth Table \(Index\)"](#)

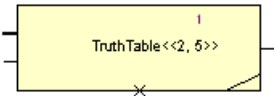
## TYPES

This package of library operators includes the following type definition:

```
ThruthTableValues = enum {T, F, X}
```

## Truth Table

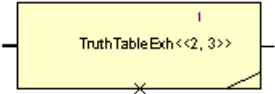
Symbol 2.29:



SCADE name	TruthTable
Package	truthtables
Function	Implements a truth table. Outputs the element from ResultValues that corresponds to the first line of Ttable that matches Conditions.
Size parameters	Ncond and Nlines
Inputs	Conditions: bool ^Ncond ResultValues: 'T'^(Nlines + 1) Ttable: TruthTableValues ^Ncond ^Nlines
Outputs	Result: 'T'
Comment	ResultValues[Nlines] is produced when no line matches. Element 'T' from Ttable matches true, 'F' matches false, 'X' matches both.

# Truth Table (Exhaustive)

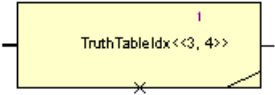
Symbol 2.30:



SCADE name	TruthTableExh
Package	truthtables
Function	Outputs the element fom ResultValues indexed exhaustively by the integer value computed from a vector of booleans BV interpreted as its boolean representation.
Size parameters	Nbool and Nres
Inputs	BV: bool ^Nbool ResultValues: 'T ^Nres
Outputs	Result: 'T
Comment	BV[0] is the highest bit. It is assumed that Nres = 2^Nbool. An "assume" in the operator checks that constraint.

# Truth Table (Index)

Symbol 2.31:



SCADE name	TruthTableIdx
Package	truthtables
Function	Implements a truth table. Outputs the index of the first line of Ttable of Nlines lines that matches Conditions.
Size parameters	Ncond and Nlines
Inputs	Conditions: bool ^Ncond Ttable: TruthTableValues ^Ncond ^Nlines
Outputs	LineIdx: int16
Comment	Index Nlines is produced when no line matches. Element 'T' from Ttable matches true, 'F' matches false, 'X' matches both.

# 3 / Library libimpl

This library contains types and operators that support sized integers.

The libimpl library contains the following packages:

- **impl**: to handle operators that support all sized integer types defined in libimpl. These are called polymorphic operators.
- **impls**: to handle operators that are the monomorphic specializations of the polymorphic operators from package impl.

As there is no additional cost in memory or execution time using polymorphic operators, it is simpler to use the operators from the **impl** package. In case, users need monomorphic operators from **impls**, it is possible.

The **impls\_arithmetic**, **impls\_comp**, and **impls\_bitwise** packages contain specializations for types int8, uint8, int16, uint16, int32, and uint32 of the polymorphic operators. These operators (more than 150) are named from the name of the polymorphic operator suffixed by 1 or 2 type identifier corresponding to the specialization.

---

## Design Verifier and Implementation

The current version of Design Verifier for SCADE 6 does not support operators from libimpl (considered as any imported operators).

---

- ["Arithmetic Operators"](#)
- ["Comparison Operators"](#)
- ["Bitshift and Bitwise Operators"](#)

Access the description of library operators in alphabetical order:

["Bit Clear"](#)

["Bit Set"](#)

["Decrease LSB"](#)

["Different"](#)

["Equal"](#)

["Extract Bit Range"](#)

["Greater Than or Equal"](#)

["Increase LSB to Ceiling"](#)

["Increase LSB to Fix"](#)

["Increase LSB to Floor"](#)

["Integer Division \(Ceiling Rounding\)"](#)

["Integer Division \(Fix Rounding\)"](#)

["Integer Division \(Floor Rounding\)"](#)

["Integer Division \(Saturation\)"](#)

["Integer Multiplication"](#)

["Integer Multiplication \(Int16\)"](#)

["Integer Multiplication \(UInt16\)"](#)

["Integer Multiplication \(Int32\)"](#)

["Integer Multiplication \(UInt32\)"](#)

["Integer Multiplication \(Saturation\)"](#)

["Integer Negation \(Saturation\)"](#)

["Integer Subtraction \(Saturation\)"](#)

["Integer Sum \(Saturation\)"](#)

["Less Than or Equal"](#)

["Strictly Greater Than"](#)

["Strictly Less Than"](#)

["Shift Left"](#)

["Shift Right"](#)

## OVERFLOW

Overflow means that the result of an operation is too large in absolute value to be represented in the type of the result. The result is wrapped with respect to the type, meaning that the result of the operation is computed modulo the largest possible value for the type plus 1, namely,  $\text{mod}(\text{max\_type} + 1)$ . For example, the subtraction operation defined as `iSub(<value=2>, <value=3>)` overflows if the type of the result is `uint8`. Saturation is one method for preventing overflows, but saturation operators are less efficient. Saturating an operation may also lead to unexpected results such as instability of the model, etc. Any libimpl operator that can overflow has a saturated version to allow you to choose between versions.



# Arithmetic Operators

The **libbimp1** library contains all basic mathematical operators such as summation, subtraction, multiplication, division, modulo or unary minus, that allow handling sized integer data.

---

## Note

For backward compatibility, all arithmetic operators defined in this library have as argument the following types: 'T1 and 'T2. Mixing signed and unsigned types is not supported. 'T1 and 'T2 must be of the same type.

For all operators, they can be int8, uint8, int16, uint16, int32, uint32.

---

Read the description of the following operators:

["Integer Division \(Ceiling Rounding\)"](#)

["Integer Division \(Fix Rounding\)"](#)

["Integer Division \(Floor Rounding\)"](#)

["Integer Division \(Saturation\)"](#)

["Integer Multiplication"](#)

["Integer Multiplication \(Int16\)"](#)

["Integer Multiplication \(UInt16\)"](#)

["Integer Multiplication \(Int32\)"](#)

["Integer Multiplication \(UInt32\)"](#)

["Integer Multiplication \(Saturation\)"](#)

["Integer Negation \(Saturation\)"](#)

["Integer Subtraction \(Saturation\)"](#)

["Integer Sum \(Saturation\)"](#)

# Integer Division (Ceiling Rounding)

Symbol 3.1:



SCADE name	iDivCeil
Function	Computes the division of integer inputs with ceiling rounding ( $Out1 = RoundCeil(In1/In2)$ ). Rounding always to upper integer value.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1
Caution	Overflow if second operand is 0.

# Integer Division (Fix Rounding)

Symbol 3.2:



SCADE name	iDivFix
Function	Computes the division of integer inputs with rounding toward zero ( $Out1 = (In1/In2)$ ).
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1
Caution	Overflow if second operand is 0.

# Integer Division (Floor Rounding)

Symbol 3.3:



SCADE name	iDivFloor
Function	Computes the division of integer inputs with floor rounding (Out1 = RoundFloor (In1/In2). Rounding always to lower integer value.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1
Caution	Overflow if second operand is 0.

# Integer Division (Saturation)

Symbol 3.4:



SCADE name	iDivSat
Function	Computes the division of integer inputs with saturation to avoid overflow. Implicit rounding is rounding toward zero (usual C integer division operation). (Out1 = (In1/In2) unless Out1 > max_int, then Out1 = max_int).
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1

# Integer Multiplication

Symbol 3.5:



SCADE name	iMul
Function	Computes the multiplication of two integer inputs.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T2
Caution	Can overflow

## Integer Multiplication (Int16)

Symbol 3.6:



SCADE name	iMul16
Function	Computes the multiplication of two int8 inputs and converts its output into int16.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 = int8
Outputs	Out1: imported int16

# Integer Multiplication (Uint16)

Symbol 3.7:



SCADE name	iMul16U
Function	Computes the multiplication of two uint8 inputs and converts its output into uint16.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 = uint8
Outputs	Out1: imported uint16

# Integer Multiplication (Int32)

Symbol 3.8:



SCADE name	iMul32
Function	Computes the multiplication of two int16 inputs and converts its output into int32.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 = int16
Outputs	Out1: imported int32

# Integer Multiplication (Uint32)

Symbol 3.9:



SCADE name	iMul32U
Function	Computes the multiplication of two uint16 inputs and converts its output into uint32.
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 = uint16
Outputs	Out1: imported uint32

# Integer Multiplication (Saturation)

Symbol 3.10:



SCADE name	iMulSat
Function	Computes the saturated multiplication of two integer inputs to avoid overflows (if Out1 > max_int, then Out1 = max_int; if Out1 < min_int, then Out1 = min_int).
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1
Caution	If the operation overflows relatively to the input type (size and signed/unsigned), the result is the max of the type. If it overflows below the lower limit, the result is the min of the type.

# Integer Negation (Saturation)

Symbol 3.11:



SCADE name	iNegSat
Function	Computes the saturated negative of an integer input (if Out1 > max_int, then Out1 = max_int).
Inputs	In1: 'T1 , with 'T1 = int8, int16, or int32
Outputs	Out1: 'T1
Caution	Implemented only for signed types int8, int16 and int32. iNegSat(min_type) is saturated to max_type, the largest possible value for the type.

# Integer Subtraction (Saturation)

Symbol 3.12:



SCADE name	iSubSat
Function	Subtracts an integer input from another with saturation (if Out1 > max_int, then Out1 = max_int; if Out1 < min_int, then Out1 = min_int).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1
Caution	If the operation overflows relatively to the input type (size and signed/unsigned), the result is the max of the type. If it overflows below the lower limit, the result is the min of the type.

# Integer Sum (Saturation)

Symbol 3.13:



SCADE name	iSumSat
Function	Computes the addition of two integer inputs with saturation (if Out1 > max_int, then Out1 = max_int; if Out1 < min_int, then Out1 = min_int).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 = 'T2 and 'T1, 'T2 as int8, int16, int32, uint8, uint16, or uint32 type
Outputs	Out1: 'T1
Caution	If the operation overflows relatively to the input type (size and signed/unsigned), the result is the max of the type. If it overflows below the lower limit, the result is the min of the type.

# Comparison Operators

All comparison operators defined in this library have as argument the following types: 'T1 and 'T2. This allows mixing types of different sizes, but does not support mixing signed and unsigned types. Types 'T1 and 'T2 can be any type from int8, int16, int32, or any type from uint8, uint16, uint32.

Mixing different size of implementation types is allowed for efficiency reason: avoiding a cast in SCADE models to make the types match may save an intermediate variable in comparison to the implicit cast performed in C code.

Read the description of the following operators:

["Strictly Less Than"](#)

["Less Than or Equal"](#)

["Equal"](#)

["Different"](#)

["Strictly Greater Than"](#)

["Greater Than or Equal"](#)

## Strictly Less Than

Symbol 3.14:



SCADE name	iLess
Function	Compares two integer inputs to check whether the second one is strictly less than the first one and returns the corresponding Boolean data (if In1 < In2, then Out1 = true, else Out1 = false).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 and 'T2 equal to any type from {int8, int16, int32} or 'T1 and 'T2 equal to any type from {uint8, uint16, uint32}
Outputs	Out1: bool



# Less Than or Equal

Symbol 3.15:



SCADE name	iLessEq
Function	Compares two integer inputs to check whether the second one is less than or equal to the first one and returns the corresponding Boolean data (if In1 <= In2, then Out1 = true, else Out1 = false).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 and 'T2 equal to any type from {int8, int16, int32} or 'T1 and 'T2 equal to any type from {uint8, uint16, uint32}
Outputs	Out1: bool

# Equal

Symbol 3.16:



SCADE name	iEqual
Function	Compares equality between two integer inputs and returns the corresponding Boolean data (if In1 = In2, then Output = true, else output = false).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 and 'T2 equal to any type from {int8, int16, int32} or 'T1 and 'T2 equal to any type from {uint8, uint16, uint32}
Outputs	Out1: bool

# Different

Symbol 3.17:



SCADE name	iDifferent
Function	Compares difference between two integer inputs and returns the corresponding Boolean data (if In1 != In2, then Output = true, else output = false).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 and 'T2 equal to any type from {int8, int16, int32} or 'T1 and 'T2 equal to any type from {uint8, uint16, uint32}
Outputs	Out1: bool

# Strictly Greater Than

Symbol 3.18:



SCADE name	iGreater
Function	Compares two integer inputs to check whether the second one is strictly greater than the first one and returns the corresponding Boolean data (if In1 > In2, then Out1 = true, else Out1 = false).
Inputs	In1: 'T1 and In2: 'T2, with 'T1 and 'T2 equal to any type from {int8, int16, int32} or 'T1 and 'T2 equal to any type from {uint8, uint16, uint32}
Outputs	Out1: bool

# Greater Than or Equal

Symbol 3.19:



SCADE name	iGreaterEq
Function	Compares two integer inputs to check whether the second one is greater than or equal to the first one and returns the corresponding Boolean data (if In1 >= In2, then Out1 = true, else Out1 = false).
Inputs	In1: 'T1 and In2: 'T2 , with 'T1 and 'T2 equal to any type from {int8, int16, int32) or 'T1 and 'T2 equal to any type from {uint8, uint16, uint32}
Outputs	Out1: bool

# Bitshift and Bitwise Operators

All bitshift and bitwise operators defined in this library can have their first operand of any type from `int8`, `uint8`, `int16`, `uint16`, `int32`, or `uint32`. The second operand is a hidden parameter of type `'T2'`. It can be either:

- 1 Type `"int"` that is not an `"implemented type"`. The objective is to use only literals and not variables of type `int`. A direct value can be set as a parameter from the graphical symbol.

Example: `iShiftRight(_L1, 2)`. This literal value is interpreted as the type of the first operand, that is, implementation casts it to the type of the first operand. No overflow check is performed on this cast.

- 2 Same type as the first operand.

Read the description of the following operators:

- ["Bit Clear"](#)
- ["Extract Bit Range"](#)
- ["Increase LSB to Floor"](#)
- ["Bit Set"](#)
- ["Increase LSB to Ceiling"](#)
- ["Shift Left"](#)
- ["Decrease LSB"](#)
- ["Increase LSB to Fix"](#)
- ["Shift Right"](#)

## Bit Clear

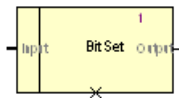
Symbol 3.20:



SCADE name	BitClear
Function	Returns the input where the bit at position <code>Index</code> was set to 0
Inputs	<code>Input</code> : 'T' where 'T' is unsigned
Hidden inputs	<code>Index</code> : 'T' where 'T' is unsigned
Outputs	<code>Output</code> : 'T' where 'T' is unsigned

# Bit Set

Symbol 3.21:



SCADE name	BitSet
Function	Returns the input where the bit at position Index was set to 1
Inputs	Input: 'T where 'T is unsigned
Hidden inputs	Index: 'T where 'T is unsigned
Outputs	Output: 'T where 'T is unsigned

# Decrease LSB

Symbol 3.22:



SCADE name	iDecLsb
Function	Computes a multiplication by a power of 2 (implemented as a logical bit shift to the left). Same behavior as <a href="#">"Shift Left"</a> .
Inputs	In1: 'T1 , with 'T1 = int8, int16, int32, uint8, uint16 or uint32
Hidden inputs	Shift : uint8
Outputs	Out1: 'T1
Caution	Overflow if one of the bits shifted outside the binary representation of the type has value 1. Used for decreasing the scale of fixed-point data flow ( <i>e.g.</i> , from lsb -2 to lsb -4) corresponds to a multiplication by a power of 2, implemented by a shift left.

# Extract Bit Range

Symbol 3.23:



SCADE name	ExtractBitRange
Function	Takes the bits of <code>Input</code> from index <code>Start</code> to index <code>End</code> and returns a value that has exactly those bits (filling the beginning with zeros)
Inputs	<code>Input</code> : uint32
Hidden inputs	<code>Start</code> : uint32 <code>End</code> : uint32
Outputs	<code>Output</code> : uint32

# Increase LSB to Ceiling

Symbol 3.24:



SCADE name	iInclsbCeil
Function	Computes a division by a power of 2 (implemented as a logical bit shift to the right) with rounding toward ceiling.
Inputs	<code>In1</code> : 'T1', with 'T1' = int8, int16, int32, uint8, uint16 or uint32
Hidden inputs	<code>Shift</code> : uint8
Outputs	<code>Out1</code> : 'T1'
Caution	Used for increasing the scale of fixed-point data flow (e.g., from lsb -4 to lsb -2) corresponds to a division by a power of 2, implemented by a shift right.

# Increase LSB to Fix

Symbol 3.25:



SCADE name	iIncLsbFix
Function	Computes a division by a power of 2 (implemented as a logical bit shift to the right) with rounding toward zero.
Inputs	In1: 'T1 , with 'T1 = int8, int16, int32, uint8, uint16, or uint32
Hidden inputs	Shift : uint8
Outputs	Out1: 'T1
Caution	Used for increasing the scale of fixed-point data flow (e.g., from lsb -4 to lsb -2) corresponds to a division by a power of 2, implemented by a shift right.

# Increase LSB to Floor

Symbol 3.26:



SCADE name	iIncLsbFloor
Function	Computes a division by a power of 2 (implemented as a logical bit shift to the right) with rounding toward floor. Same operator behavior as <a href="#">"Shift Right"</a> .
Inputs	In1: 'T1 , with 'T1 = int8, int16, int32, uint8, uint16, or uint32
Hidden inputs	Shift : uint8
Outputs	Out1: 'T1
Caution	Used for increasing the scale of fixed-point data flow (e.g., from lsb -4 to lsb -2) corresponds to a division by a power of 2, implemented by a shift right.

# Shift Left

Symbol 3.27:



SCADE name	iShiftLeft
Function	A logical bit shift to the left.
Inputs	In1: 'T1 , with 'T1 = int8, int16, int32, uint8, uint16, or uint32
Hidden inputs	Shift : uint8
Outputs	Out1: 'T1

# Shift Right

Symbol 3.28:



SCADE name	iShiftRight
Function	A logical bit shift to the right.
Inputs	In1: 'T1 , with 'T1 = int8, int16, int32, uint8, uint16, or uint32
Hidden inputs	Shift : uint8
Outputs	Out1: 'T1
Comment	Corresponds to an arithmetic division rounded to floor with no underflow check. Result of division can be rounded in different ways using other operators like <a href="#">"Increase LSB to Ceiling"</a> , <a href="#">"Increase LSB to Fix"</a> , or <a href="#">"Increase LSB to Floor"</a> .



# 4 / Library liblinear

This library contains operators that compute linear functions, discrete filters, or discrete transfer functions. You can read the technical description of the library components by package:

- [“Filter and Transfer Function Operators”](#) in **filters** package
- [“Linear Function Operators”](#) in **linear** package

Access the description of library operators in alphabetical order:

<a href="#">“Backlash”</a>	<a href="#">“Discrete Filter (Numerator deg. Ns, Denominator deg. Ds normalized)”</a>
<a href="#">“Derivative”</a>	<a href="#">“Discrete Filter (Numerator deg. Ns, Denominator deg. Ns)”</a>
<a href="#">“Detect (Change)”</a>	<a href="#">“Discrete Filter (Numerator deg. Ns, Denominator deg. Ns normalized)”</a>
<a href="#">“Detect (Decrease)”</a>	<a href="#">“Filter (First Order Loop)”</a>
<a href="#">“Detect (Fall Negative)”</a>	<a href="#">“Gain”</a>
<a href="#">“Detect (Fall Non-Positive)”</a>	<a href="#">“Hit Crossing (Either Direction)”</a>
<a href="#">“Detect (Increase)”</a>	<a href="#">“Hit Crossing (Falling Direction)”</a>
<a href="#">“Detect (Rise Non-Negative)”</a>	<a href="#">“Hit Crossing (Rising Direction)”</a>
<a href="#">“Detect (Rise Positive)”</a>	<a href="#">“Integrator (Backward)”</a>
<a href="#">“Discrete Filter (Numerator scalar, Denominator deg. 1)”</a>	<a href="#">“Integrator (Forward)”</a>
<a href="#">“Discrete Filter (Numerator scalar, Denominator deg. 1 normalized)”</a>	<a href="#">“Integrator (Trapezoidal)”</a>
<a href="#">“Discrete Filter (Numerator scalar, Denominator deg. 2)”</a>	<a href="#">“Mean Cycle (2 values)”</a>
<a href="#">“Discrete Filter (Numerator scalar, Denominator deg. 2 normalized)”</a>	<a href="#">“Mean Cycle (3 values)”</a>
<a href="#">“Discrete Filter (Numerator scalar, Denominator deg. Ds)”</a>	<a href="#">“Memory”</a>
<a href="#">“Discrete Filter (Numerator scalar, Denominator deg. Ds normalized)”</a>	<a href="#">“Memory (Basic)”</a>
<a href="#">“Discrete Filter (Numerator deg. 1, Denominator deg. 1)”</a>	<a href="#">“Transfer Function (Numerator scalar, Denominator deg. 1)”</a>
<a href="#">“Discrete Filter (Numerator deg. 1, Denominator deg. 1 normalized)”</a>	<a href="#">“Transfer Function (Numerator scalar, Denominator deg. 2)”</a>
<a href="#">“Discrete Filter (Numerator deg. 1, Denominator deg. 2)”</a>	<a href="#">“Transfer Function (Numerator scalar, Denominator deg. Ds)”</a>
<a href="#">“Discrete Filter (Numerator deg. 1, Denominator deg. 2 normalized)”</a>	<a href="#">“Transfer Function (Numerator deg. 1, Denominator deg. 2)”</a>
<a href="#">“Discrete Filter (Numerator deg. Ns, Denominator deg. Ds)”</a>	<a href="#">“Transfer Function (Numerator deg. Ns, Denominator deg. Ds)”</a>

# Filter and Transfer Function Operators

Read the description of the following operators:

["Discrete Filter \(Numerator scalar, Denominator deg. 1\)"](#)

["Discrete Filter \(Numerator scalar, Denominator deg. 1 normalized\)"](#)

["Discrete Filter \(Numerator scalar, Denominator deg. 2\)"](#)

["Discrete Filter \(Numerator scalar, Denominator deg. 2 normalized\)"](#)

["Discrete Filter \(Numerator scalar, Denominator deg. Ds\)"](#)

["Discrete Filter \(Numerator scalar, Denominator deg. Ds normalized\)"](#)

["Discrete Filter \(Numerator deg. 1, Denominator deg. 1\)"](#)

["Discrete Filter \(Numerator deg. 1, Denominator deg. 1 normalized\)"](#)

["Discrete Filter \(Numerator deg. 1, Denominator deg. 2\)"](#)

["Discrete Filter \(Numerator deg. 1, Denominator deg. 2 normalized\)"](#)

["Discrete Filter \(Numerator deg. Ns, Denominator deg. Ds\)"](#)

["Discrete Filter \(Numerator deg. Ns, Denominator deg. Ds normalized\)"](#)

["Discrete Filter \(Numerator deg. Ns, Denominator deg. Ns\)"](#)

["Discrete Filter \(Numerator deg. Ns, Denominator deg. Ns normalized\)"](#)

["Transfer Function \(Numerator scalar, Denominator deg. 1\)"](#)

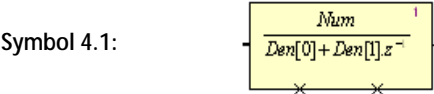
["Transfer Function \(Numerator scalar, Denominator deg. 2\)"](#)

["Transfer Function \(Numerator scalar, Denominator deg. Ds\)"](#)

["Transfer Function \(Numerator deg. 1, Denominator deg. 2\)"](#)

["Transfer Function \(Numerator deg. Ns, Denominator deg. Ds\)"](#)

# Discrete Filter (Numerator scalar, Denominator deg. 1)



SCADE name	Filter01
Package	filters
Function	Discrete filter expressed as a Z-transform function: Num / (Den[0] + Den[1]*z <sup>-1</sup> ) Optimization of FilterND when numerator order is 0 and denominator order is 1.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T Den: 'T ^2 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*lnX)/Den[0].

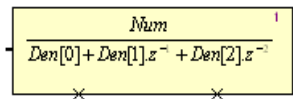
# Discrete Filter (Numerator scalar, Denominator deg. 1 normalized)



SCADE name	Filter01Norm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: Num / (1 + Den*z <sup>-1</sup> ) Optimization of FilterND when numerator order is 0 and denominator order is 1.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T Den: 'T where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*lnX.

# Discrete Filter (Numerator scalar, Denominator deg. 2)

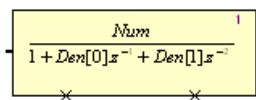
Symbol 4.3:



SCADE name	Filter02
Package	filters
Function	Discrete filter expressed as a Z-transform function: Num / (Den[ 0 ] + Den[ 1 ] * z^-1 + Den[ 2 ] * z^-2 ) Optimization of FilterND when numerator order is 0 and denominator order is 2.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^3 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*InX)/Den[0].

# Discrete Filter (Numerator scalar, Denominator deg. 2 normalized)

Symbol 4.4:



SCADE name	Filter02Norm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: Num / ( 1 + Den[ 0 ] * z^-1 + Den[ 1 ] * z^-2 ) Optimization of FilterND when numerator order is 0 and denominator order is 2.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^2 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*InX.

# Discrete Filter (Numerator scalar, Denominator deg. Ds)

Symbol 4.5:

Num

Den[0]+...+Den[Ds-1]z<sup>-(Ds-1)</sup>

×

×

SCADE name	Filter0D
Package	filters
Function	Discrete filter expressed as a Z-transform function: Num / (Den[0] + Den[1]*z <sup>-1</sup> + ... + Den[Ds - 1]*z <sup>-(Ds - 1)</sup> ) Optimization of FilterND when numerator order is 0 and denominator order is Ds - 1.
Size parameters	Ds is denominator's array size (Ds > 2)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^Ds where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*lnX)/Den[0].

Discrete Filter (Numerator scalar, Denominator deg. Ds normalized)

**Symbol 4.6:**

$$\frac{Num}{1 + Den[0]z^{-1} + \dots + Den[Dc-1]z^{-Dc}}$$

SCADE name	Filter0DNorm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: $\text{Num} / (1 + \text{Den}[0]*z^{-1} + \dots + \text{Den}[\text{Ds} - 1]*z^{-(\text{Ds}-2)})$ Optimization of FilterND when numerator order is 0 and denominator order is Ds.
Size parameters	Ds is denominator's array size (Ds > 2)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^Ds where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*InX.

# Discrete Filter (Numerator deg. 1, Denominator deg. 1)

Symbol 4.7:

$$\frac{Num[0] + Num[1]z^{-1}}{Den[0] + Den[1]z^{-1}}$$

SCADE name	Filter11
Package	filters
Function	Discrete filter expressed as a Z-transform function: (Num[ 0] + Num[ 1]*z^-1) / (Den[ 0] + Den[ 1]*z^-1) Optimization of FilterND when numerator order is 1 and denominator order is 1.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^2 and Den: 'T ^2 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*InX)/Den[0].

# Discrete Filter (Numerator deg. 1, Denominator deg. 1 normalized)

Symbol 4.8:

$$\frac{Num[0] + Num[1]z^{-1}}{1 + Denz^{-1}}$$

SCADE name	Filter11Norm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: (Num[ 0] + Num[ 1]*z^-1) / ( 1 + Den*z^-1) Optimization of FilterND when numerator order is 1 and denominator order is 1.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^2 and Den: 'T where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*InX.

# Discrete Filter (Numerator deg. 1, Denominator deg. 2)

Symbol 4.9:

$$\frac{Num[0]+Num[1].s^{-1}}{Den[0]+Den[1].s^{-1}+Den[2].s^{-2}}$$

×

×

SCADE name	Filter12
Package	filters
Function	Discrete filter expressed as a Z-transform function: (Num[0] + Num[1]*z^-1) / (Den[0] + Den[1]*z^-1 + Den[2]*z^-2) Optimization of FilterND when numerator order is 1 and denominator order is 2.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^2 and Den: 'T ^3 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*InX)/Den[0].

# Discrete Filter (Numerator deg. 1, Denominator deg. 2 normalized)

Symbol 4.10:

$$\frac{Num[0]+Num[1].s^{-1}}{1+Den[0].s^{-1}+Den[1].s^{-2}}$$

×

×

SCADE name	Filter12Norm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: (Num[0] + Num[1]*z^-1) / (1 + Den[0]*z^-1 + Den[1]*z^-2) Optimization of FilterND when numerator order is 1 and denominator order is 2.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^2 and Den: 'T ^2 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*InX.



# Discrete Filter (Numerator deg. Ns, Denominator deg. Ds)

Symbol 4.11:

$$\frac{Num[0] + \dots + Num[Ns - 1] \cdot x^{-(Ns - 1)}}{Den[0] + \dots + Den[Ds - 1] \cdot x^{-(Ds - 1)}}$$

SCADE name	FilterND
Package	filters
Function	Discrete filter expressed as a Z-transform function: (Num[0] + Num[1]*z^-1 + Num[Ns - 1]*z^(Ns - 1)) / (Den[0] + Den[1]*z^-1 + ... + Den[Ds - 1]*z^-(Ds - 1))
Size parameters	Ns is the numerator array size and Ds is the denominator array size (Ds > Ns > 1)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns and Den: 'T ^Ds where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*InX)/Den[0].

# Discrete Filter (Numerator deg. Ns, Denominator deg. Ds normalized)

Symbol 4.12:

$$\frac{Num[0]+...+Num[Ns-1]z^{-(Ns-1)}}{1+Den[0]z^{-1}+...+Den[Ds-1]z^{-Ds}}$$

SCADE name	FilterNDNorm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: (Num[0] + Num[1]*z <sup>-1</sup> + Num[Ns - 1]*z <sup>-(Ns - 1)</sup> ) / (1 + Den[0]*z <sup>-1</sup> + ... + Den[Ds - 1]*z <sup>-Ds</sup> )
Size parameters	Ns is the numerator array size and Ds is the denominator array size (Ds => Ns > 1)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns and Den: 'T ^Ds where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*InX.

# Discrete Filter (Numerator deg. Ns, Denominator deg. Ns)

Symbol 4.13:

$$\frac{Num[0] + \dots + Num[Ns - 1] \cdot x^{-(Ns - 1)}}{Den[0] + \dots + Den[Ns - 1] \cdot x^{-(Ns - 1)}}$$

SCADE name	FilterNN
Package	filters
Function	Discrete filter expressed as a Z-transform function: (Num[0] + Num[1]*z^-1 + ... + Num[Ns - 1]*z^-(Ns - 1)) / (Den[0] + Den[1]*z^-1 + ... + Den[Ns - 1]*z^-(Ns - 1))
Size parameters	Ns is numerator's and denominator's array size (Ns > 1)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns and Den: 'T ^Ns where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = (Num*InX)/Den[0].

# Discrete Filter (Numerator deg. Ns, Denominator deg. Ns normalized)

Symbol 4.14:

$$\frac{Num[0] + \dots + Num[Ns - 1]z^{-(Ns-1)}}{1 + Den[0] + \dots + Den[Ns - 2]z^{-(Ns-1)}}$$

SCADE name	FilterNNNorm
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: (Num[0] + Num[1]*z^-1 + Num[Ns - 1]*z^-(Ns - 1)) / (1 + Den[0]*z^-1 + ... + Den[Ns - 2]*z^-(Ns - 1))
Size parameters	Ns is numerator's array size and Ns - 1 is denominator's array size (Ns > 1)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns and Den: 'T ^ (Ns - 1) where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = Num*InX.

# Transfer Function (Numerator scalar, Denominator deg. 1)

Symbol 4.15:

$$\frac{Num}{Den[0]z + Den[1]}$$

SCADE name	TransferFcn01
Package	filters
Function	Discrete transfer function expressed as a Z-transform function: (Num) / (Den[0]*z + Den[1]) Optimization of TransferFcnND when numerator order is 0 and the denominator order is 1.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^2 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = 0.

# Transfer Function (Numerator scalar, Denominator deg. 2)

Symbol 4.16:

$$\frac{Num}{Den[0]z^2 + Den[1]z + Den[2]}$$

×

×

SCADE name	TransferFcn02
Package	filters
Function	Discrete transfer function expressed as a Z-transform function: (Num) / (Den[0]*z^2 + Den[1]*z + Den[2]) Optimization of TransferFcnND when numerator order is 0 and the denominator order is 2.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^3 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = 0.

# Transfer Function (Numerator scalar, Denominator deg. Ds)

Symbol 4.17:

$$\frac{Num}{Den[0]z^{Ds-1} + \dots + Den[Ds-1]}$$

×

×

SCADE name	TransferFcn0D
Package	filters
Function	Discrete transfer function expressed as a Z-transform function: (Num) / (Den[0]*z^(Ds - 1) + ... + Den[Ds - 2]*z + Den[Ds-1]) Optimization of TransferFcnND when numerator order is 0 and the denominator order is Ds - 1.
Size parameters	Ds is denominator's array size (Ds > 2)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T and Den: 'T ^Ds where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = 0.

# Transfer Function (Numerator deg. 1, Denominator deg. 2)

Symbol 4.18:

$$\frac{Num[0].z + Num[1]}{Den[0].z^2 + Den[1].z + Den[2]}$$

×

×

SCADE name	TransferFcn12
Package	filters
Function	Discrete transfer function expressed as a Z-transform function: (Num[0]*z + Num[1]) / (Den[0]*z^2 + Den[1]*z + Den[2]) Optimization of TransferFcnND when numerator order is 1 and the denominator order is 2.
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^2 and Den: 'T ^3 where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = 0.

# Transfer Function (Numerator deg. Ns, Denominator deg. Ds)

Symbol 4.19:

$$\frac{Num[0].z^{Ns-1} + \dots + Num[Ns-1]}{Den[0].z^{Ds-1} + \dots + Den[Ds-1]}$$

×

×

SCADE name	TransferFcnND
Package	filters
Function	Discrete transfer function expressed as a Z-transform function: (Num[0]*z^(Ns - 1) + ... + Num[Ns - 2]*z + Num[Ns - 1]) / (Den[0]*z^(Ds - 1) + ... + Den[Ds - 2]*z + Den[Ds - 1])
Size parameters	Ns is the numerator array size and Ds is the denominator array size (Ds > Ns > 1)
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns and Den: 'T ^Ds where 'T is float
Outputs	Out: 'T where 'T is float
Comment	At first tick, Out = 0.

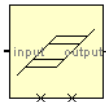
# Linear Function Operators

Read the description of the following operators:

<a href="#">"Backlash"</a>	<a href="#">"Detect (Rise Non-Negative)"</a>	<a href="#">"Integrator (Backward)"</a>
<a href="#">"Derivative"</a>	<a href="#">"Detect (Rise Positive)"</a>	<a href="#">"Integrator (Forward)"</a>
<a href="#">"Detect (Change)"</a>	<a href="#">"Filter (First Order Loop)"</a>	<a href="#">"Integrator (Trapezoidal)"</a>
<a href="#">"Detect (Decrease)"</a>	<a href="#">"Gain"</a>	<a href="#">"Mean Cycle (2 values)"</a>
<a href="#">"Detect (Fall Negative)"</a>	<a href="#">"Hit Crossing (Either Direction)"</a>	<a href="#">"Mean Cycle (3 values)"</a>
<a href="#">"Detect (Fall Non-Positive)"</a>	<a href="#">"Hit Crossing (Falling Direction)"</a>	<a href="#">"Memory"</a>
<a href="#">"Detect (Increase)"</a>	<a href="#">"Hit Crossing (Rising Direction)"</a>	<a href="#">"Memory (Basic)"</a>

## Backlash

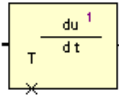
Symbol 4.20:



SCADE name	<b>Backlash</b>
Package	linear
Function	The behavior is as follows: if the input was decreasing before and is now increasing, the output remains constant until it has crossed <code>deadband_width</code> , and then it increases at the same rhythm as the input. And vice versa.
Inputs	<code>input</code> : 'T where 'T is float
Hidden inputs	<code>deadband_width</code> : 'T and <code>initial_output</code> : 'T where 'T is float
Outputs	<code>output</code> : 'T where 'T is float

# Derivative

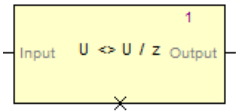
Symbol 4.21:



SCADE name	Derivative
Package	linear
Function	Computes a simple approximation of the input derivative: Output(0)=0.0 Output(k)=(u(k)-u(k-1))/TimeCycle
Inputs	U: 'T where 'T is float
Hidden inputs	TimeCycle: 'T where 'T is float
Outputs	Derivative: 'T where 'T is float
Caution	Very sensitive to noise

# Detect (Change)

Symbol 4.22:

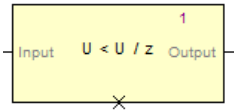


SCADE name	DetectChange
Package	linear
Function	The output is true if the value of the input is different from that of previous cycle
Inputs	Input: 'T where 'T is numeric
Hidden inputs	Init: 'T where 'T is numeric
Outputs	Output: bool



# Detect (Decrease)

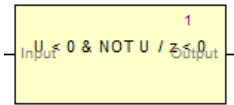
Symbol 4.23:



SCADE name	DetectDecrease
Package	linear
Function	The output is true if the value of the input is strictly lower than that of previous cycle
Inputs	Input: 'T where 'T is numeric
Hidden inputs	Init: 'T where 'T is numeric
Outputs	Output: bool

# Detect (Fall Negative)

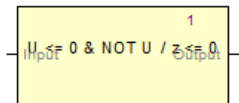
Symbol 4.24:



SCADE name	DetectFallNegative
Package	linear
Function	The output is true if the input is negative and was non-negative on previous cycle
Inputs	Input: 'T where 'T is numeric
Outputs	Output: bool

# Detect (Fall Non-Positive)

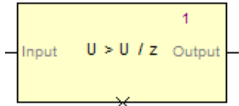
Symbol 4.25:



SCADE name	DetectFallNonPositive
Package	linear
Function	The output is <code>true</code> if the input is non-positive and was positive on previous cycle
Inputs	Input: 'T' where 'T' is numeric
Outputs	Output: bool

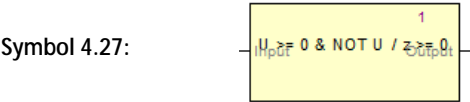
# Detect (Increase)

Symbol 4.26:



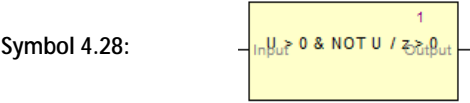
SCADE name	DetectIncrease
Package	linear
Function	The output is <code>true</code> if the value of the input is strictly greater than that of previous cycle
Inputs	Input: 'T' where 'T' is numeric
Hidden inputs	Init: 'T' where 'T' is numeric
Outputs	Output: bool

# Detect (Rise Non-Negative)



SCADE name	DetectRiseNonNegative
Package	linear
Function	The output is <code>true</code> if the input is non-negative and was negative on previous cycle
Inputs	Input: 'T where 'T is numeric
Outputs	Output: bool

# Detect (Rise Positive)



SCADE name	DetectRisePositive
Package	linear
Function	The output is <code>true</code> if the input is positive and was non-positive on previous cycle
Inputs	Input: 'T where 'T is numeric
Outputs	Output: bool

# Filter (First Order Loop)

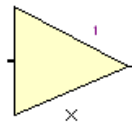
Symbol 4.29:



SCADE name	Filter1stOrderLoop
Package	linear
Function	Creates a customizable first order filter working on two cycles. At initialization, output equals Init. After initialization, output equals K1*Input(k)- k2*Output_R(k-1) where k stands for the current cycle.
Inputs	F1OL_Input: 'T where 'T is numeric
Hidden inputs	K1: 'T - K2: 'T - Init: 'T where 'T is numeric
Outputs	F1OL_Output: 'T where 'T is numeric
KCG pragma	Operator expansion
Comment	Filter created using transfer function: K1/(1+k2*z <sup>-1</sup> ).

## Gain

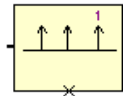
Symbol 4.30:



SCADE name	Gain
Package	linear
Function	Computes the value of an input multiplied by a set gain value. G_Output equals G_Input multiplied by Gain.
Inputs	G_Input: 'T
Hidden inputs	Gain: 'T
KCG pragma	Operator expansion
Outputs	G_Output: 'T where 'T is numeric

# Hit Crossing (Either Direction)

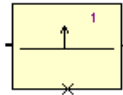
Symbol 4.31:



SCADE name	HitCrossingEither
Package	linear
Function	Detects crossing of a specified value by a signal from either direction. Output is <code>true</code> when the input reaches the offset by growing or decreasing.
Inputs	HCE_Input: 'T
Hidden inputs	Offset: 'T where 'T is numeric
Outputs	HCE _Output: bool

# Hit Crossing (Falling Direction)

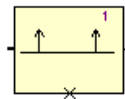
Symbol 4.32:



SCADE name	HitCrossingFalling
Package	linear
Function	Detects crossing of a specified value by a falling signal. Output is <code>true</code> when the input reaches the offset by decreasing.
Inputs	HCF_Input: 'T
Hidden inputs	Offset: 'T where 'T is numeric
Outputs	HCF_Output: bool

# Hit Crossing (Rising Direction)

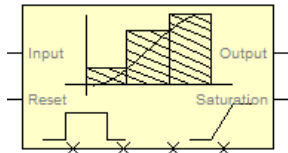
Symbol 4.33:



SCADE name	HitCrossingRising
Package	linear
Function	Detects crossing of a specified value by a rising signal. Output is <code>true</code> when the input reaches the offset by growing.
Inputs	HCR_Input: 'T where 'T is numeric
Hidden inputs	Offset: 'T where 'T is numeric
Outputs	HCR _Output: bool

# Integrator (Backward)

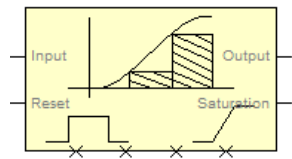
Symbol 4.34:



SCADE name	IntegralBackward
Package	linear
Function	Backward Euler discrete-time integrator. Computes the integral of <code>Input</code> using the backward rule. If the integral is outside of the <code>]LowLimit, Highlimit[</code> range, it is saturated to this range. If saturation is active, the <code>Saturation</code> output is <code>true</code> .
Inputs	Input: 'T where 'T is numeric
Hidden inputs	Reset: bool - HighLimit: 'T (upper bound of integral) - LowLimit: 'T (lower bound of integral) init: 'T - deltaT: 'T where 'T is numeric
Outputs	Output: 'T and Saturation: bool where 'T is numeric

# Integrator (Forward)

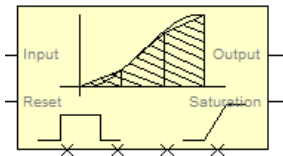
Symbol 4.35:



SCADE name	IntegralForward
Package	linear
Function	Forward Euler discrete-time integrator. The <b>IntegralForward</b> operator computes the integral of <code>Input</code> using the forward rule. If the integral is outside of the <code>]LowLimit, HighLimit[</code> range, it is saturated to this range. If saturation is active, the <code>Saturation</code> output is <code>true</code> .
Inputs	<code>Input</code> : 'T where 'T is numeric and <code>Reset</code> : bool
Hidden inputs	<code>HighLimit</code> : 'T (upper bound of integral)- <code>LowLimit</code> : 'T (lower bound of integral) <code>init</code> : 'T - <code>deltaT</code> : 'T where 'T is numeric where 'T is numeric
Outputs	<code>Output</code> : 'T and <code>Saturation</code> : bool where 'T is numeric

# Integrator (Trapezoidal)

Symbol 4.36:



SCADE name	IntegralTrapezoid
Package	linear
Function	Trapezoidal discrete-time integrator. The <b>IntegralTrapezoid</b> operator computes the integral of <b>Input</b> using the trapezoid rule. If the integral is outside of the <code>]LowLimit, HighLimit[</code> range, it is saturated to this range. If saturation is active, the <b>Saturation</b> output is <code>true</code> .
Inputs	<b>Input</b> : 'T where 'T is numeric and <b>Reset</b> : bool
Hidden inputs	<b>HighLimit</b> : 'T (upper bound of integral) - <b>LowLimit</b> : 'T (lower bound of integral) <b>init</b> : 'T - <b>deltaT</b> : 'T where 'T is numeric
Outputs	<b>Output</b> : 'T and <b>Saturation</b> : bool where 'T is numeric
Comment	Forward integrator $y(k) = y(k-1) + T/2 * (u(k) + u(k-1))$ . Output is limited to the range <code>[LowLimit,HighLimit]</code> . Output is reset to zero at first cycle and when reset is <code>true</code> .



# Mean Cycle (2 values)

Symbol 4.37:

$$\frac{I(k) + I(k-1)}{2}$$

SCADE name	MeanCycle2
Package	linear
Function	Computes the mean of two iterative values of the signal input.  $\text{MeanOn2Step}(0) = \frac{U(0) + U(0)}{2.0}$  $\text{MeanOn2Step}(k) = \frac{U(k) + U(k-1)}{2.0}$
Inputs	U: 'T where 'T is float
Outputs	MeanOn2Step: 'T where 'T is float

# Mean Cycle (3 values)

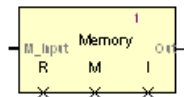
Symbol 4.38:

$$\frac{I(k) + I(k-1) + I(k-2)}{3}$$

SCADE name	MeanCycle3
Package	linear
Function	Computes the mean of three iterative values of the signal input.  $\text{MeanOn3Steps}(0) = \frac{U(0) + U(0) + U(0)}{3.0}$  $\text{MeanOn3Steps}(1) = \frac{U(1) + U(0) + U(0)}{3.0}$  $\text{MeanOn3Steps}(k) = \frac{U(k) + U(k-1) + U(k-2)}{3.0}$
Inputs	U: 'T where 'T is float
Outputs	MeanOn2Step: 'T where 'T is float

# Memory

Symbol 4.39:



SCADE name	Memory
Package	linear
Function	Stores a value with resettable initial value. If Reset is <code>true</code> , then <code>Out = InitVal</code> , else if <code>MemCond</code> is <code>true</code> , <code>Out = Input</code> , else <code>Out</code> equals last stored value.
Inputs	<code>M_Input</code> : 'T where 'T numeric
Hidden inputs	<code>R</code> (Reset): bool - <code>M</code> ( <code>MemCond</code> ): bool - <code>I</code> ( <code>InitVal</code> ): 'T where 'T numeric
Outputs	<code>Out</code> : 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Initialization condition overcomes memorization condition. For any value of <code>MemCond</code> , if <code>Reset</code> is <code>true</code> , <code>Output</code> equals <code>InitVal</code> .

## Memory (Basic)

Symbol 4.40:



SCADE name	MemoryBasic
Package	linear
Function	Stores a value without resetting memory to its initial value. If <code>Write</code> condition is <code>true</code> , then <code>Out = Input</code> , else <code>Out</code> equals last output. Initial value is set to <code>Init</code> .
Inputs	<code>BM_Input</code> : 'T where 'T numeric
Hidden inputs	<code>Init</code> : 'T where 'T numeric and <code>write</code> : bool
Outputs	<code>Memorized</code> : 'T where 'T numeric
KCG pragma	Operator expansion
Comment	At initialization, if <code>Write</code> equals <code>false</code> , the output value and the memory equal <code>Init</code> . <code>Init</code> and <code>BM_Input</code> are often linked to the same source.

# 5 / Library libmath

This library contains basic mathematical operators and mathematical operations on vectors and matrices. You can read the technical description of the library components by package:

- ["Arithmetic Operators"](#) in **math** package
- ["Conversion Operators"](#) in **math** package
- ["Interval Operators"](#) in **math** package
- ["Vector and Matrix Operators"](#) in **vect** package

Access the description of library operators in alphabetical order:

["Absolute Value"](#)

["Boolean to Float32"](#)

["Boolean to Float64"](#)

["Boolean to Int8"](#)

["Boolean to Int16"](#)

["Boolean to Int32"](#)

["Boolean to Int64"](#)

["Boolean to UInt8"](#)

["Boolean to UInt16"](#)

["Boolean to UInt32"](#)

["Boolean to UInt64"](#)

["BuildArray"](#)

["Determinant of Square Matrix \(2x2\)"](#)

["Determinant of Square Matrix \(3x3\)"](#)

["Determinant of Square Matrix \(4x4\)"](#)

["In Range \(In-In\)"](#)

["In Range \(In-Out\)"](#)

["In Range \(Out-In\)"](#)

["In Range \(Out-Out\)"](#)

["Integer to Boolean"](#)

["Inverse of Square Matrix \(2x2\)"](#)

["Inverse of Square Matrix \(3x3\)"](#)

["Inverse of Square Matrix \(4x4\)"](#)

["Matrix Addition"](#)

["Matrix Difference"](#)

["Matrix Product"](#)

["Matrix Product by Vector"](#)

["Maximum \(2 inputs\)"](#)

["Maximum \(3 inputs\)"](#)

["Mean \(2 inputs\)"](#)

["Mean \(3 inputs\)"](#)

["Minimum \(2 inputs\)"](#)

["Minimum \(3 inputs\)"](#)

["Mod"](#)

["Octet to UInt8"](#)

["Polynomial"](#)

["Real to Boolean"](#)

["Rounding Function"](#)

["Rounding to Ceiling Function"](#)

["Rounding to Floor Function"](#)

["Scalar Product"](#)

["Selector"](#)

["Sign"](#)

["Vector Addition"](#)

["Vector Difference"](#)

["Vector Product by Matrix"](#)

# Arithmetic Operators

The `libmath` library contains several basic arithmetic operators that allow to handle integer or real values.

Read the description of the following operators:

- ["Absolute Value"](#)
- ["Maximum \(2 inputs\)"](#)
- ["Maximum \(3 inputs\)"](#)
- ["Mean \(2 inputs\)"](#)
- ["Mean \(3 inputs\)"](#)
- ["Minimum \(2 inputs\)"](#)
- ["Minimum \(3 inputs\)"](#)
- ["Mod"](#)
- ["Polynomial"](#)
- ["Sign"](#)

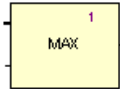
## Absolute Value



SCADE name	Abs
Package	math
Function	Computes the absolute value of real or integer values.
Inputs	A_Input: 'T where 'T float
Outputs	A_Output: 'T where 'T float
KCG pragma	Operator expansion
Comment	If the input is positive or null then the output is equal to the input, else the output is equal to the opposite of input.

# Maximum (2 inputs)

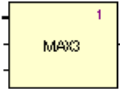
Symbol 5.2:



SCADE name	Max
Package	math
Function	Returns the maximum output from two real or integer values (if In1 >= In2, return In1, else return In2).
Inputs	I1 and I2: 'T where 'T numeric
Outputs	Ma_Output: 'T where 'T numeric
KCG pragma	Operator expansion

# Maximum (3 inputs)

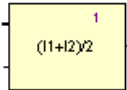
Symbol 5.3:



SCADE name	Max3
Package	math
Function	Returns the maximum output from three real or integer values (if In1 >= In2 AND In1 >= In3, return In1, else if In2 >= In3, return In2, else return In3).
Inputs	I1, I2, and I3: 'T where 'T numeric
Outputs	Ma3_Output: 'T where 'T numeric

# Mean (2 inputs)

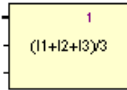
Symbol 5.4:



SCADE name	Mean
Package	math
Function	Computes the mean of two real inputs. $( I1 + I2 ) / 2.0$
Inputs	I1 and I2: 'T where 'T numeric
Outputs	Me_output: 'T where 'T numeric
KCG pragma	Operator expansion

# Mean (3 inputs)

Symbol 5.5:



SCADE name	Mean3
Package	math
Function	Computes the mean of three real inputs. $(I1 + I2 + I3) / 3.0$
Inputs	I1 and I2: 'T where 'T numeric
Outputs	Me3_output: 'T where 'T numeric
KCG pragma	Operator expansion

# Minimum (2 inputs)

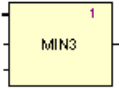
Symbol 5.6:



SCADE name	Min
Package	math
Function	Returns the minimum output from two real or integer values (if In1 <= In2, return In1, else return In2).
Inputs	I1 and I2: 'T where 'T numeric
Outputs	Mi_Output: 'T where 'T numeric
KCG pragma	Operator expansion

# Minimum (3 inputs)

Symbol 5.7:

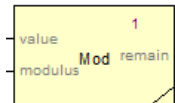


SCADE name	Min3
Package	math
Function	Returns the minimum output from three real or integer values (if In1 <= In2 AND In1 <= In3, return In1, else if In2 <= In3, return In2, else return In3).
Inputs	I1, I2, and I3: 'T where 'T numeric
Outputs	Mi3_Output: 'T where 'T numeric



# Mod

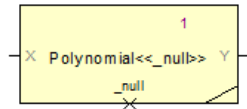
Symbol 5.8:



SCADE name	Mod
Package	math
Function	Computes the modulo function. If the divisor (modulus) is strictly equal to 0, a default value is returned instead.
Inputs	value: 'T where 'T numeric modulus: 'T where 'T numeric
Outputs	remain: 'T where 'T numeric

# Polynomial

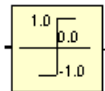
Symbol 5.9:



SCADE name	Polynomial
Package	math
Function	If C is the list $[C_{(N-1)}, C_{(N-2)}, \dots, C_0]$ , the operator computes the following: $C_{(N-1)} \cdot X^{(N-1)} + C_{(N-2)} \cdot X^{(N-2)} + \dots + C_0$
Inputs	X: 'T
Hidden inputs	C: 'T ^N where 'T numeric
Outputs	Y: 'T where 'T numeric

# Sign

Symbol 5.10:



SCADE name	Sign
Package	math
Function	Extracts the sign (+ or -) of a real input and returns a real value from set {-1.0, 0.0, 1.0} based on the sign of the input. If input is positive, then output is equal to 1. If input is negative, then output is equal to -1. If input is 0, then output is equal to 0.
Inputs	S_Input: 'T where 'T numeric
Outputs	S_Output: 'T where 'T numeric
Caution	Relies on a strict equality between real numbers for the 0.0 value.

# Conversion Operators

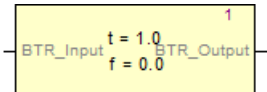
The `libmath` library contains operators that allow to convert some input values or round real numbers.

Read the description of the following operators:

- ["Boolean to Float32"](#)
- ["Boolean to Float64"](#)
- ["Boolean to Int8"](#)
- ["Boolean to Int16"](#)
- ["Boolean to Int32"](#)
- ["Boolean to Int64"](#)
- ["Boolean to UInt8"](#)
- ["Boolean to UInt16"](#)
- ["Boolean to UInt32"](#)
- ["Boolean to UInt64"](#)
- ["Integer to Boolean"](#)
- ["Octet to UInt8"](#)
- ["Real to Boolean"](#)
- ["Rounding Function"](#)
- ["Rounding to Ceiling Function"](#)
- ["Rounding to Floor Function"](#)

## Boolean to Float32

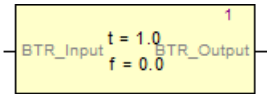
Symbol 5.11:



SCADE name	<b>BoolToFloat32</b>
Package	math
Function	Converts a Boolean value into a real value. When the input is <code>true</code> , the output is equal to 1.0, otherwise it is equal to 0.0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: float32
KCG pragma	Operator expansion

# Boolean to Float64

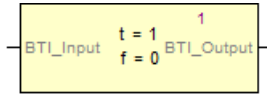
Symbol 5.12:



SCADE name	BoolToFloat64
Package	math
Function	Converts a Boolean value into a real value. When the input is <code>true</code> , the output is equal to 1.0, otherwise it is equal to 0.0.
Inputs	BTR_Input: bool
Outputs	BTR_Output: float64
KCG pragma	Operator expansion

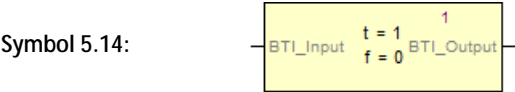
# Boolean to Int8

Symbol 5.13:



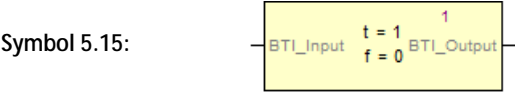
SCADE name	BoolToInt8
Package	math
Function	Converts a Boolean input into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: int8
KCG pragma	Operator expansion

# Boolean to Int16



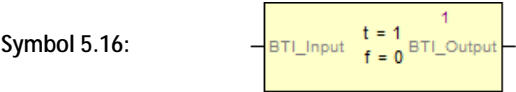
SCADE name	BoolToInt16
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: int16
KCG pragma	Operator expansion

# Boolean to Int32



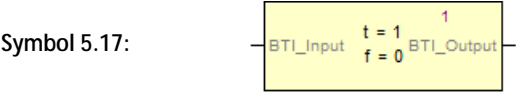
SCADE name	BoolToInt32
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: int32
KCG pragma	Operator expansion

# Boolean to Int64



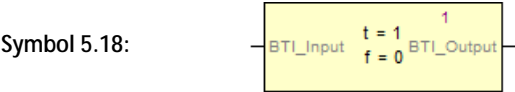
SCADE name	BoolToInt64
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: int64
KCG pragma	Operator expansion

# Boolean to UInt8



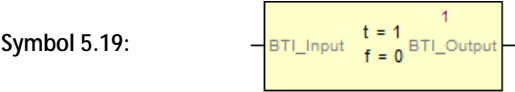
SCADE name	BoolToUInt8
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: uint8
KCG pragma	Operator expansion

# Boolean to Uint16



SCADE name	BoolToUint16
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: uint16
KCG pragma	Operator expansion

# Boolean to Uint32



SCADE name	BoolToUint32
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: uint32
KCG pragma	Operator expansion

# Boolean to Uint64

Symbol 5.20:

BTI\_Input

t = 1

f = 0

BTI\_Output

1

SCADE name	BoolToUint64
Package	math
Function	Converts a Boolean value into an integer value. When the input is <code>true</code> , the output is equal to 1, otherwise the output is equal to 0.
Inputs	BTI_Input: bool
Outputs	BTI_Output: uint64
KCG pragma	Operator expansion

# Integer to Boolean

Symbol 5.21:

≠ 0

1

SCADE name	IntToBool
Package	math
Function	Converts an integer value into a Boolean value. When the input is 0, the output is equal to <code>false</code> , otherwise the output is equal to <code>true</code> .
Inputs	ITB_Input: 'T where 'T is integer
Outputs	ITB_Output: bool
KCG pragma	Operator expansion



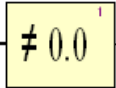
# Octet to Uint8

Symbol 5.22:

SCADE	OctetToUint8
Package	math
Function	Interprets 8 bits (Boolean inputs) as an unsigned integer value between 0 and 255.
Inputs	b1, b2, b3, b4, b5, b6, b7, and b8: bool
Outputs	OTI_Output: uint8

# Real to Boolean

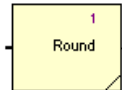
Symbol 5.23:



SCADE name	RealToBool
Package	math
Function	Converts a real value into a Boolean value. When the input is 0.0, the output is equal to <code>false</code> , otherwise the output is equal to <code>true</code> .
Inputs	RTB_Input: 'T where 'T is float
Outputs	RTB_Output: bool
KCG pragma	Operator expansion
Caution	Relies on a strict equality between real numbers.

# Rounding Function

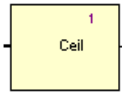
Symbol 5.24:



SCADE name	Round
Package	math
Function	Rounds a real input. R_Output = x when R_Input [ x-0.5, x+0.5 [
Inputs	R_Input: 'T where 'T is float
Outputs	R_Output: int32

# Rounding to Ceiling Function

Symbol 5.25:



SCADE name	RoundCeil
Package	math
Function	Rounds a real input up to the next integer value. RC_Output = x when R_Input ]x-1, x]
Inputs	RC_Input: 'T where 'T is float
Outputs	RC_Output: int32

# Rounding to Floor Function



SCADE name	RoundFloor
Package	math
Function	Rounds a real input down to the next integer value. RF_Output = x when R_Input [x, x+1[
Inputs	RF_Input: 'T where 'T is float
Outputs	RF_Output: int32

# Interval Operators

The `libmath` library contains operators that allow to compute various types of intervals between input values.

Read the description of the following operators:

["In Range \(In-In\)"](#)

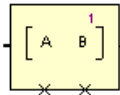
["In Range \(In-Out\)"](#)

["In Range \(Out-In\)"](#)

["In Range \(Out-Out\)"](#)

## In Range (In-In)

Symbol 5.27:



SCADE name	InRangeInIn
Package	math
Function	Returns a <code>true</code> output if the input belongs to the [A , B ] range; otherwise it is <code>false</code> .
Inputs	IRII_Input: 'T where 'T numeric
Hidden inputs	A: 'T (lower border) and B: 'T (upper border) where 'T numeric
Outputs	IRII_Output: bool
KCG pragma	Operator expansion

# In Range (In-Out)



SCADE name	InRangeInOut
Package	math
Function	Returns a <code>true</code> output if the input belongs to the <code>[A , B [</code> range; otherwise it is <code>false</code> .
Inputs	IRIO_Input: 'T where 'T numeric Value in the [-MaximumValue ; MaximumValue] range.
Hidden inputs	A: 'T (lower border) and B: 'T (upper border) where 'T numeric
Outputs	IRIO_Output: bool
KCG pragma	Operator expansion

# In Range (Out-In)



SCADE name	InRangeOutIn
Package	math
Function	Returns a <code>true</code> output if the input belongs to the <code>] A , B ]</code> range; otherwise it is <code>false</code> .
Inputs	IROI_Input: 'T where 'T numeric
Hidden inputs	A: 'T (lower border) and B: 'T (upper border) where 'T numeric
Outputs	IROI_Output: bool
KCG pragma	Operator expansion

# In Range (Out-Out)



SCADE name	InRangeOutOut
Package	math
Function	Returns a <code>true</code> output if the input belongs to the <code>] A , B [</code> range; otherwise it is <code>false</code> .
Inputs	IROO_Input: 'T where 'T numeric
Hidden inputs	A: 'T (lower border) and B: 'T (upper border) where 'T numeric
Outputs	IROO_Output: bool
KCG pragma	Operator expansion

# Vector and Matrix Operators

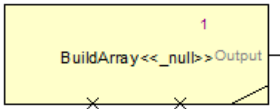
The `libmath` library contains several basic operators that allow to handle vectors or matrices.

Read the description of the following operators:

- ["BuildArray"](#)  
["Determinant of Square Matrix \(2x2\)"](#)  
["Determinant of Square Matrix \(3x3\)"](#)  
["Determinant of Square Matrix \(4x4\)"](#)  
["Inverse of Square Matrix \(2x2\)"](#)  
["Inverse of Square Matrix \(3x3\)"](#)  
["Inverse of Square Matrix \(4x4\)"](#)  
["Matrix Addition"](#)
- ["Matrix Difference"](#)  
["Matrix Product"](#)  
["Matrix Product by Vector"](#)  
["Scalar Product"](#)  
["Selector"](#)  
["Vector Addition"](#)  
["Vector Difference"](#)  
["Vector Product by Matrix"](#)

## BuildArray

Symbol 5.31:



SCADE name	BuildArray
Package	vect
Function	Builds an array of values where the first value is <code>StartValue</code> and every other value is "Increment"-higher than the previous one. For instance: if <code>StartValue</code> is 4.5 and <code>Increment</code> is 0.5, the array is [4.5, 5.0, 5.5, 6.0...].
Hidden inputs	<code>StartValue</code> : 'T where 'T numeric <code>Increment</code> : 'T where 'T numeric
Outputs	<code>Output</code> : 'T ^N where 'T numeric

# Determinant of Square Matrix (2x2)

Symbol 5.32:



SCADE name	Det2x2
Package	vect
Function	Computes the determinant of a square matrix of type 'T^2^2
Inputs	A: 'T ^2^2
Outputs	Det: 'T where 'T numeric

# Determinant of Square Matrix (3x3)

Symbol 5.33:



SCADE name	Det3x3
Package	vect
Function	Computes the determinant of a square matrix of type 'T^3^3
Inputs	A: 'T ^3^3
Outputs	Det: 'T where 'T numeric

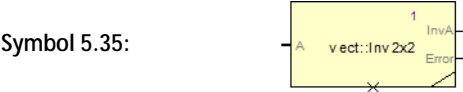


# Determinant of Square Matrix (4x4)



SCADE name	Det4x4
Package	vect
Function	Computes the determinant of a square matrix of type 'T'^4^4
Inputs	A: 'T ^4^4
Outputs	Det: 'T where 'T numeric

# Inverse of Square Matrix (2x2)



SCADE name	Inv2x2
Package	vect
Function	Inverses a square matrix of type 'T'^2^2
Inputs	A: 'T ^2^2 where 'T is float
Hidden Inputs	Epsilon: 'T where 'T is float
Outputs	InvA: 'T ^2^2 where 'T is float Error: bool
Comment	Error output is set to true when the determinant of the matrix is lower or equal to Epsilon input. It is assumed that Epsilon is a positive value.

# Inverse of Square Matrix (3x3)

Symbol 5.36:



SCADE name	Inv3x3
Package	vect
Function	Inverses a square matrix of type 'T'^3
Inputs	A: 'T'^3 where 'T' is float
Hidden Inputs	Epsilon: 'T' where 'T' is float
Outputs	InvA: 'T'^3 where 'T' is float Error: bool
Comment	Error output is set to true when the determinant of the matrix is lower or equal to Epsilon input. It is assumed that Epsilon is a positive value.

# Inverse of Square Matrix (4x4)

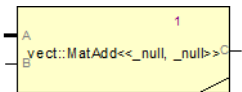
Symbol 5.37:



SCADE name	Inv4x4
Package	vect
Function	Inverses a square matrix of type 'T'^4
Inputs	A: 'T'^4 where 'T' is float
Hidden Inputs	Epsilon: 'T' where 'T' is float
Outputs	InvA: 'T'^4 where 'T' is float Error: bool
Comment	Error output is set to true when the determinant of the matrix is lower or equal to Epsilon input. It is assumed that Epsilon is a positive value.

# Matrix Addition

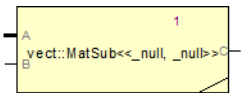
Symbol 5.38:



SCADE name	MatAdd
Package	vect
Function	Computes the sum of two matrices of type 'T <sup>n</sup> <sup>m</sup>
Size parameters	m and n
Inputs	A: 'T <sup>n</sup> <sup>m</sup> B: 'T <sup>n</sup> <sup>m</sup> where 'T numeric
Outputs	C: 'T <sup>n</sup> <sup>m</sup>

# Matrix Difference

Symbol 5.39:



SCADE name	MatSub
Package	vect
Function	Computes the difference of two matrices of type 'T <sup>n</sup> <sup>m</sup>
Size parameters	m and n
Inputs	A: 'T <sup>n</sup> <sup>m</sup> B: 'T <sup>n</sup> <sup>m</sup> where 'T numeric
Outputs	C: 'T <sup>n</sup> <sup>m</sup>

# Matrix Product

Symbol 5.40:



SCADE name	MatProd
Package	vect
Function	Computes the matrix product A*B, matrixes (m,n) implemented as 'T^n^m
Size parameters	m, n, and p
Inputs	A: 'T ^n^m B: 'T ^p^n where 'T numeric
Outputs	C: 'T ^p^m

# Matrix Product by Vector

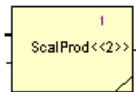
Symbol 5.41:



SCADE name	MatVectProd
Package	vect
Function	Computes the product of a matrix by a vector, matrix (m,n) implemented as 'T^n^m
Size parameters	m and n
Inputs	A: 'T ^n^m V: 'T ^n where 'T numeric
Outputs	R: 'T ^m

# Scalar Product

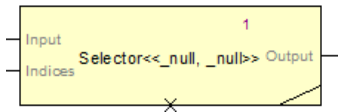
Symbol 5.42:



SCADE name	ScalProd
Package	vect
Function	Computes the scalar product of vectors V and W: $SP = V[0]*W[0] + \dots + V[N-1]*W[N-1]$
Size parameters	n
Inputs	V: 'T ^n W: 'T ^n where 'T numeric
Outputs	SP: 'T
KCG pragma	Operator expansion

# Selector

Symbol 5.43:



SCADE name	Selector
Package	vect
Function	Takes an array Input of size SrcSize and outputs an array of size DstSize where the contents are the elements of the input array at the indices given by Indices. If any of the Indices is outside of the range, the value Default is set in the output array instead.
Inputs	Input: 'T ^SrcSize where 'T numeric Indices: 'I where 'I integer
Hidden inputs	Default: 'T where 'T numeric
Outputs	Output: 'T ^DstSize where 'T numeric

# Vector Addition

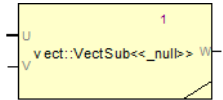
Symbol 5.44:



SCADE name	VectAdd
Package	vect
Function	Computes the addition of two vectors of type 'T^n
Size parameters	n
Inputs	U: 'T ^n V: 'T ^n where 'T numeric
Outputs	W: 'T ^n

# Vector Difference

Symbol 5.45:



SCADE name	VectSub
Package	vect
Function	Computes the difference of two vectors of type 'T^n
Size parameters	n
Inputs	U: 'T ^n V: 'T ^n where 'T numeric
Outputs	W: 'T ^n

# Vector Product by Matrix

Symbol 5.46:



SCADE name	VectMatProd
Package	vect
Function	Computes the product of a vector by a matrix, matrix (m,n) implemented as 'T^n^m
Size parameters	m and n
Inputs	V: 'T ^m A: 'T ^n^m where 'T numeric
Outputs	R: 'T ^n





# 6 / Library libmathext

This library contains trigonometric functions, power-based functions, as well as some advanced functions. All C macro functions necessary for this library are defined in `macro_libmathext32.h` and `macro_libmathext64.h` files located under `SCADE/libraries/SC65/libmathext`.

---

## Note

Depending on the target compiler, you must modify this file.

---

- ["Trigonometric and Power-Based Functions"](#)
- ["Advanced Mathematical Operators"](#)
- ["Conversion Operators"](#)

# Trigonometric and Power-Based Functions

All operators corresponding to these functions are provided for a single output.

Symbol 6.1:



SCADE name	See below			
Package	mathext			
Function	<MathFunction>	SCADE name	<MathFunction>	SCADE name
	cos (x)	CosR	atan2 (x)	Atan2R
	sin (x)	SinR	acosh (x)	AcoshR
	tan (x)	TanR	asinh (x)	AsinhR
	cosh (x)	CoshR	atanh (x)	AtanhR
	sinh (x)	SinhR	ln (x)	LnR
	tanh (x)	TanhR	e <sup>x</sup>	ExpR
	acos (x)	AcosR	log (x)	LogR
	asin (x)	AsinR	√x	SqrtR
	atan (x)	AtanR	10 <sup>x</sup>	TenPowR
		HypotR	(sin(x), cos(x))	SinCosR
Inputs	Input1: 'T Input1 and Input2: 'T for <b>Atan2R</b> HypotR_I1 and HypotR_I2: 'T for <b>HypotR</b> where 'T is float			
Outputs	Output1: 'T Output1 and Output2: 'T for <b>SinCosR</b> HypotR_O: 'T for <b>HypotR</b> where 'T is float			
Comment	Behavior may vary depending on compiler math libraries.			

# Advanced Mathematical Operators

The `libmath` library contains advanced mathematical functions that allow to handle integer and/or real values.

Read the description of the following operators:

["Integer Power for Integers"](#)

["Real Power for Reals"](#)

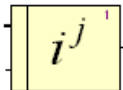
["Integer Power for Reals"](#)

["Square"](#)

["Inverse"](#)

## Integer Power for Integers

Symbol 6.2:



SCADE name	PowerI
Package	mathext
Function	P_Ouput equals P_Input powered to Power.
Dependencies	C macro defined in macro_libmathext32.h and macro_libmathext64.h files under SCADE/libraries/SC65/libmathext
Inputs	P_Input: int32 and Power: int32
Outputs	P_Output: int32

# Integer Power for Reals

Symbol 6.3:



SCADE name	PowerR
Package	mathext
Function	P_Ou <sub>p</sub> ut equals P_In <sub>p</sub> ut powered to P_o <sub>w</sub> er.
Dependencies	C macro defined in macro_libmathext32.h and macro_libmathext64.h files under SCADE/libraries/SC65/libmathext
Inputs	P_In <sub>p</sub> ut: 'T and P_o <sub>w</sub> er: int32 where 'T is float
Outputs	P_Ou <sub>p</sub> ut: 'T where 'T is float

## Inverse

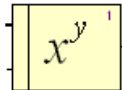
Symbol 6.4:



SCADE name	Inv
Package	mathext
Function	Computes the reciprocal of a real or int number to return a real output
Inputs	Inv_In: 'T where 'T is float
Outputs	Inv_Ou <sub>t</sub> : 'T where 'T is float
Caution	If Inv_In equals 0, division by zero.

# Real Power for Reals

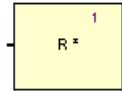
Symbol 6.5:



SCADE name	PowerRR
Package	mathext
Function	P_Output equals P_Input powered to Power, Power being a real. It is assumed that: "P_Input > 0" or "(P_Input == 0 && Power > 0)"
Dependencies	C macro defined in macro_libmathext32.h and macro_libmathext64.h files under SCADE/libraries/SC65/libmathext
Inputs	P_Input: 'T and Power: 'T where 'T is float
Outputs	P_Output: 'T where 'T is float
Comment	According to compiler and function. SCADE simulator checks that P_input is strictly positive, or if null that Power is strictly positive. If it is not the case ( <i>i.e.</i> , P_input is negative), SCADE Simulator raises an exception.

# Square

Symbol 6.6:



SCADE name	Square
Package	mathext
Function	Computes the square value of real or integer numbers.
Inputs	Square_In: 'T where 'T numeric
Outputs	Square_Out: 'T

# Conversion Operators

The `libmathext` library contains conversion functions.

Read the description of the following operators:

["Cartesian to Polar"](#)

["Cartesian to Spherical"](#)

["Celsius to Fahrenheit"](#)

["Degrees to Radians"](#)

["Fahrenheit to Celsius"](#)

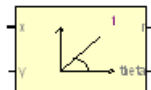
["Polar to Cartesian"](#)

["Radians to Degrees"](#)

["Spherical to Cartesian"](#)

## Cartesian to Polar

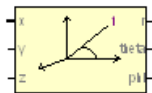
Symbol 6.7:



SCADE name	<b>CartesianToPolar</b>
Package	libmathext
Function	Conversion from Cartesian coordinates (x,y) into polar coordinates
Inputs	x: 'T and y: 'T where 'T is float
Outputs	r: 'T and theta: 'T where 'T is float
KCG pragma	Operator expansion

# Cartesian to Spherical

Symbol 6.8:



SCADE name	CartesianToSpherical
Package	libmathext
Function	Conversion from Cartesian coordinates (x,y,z) into spherical coordinates
Inputs	x: 'T , y: 'T, and z: 'T where 'T is float
Outputs	r: 'T, theta: 'T, and phi: 'T where 'T is float
KCG pragma	Operator expansion

# Celsius to Fahrenheit

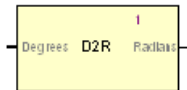
Symbol 6.9:



SCADE name	CelsiusToFahrenheit
Package	libmathext
Function	Conversion from Celsius degrees to Fahrenheit degrees
Inputs	Celsius: 'T where 'T is float
Outputs	Fahrenheit: 'T where 'T is float
KCG pragma	Operator expansion

# Degrees to Radians

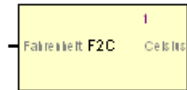
Symbol 6.10:



SCADE name	DegreesToRadians
Package	libmathext
Function	Conversion from degrees to radians
Inputs	Degrees: 'T where 'T is float
Outputs	Radians: 'T where 'T is float
KCG pragma	Operator expansion

# Fahrenheit to Celsius

Symbol 6.11:

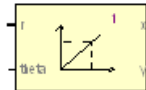


SCADE name	FahrenheitToCelsius
Package	libmathext
Function	Conversion from Fahrenheit degrees to Celsius degrees
Inputs	Fahrenheit: 'T where 'T is float
Outputs	Celsius: 'T where 'T is float
KCG pragma	Operator expansion



# Polar to Cartesian

Symbol 6.12:



SCADE name	PolarToCartesian
Package	libmathext
Function	Conversion from polar coordinates (r,θ) into Cartesian coordinates
Inputs	x: 'T and theta: 'T where 'T is float
Outputs	x: 'T and y: 'T where 'T is float
KCG pragma	Operator expansion

# Radians to Degrees

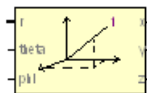
Symbol 6.13:



SCADE name	RadiansToDegrees
Package	libmathext
Function	Conversion from radians to degrees
Inputs	Radians: 'T where 'T is float
Outputs	Degrees: 'T where 'T is float
KCG pragma	Operator expansion

# Spherical to Cartesian

Symbol 6.14:



SCADE name	SphericalToCartesian
Package	libmathext
Function	Conversion from spherical coordinates ( $r, \theta, \varphi$ ) into Cartesian coordinates
Inputs	$r: 'T$ , $\theta: 'T$ , and $\varphi: 'T$ where $'T$ is float
Outputs	$x: 'T$ , $y: 'T$ , and $z: 'T$ where $'T$ is float
KCG pragma	Operator expansion

# 7 / Library libpwlinear

This library contains operators that compute piecewise linear functions and LookUp Table operations. You can read the technical description of the library components by package:

- ["Piecewise Linear Functions"](#) in **pwlinear** package
- ["Look-Up Table Operators"](#) in **lut** package

Access the description of library operators in alphabetical order:

<a href="#">"CheckSlope"</a>	<a href="#">"LUT 1D (Nearest value)"</a>
<a href="#">"Clock Counter"</a>	<a href="#">"LUT 2D"</a>
<a href="#">"Counter"</a>	<a href="#">"LUT 2D (Value above)"</a>
<a href="#">"Dead Band (Asymmetrical)"</a>	<a href="#">"LUT 2D (Value below)"</a>
<a href="#">"Dead Band (Symmetrical)"</a>	<a href="#">"LUT 2D (Nearest value)"</a>
<a href="#">"Hysteresis (Falling)"</a>	<a href="#">"LUT 3D"</a>
<a href="#">"Hysteresis (Rising)"</a>	<a href="#">"LUT 3D (Nearest value)"</a>
<a href="#">"Interpolation 1D"</a>	<a href="#">"MaxReset"</a>
<a href="#">"Interpolation 1D (Floor)"</a>	<a href="#">"MinReset"</a>
<a href="#">"Interpolation 2D"</a>	<a href="#">"Pre-load (Asymmetrical)"</a>
<a href="#">"Interpolation 2D (Floor)"</a>	<a href="#">"Pre-load (Symmetrical)"</a>
<a href="#">"Limiter (Asymmetrical)"</a>	<a href="#">"Pre-LUT"</a>
<a href="#">"Limiter (Symmetrical)"</a>	<a href="#">"Pre-LUT (Direct)"</a>
<a href="#">"LUT 1D"</a>	<a href="#">"Quantizer"</a>
<a href="#">"LUT 1D (Value above)"</a>	<a href="#">"Rate Limiter"</a>
<a href="#">"LUT 1D (Value below)"</a>	

# Piecewise Linear Functions

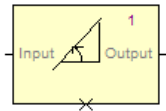
The **libpwlinear** library contains several basic operators that compute piecewise linear functions.

Read the description of the following operators:

- ["CheckSlope"](#)
- ["Clock Counter"](#)
- ["Counter"](#)
- ["Dead Band \(Asymmetrical\)"](#)
- ["Dead Band \(Symmetrical\)"](#)
- ["Hysteresis \(Falling\)"](#)
- ["Hysteresis \(Rising\)"](#)
- ["Limiter \(Asymmetrical\)"](#)
- ["Limiter \(Symmetrical\)"](#)
- ["MaxReset"](#)
- ["MinReset"](#)
- ["Pre-load \(Asymmetrical\)"](#)
- ["Pre-load \(Symmetrical\)"](#)
- ["Quantizer"](#)
- ["Rate Limiter"](#)

## CheckSlope

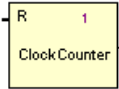
Symbol 7.1:



SCADE name	CheckSlope
Package	pwlinear
Function	Returns true at first cycle if the difference between the value of <code>Input</code> and that of previous cycle is greater than <code>Max</code> (in absolute value)
Inputs	<code>Input</code> : 'T where 'T numeric
Hidden inputs	<code>Max</code> : T where 'T numeric
Outputs	<code>Output</code> : bool

# Clock Counter

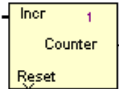
Symbol 7.2:



SCADE name	ClockCounter
Package	pwlinear
Function	Basic counter. Increments the output by one at each cycle. At initialization, or if Reset equals <code>true</code> , output is set to 0.
Inputs	Reset: bool
Outputs	Count: int32

# Counter

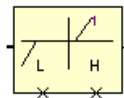
Symbol 7.3:



SCADE name	Counter
Package	pwlinear
Function	The output increments at each cycle by Incr. At initialization or if Reset equals <code>true</code> , the output is set to 0.
Inputs	Incr: 'T where 'T numeric
Hidden inputs	Reset: bool
Outputs	Count: 'T where 'T numeric

# Dead Band (Asymmetrical)

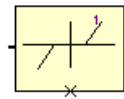
Symbol 7.4:



SCADE name	DeadBandUnSymmetrical
Package	pwlinear
Function	Asymmetrical deadband. Keeps output equal to 0.0 as long as input is in [lower tolerance, upper tolerance] range. When input reaches either limit, output is equal to input minus corresponding limit.
Inputs	DBUS_Input: 'T where 'T numeric
Hidden inputs	LowTol: 'T and HiTol: 'T where 'T numeric
Outputs	DBUS_Output: 'T where 'T numeric
Comment	For a correct behavior, HiTol must be larger than LowTol.

# Dead Band (Symmetrical)

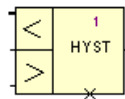
Symbol 7.5:



SCADE name	DeadBandSymmetrical
Package	pwlinear
Function	Basic symmetrical deadband. Keeps output equal to 0.0 as long as input is within [-Tolerance, +Tolerance] range. Before the lower bound, output is equal to the input plus the tolerance; after the higher bound, it is equal to input minus tolerance.
Inputs	DBS_Input: 'T where 'T numeric
Hidden inputs	Tolerance: 'T where 'T numeric
Outputs	DBS_Output: 'T where 'T numeric
Comment	For a correct behavior, Tolerance must be positive.

# Hysteresis (Falling)

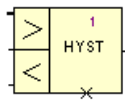
Symbol 7.6:



SCADE name	FallingHysteresis
Package	pwlinear
Function	Detects with hysteresis if the input is above a threshold. Returns a Boolean value (if FH_Input < LL, then output is true, else if FH_Input > UL, then output is false, else output equals previous output).
Inputs	LL (Lower limit), UL (Upper limit), and FH_Input: 'T where 'T numeric
Hidden inputs	Init: bool
Outputs	FH_Output: bool
Comment	On first step, output equals Init if (LL <= FH_Input <= UL). For a correct behavior, UL must be larger than LL.

# Hysteresis (Rising)

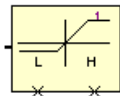
Symbol 7.7:



SCADE name	RisingHysteresis
Package	pwlinear
Function	Detects with hysteresis if the input is below a threshold. Returns a Boolean value (if A > UL, then output is true, else if A < LL, then output is false, else output equals previous output).
Inputs	A, UL (Upper limit), and LL (Lower limit): 'T where 'T numeric
Hidden inputs	Init: bool
Outputs	S: bool
Comment	On first step, output equals Init if (LL <= A <= UL). For a correct behavior, UL must be larger than LL.

# Limiter (Asymmetrical)

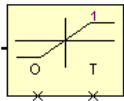
Symbol 7.8:



SCADE name	LimiterUnSymmetrical
Package	pwlinear
Function	Asymmetrical limiting band with specified upper and lower limits, not necessarily including 0. If the input is lower than L, the output is L; if the input is greater than H, the output is H; otherwise the output is equal to the input.
Inputs	LUS_Input: 'T where 'T numeric
Hidden inputs	LowLimit: 'T and HighLimit: 'T where 'T numeric
Outputs	LUS_Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	For a correct behavior, HighLimit must be larger than LowLimit.

# Limiter (Symmetrical)

Symbol 7.9:



SCADE name	LimiterSymmetrical
Package	pwlinear
Function	Basic symmetrical limiting band for some arbitrary values, not necessarily 0. Let's call L the value (origin - tolerance) and U the value (origin + tolerance). If the input is lower than L, the output is L; if the input is greater than U, the output is U; if not output is equal to the input.
Inputs	LS_Input: 'T where 'T numeric
Hidden inputs	BandOrigin: 'T and Tolerance: 'T where 'T numeric
Outputs	LS_Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	For a correct behavior, Tolerance should be positive.



# MaxReset

Symbol 7.10:



SCADE name	MaxReset
Package	pwlinear
Function	When Reset is true and before Init, Output is set to Init. Whenever Input is greater than the previous Output, Output is set to Input.
Inputs	Input: 'T where 'T numeric Reset: bool
Hidden inputs	Init: 'T where 'T numeric
Outputs	Output: 'T where 'T numeric

# MinReset

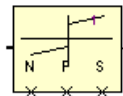
Symbol 7.11:



SCADE name	MinReset
Package	pwlinear
Function	When Reset is true and before Init, Output is set to Init. Whenever Input is lower than the previous Output, Output is set to Input.
Inputs	Input: 'T where 'T numeric Reset: bool
Hidden inputs	Init: 'T where 'T numeric
Outputs	Output: 'T where 'T numeric

# Pre-load (Asymmetrical)

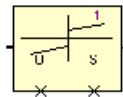
Symbol 7.12:



SCADE name	PreLoadUnSymmetrical
Package	pwlinear
Function	Asymmetrical preload with specified offset when negative or positive, and slope. The output is equal to (input*Slope)+PosOffset if the input is positive or zero, otherwise equal to (input*Slope)+NegOffset if the input is negative.
Inputs	PLUS_Input: 'T where 'T numeric
Hidden inputs	NegOffset: 'T, PosOffset: 'T, and Slope: 'T
Outputs	PLUS_Output: 'T where 'T numeric
KCG pragma	Operator expansion

# Pre-load (Symmetrical)

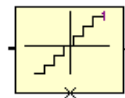
Symbol 7.13:



SCADE name	PreLoadSymmetrical
Package	pwlinear
Function	Basic symmetrical preload with specified offset and slope. The output is equal to (input*slope) + offset if the input is strictly positive, otherwise the output is equal to (input * slope) - offset.
Inputs	PLS_Input: 'T where 'T numeric
Hidden inputs	Offset: 'T and Slope: 'T where 'T numeric
Outputs	PLS_Output: 'T where 'T numeric

# Quantizer

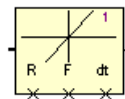
Symbol 7.14:



SCADE name	Quantizer
Package	pwlinear
Function	Basic quantizer based on specified interval. Discretizes the input at a specified interval and produces a stair-step output. $Q\_Output = Interval * Round(Q\_Input / Interval)$
Inputs	$Q\_Input$ : 'T where 'T is float
Hidden inputs	$Interval$ : 'T where 'T is float
Outputs	$Q\_Output$ : 'T where 'T is float
Comment	For a correct behavior, $Interval$ must never be strictly positive. If $Interval$ equals 0, there is a division by 0.

# Rate Limiter

Symbol 7.15:



SCADE name	RateLimiter
Package	pwlinear
Function	Limits the first derivative of the input signal. The output does not change faster than the specified limit. Let's call Rate the derivative $((RL\_Input - pre(RL\_Output)) / \Delta T)$ . If $Rate > Rising$ , then output equals $(Rising * \Delta T + pre(RL\_Output))$ , else if $Rate < Falling$ , then output equals $(Falling * \Delta T + pre(RL\_Output))$ , otherwise, output equals input.
Inputs	$RL\_Input$ : 'T where 'T is float
Hidden inputs	$Rising$ : 'T (Rising slew rate), $Falling$ : 'T (Falling slew rate), and $\Delta T$ : 'T where 'T is float
Outputs	$RL\_Output$ : 'T where 'T is float
Comment	For a correct behavior, $\Delta T$ must be strictly positive. If it equals 0, there is a division by 0.

# Look-Up Table Operators

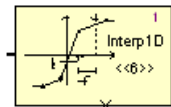
The **libpwlinear** library contains several basic operators that compute Look-Up Table (LUT) operations.

Read the description of the following operators:

<a href="#">"Interpolation 1D"</a>	<a href="#">"LUT 1D (Value below)"</a>	<a href="#">"LUT 3D"</a>
<a href="#">"Interpolation 1D (Floor)"</a>	<a href="#">"LUT 1D (Nearest value)"</a>	<a href="#">"LUT 3D (Nearest value)"</a>
<a href="#">"Interpolation 2D"</a>	<a href="#">"LUT 2D"</a>	<a href="#">"Pre-LUT"</a>
<a href="#">"Interpolation 2D (Floor)"</a>	<a href="#">"LUT 2D (Value above)"</a>	<a href="#">"Pre-LUT (Direct)"</a>
<a href="#">"LUT 1D"</a>	<a href="#">"LUT 2D (Value below)"</a>	
<a href="#">"LUT 1D (Value above)"</a>	<a href="#">"LUT 2D (Nearest value)"</a>	

## Interpolation 1D

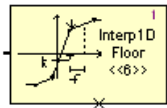
Symbol 7.16:



SCADE name	Interp1D
Package	lut
Function	Uses the pre-calculated index and interval fraction from the PreLut function to interpolate output value from <Xs> points set as vector Y.
Size parameters	Xs
Inputs	IdxX: LutIndex = {k:int32, f:'T'} where 'T' is float
Hidden inputs	Y: 'T' ^Xs where 'T' is float
Outputs	OutY: 'T' where 'T' is float

# Interpolation 1D (Floor)

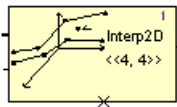
Symbol 7.17:



SCADE name	Interp1DFloor
Package	lut
Function	Uses the calculated index and interval fraction from the PreLut function to return corresponding output value to nearest and below y element from vector Y of size <Xs>. If no value is found then the nearest is returned.
Size parameters	Xs
Inputs	$\text{Id}\times\text{X}$ : LutIndex = {k:int32, f:'T'} where 'T' is float
Hidden inputs	$\text{Y}$ : 'T' ^Xs where 'T' is float
Outputs	$\text{Out}\text{Y}$ : 'T' where 'T' is float

# Interpolation 2D

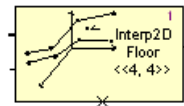
Symbol 7.18:



SCADE name	Interp2D
Package	lut
Function	Uses two precalculated indices and interval fractions from the PreLut function to interpolate output value from <Xs>*<Ys> points set as array Z.
Size parameters	Xs and Ys
Inputs	$\text{Id}\times\text{X}$ : LutIndex = {k:int32, f:'T'} and $\text{Id}\times\text{Y}$ : LutIndex = {k:int32, f:'T'}
Hidden inputs	$\text{Z}$ : 'T' ^Ys^Xs where 'T' is float
Outputs	$\text{Out}\text{Z}$ : 'T' where 'T' is float

# Interpolation 2D (Floor)

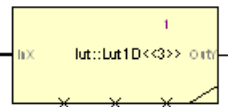
Symbol 7.19:



SCADE name	Interp2DFloor
Package	lut
Function	Uses two precalculated indices and interval fractions from the PreLut function to interpolate output value from <Xs>*<Ys> points set as array Z.
Size parameters	Xs and Ys
Inputs	$I_{dx}X$ : LutIndex = {k:int32, f:'T'} and $I_{dy}Y$ : LutIndex = {k:int32, f:'T'}
Hidden inputs	$z$ : 'T' ^Ys^Xs where 'T' is float
Outputs	$Out_z$ : 'T' where 'T' is float

# LUT 1D

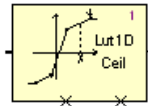
Symbol 7.20:



SCADE name	Lut1D
Package	lut
Function	Performs piecewise linear interpolation to approximate $y = f(x)$ with <Xs> points set as vectors X and Y. X values must increase strictly monotonically. For out-of-bounds input, Extrapol parameter allows linear extrapolation, otherwise end-point values are output.
Size parameters	Xs
Inputs	$InX$ : 'T' where 'T' is float
Hidden inputs	$x$ : 'T' ^Xs, $y$ : 'T' ^Xs, and Extrapol: bool where 'T' is float
Outputs	$OutY$ : 'T' where 'T' is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.
KCG pragma	Operator expansion

# LUT 1D (Value above)

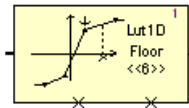
Symbol 7.21:



SCADE name	Lut1DCeil
Package	lut
Function	Returns the output value corresponding to nearest and above x element from <Xs> points set as vectors X and Y. If no value is found then the nearest is returned. X values must increase strictly monotonically.
Size parameters	Xs
Inputs	InX: 'T where 'T is float
Hidden inputs	x: 'T ^Xs and y: 'T ^Xs where 'T is float
Outputs	OutY: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.

# LUT 1D (Value below)

Symbol 7.22:

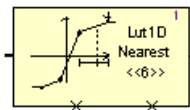


SCADE name	Lut1DFloor
Package	lut
Function	Returns the output value corresponding to nearest and below x element from <Xs> points set as vectors X and Y. If no value is found then the nearest is returned. X values must increase strictly monotonically.
Size parameters	Xs
Inputs	InX: 'T where 'T is float
Hidden inputs	x: 'T ^Xs and y: 'T ^Xs where 'T is float
Outputs	OutY: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.



# LUT 1D (Nearest value)

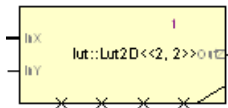
Symbol 7.23:



SCADE name	Lut1DNearest
Package	lut
Function	Returns the output value corresponding to nearest x element from <Xs> points set as vectors X and Y. X values must increase strictly monotonically.
Size parameters	Xs
Inputs	InX: 'T where 'T is float
Hidden inputs	x: 'T ^Xs and y: 'T ^Xs where 'T is float
Outputs	OutY: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.

# LUT 2D

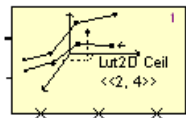
Symbol 7.24:



SCADE name	Lut2D
Package	lut
Function	Performs piecewise linear interpolation to approximate $z = f(x,y)$ with $\langle Xs \rangle$ , $\langle Ys \rangle$ points set as vectors X, Y, and $\langle Xs \rangle * \langle Ys \rangle$ points set as array Z. X and Y values must increase strictly monotonically. For out-of-bounds inputs, Extrapol parameter allows linear extrapolation, otherwise the end-point values are output.
Size parameters	Xs and Ys
Inputs	InX: 'T and InY: 'T where 'T is float
Hidden inputs	x: 'T ^Xs, y: 'T ^Ys, z: 'T ^Ys^Xs, Extrapol: bool where 'T is float
Outputs	OutZ: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.
KCG pragma	Operator expansion

# LUT 2D (Value above)

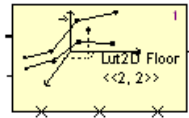
Symbol 7.25:



SCADE name	Lut2DCeil
Package	lut
Function	Returns output value corresponding to nearest and above x and y elements from <Xs>, <Ys> points set as vectors X, Y, and <Xs>*<Ys> points set as array Z. If no value is found then the nearest is returned. X and Y values must increase strictly monotonically.
Size parameters	Xs and Ys
Inputs	InX: 'T and InY: 'T where 'T is float
Hidden inputs	x: 'T ^Xs, y: 'T ^Ys, and z: 'T ^Ys^Xs where 'T is float
Outputs	OutZ: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.

# LUT 2D (Value below)

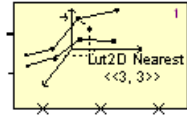
Symbol 7.26:



SCADE name	Lut2DFloor
Package	lut
Function	Returns output value corresponding to nearest and below x and y elements from <Xs>, <Ys> points set as vectors X, Y, and <Xs>*<Ys> points set as array Z. If no value is found then the nearest is returned. X and Y values must increase strictly monotonically.
Size parameters	Xs and Ys
Inputs	InX: 'T and InY: 'T where 'T is float
Hidden inputs	x: 'T ^Xs, y: 'T ^Ys, and z: 'T ^Ys^Xs where 'T is float
Outputs	OutZ: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.

## LUT 2D (Nearest value)

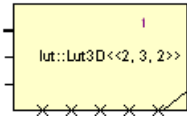
Symbol 7.27:



SCADE name	Lut2DNearest
Package	lut
Function	Returns output value corresponding to nearest x and y elements from <Xs>, <Ys> points set as vectors X, Y, and <Xs>*<Ys> points set as array Z. X and Y values must increase strictly monotonically.
Size parameters	Xs and Ys
Inputs	InX: 'T and InY: 'T where 'T is float
Hidden inputs	x: 'T ^Xs, y: 'T ^Ys, and z: 'T ^Ys^Xs where 'T is float
Outputs	OutZ: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case increase strictly monotonically.

# LUT 3D

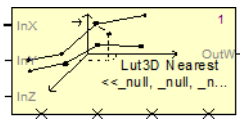
Symbol 7.28:



SCADE name	Lut3D
Package	lut
Function	Performs piecewise linear interpolation to approximate $w = f(x,y,z)$ with $\langle Xs \rangle$ , $\langle Ys \rangle$ , $\langle Zs \rangle$ points set as vectors X, Y, Z, and $\langle Xs \rangle * \langle Ys \rangle * \langle Zs \rangle$ points set as array W. X, Y, and Z values must increase strictly monotonically. For out-of-bounds inputs, Extrapol parameter allows linear extrapolation, otherwise the end-point values are output.
Size parameters	Xs, Ys, and Zs
Inputs	InX: 'T, InY: 'T, and InZ: 'T where 'T is float
Hidden inputs	x: 'T ^Xs, y: 'T ^Ys, z: 'T ^Zs, w: 'T ^Zs^Ys^Xs, and Extrapol: bool where 'T is float
Outputs	OutW: 'T where 'T is float
KCG pragma	Operator expansion

# LUT 3D (Nearest value)

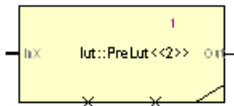
Symbol 7.29:



SCADE name	Lut3DNearest
Package	lut
Function	Returns output value corresponding to nearest x, y, z elements from <Xs>, <Ys>, and <Zs> points set as vectors X, Y, Z, and <Xs>*<Ys>*<Zs> points set as array W. X, Y, and Z values must increase strictly monotonically.
Size parameters	Xs, Ys, and Zs
Inputs	InX: 'T, InY: 'T, and InZ: 'T where 'T is float
Hidden inputs	x: 'T ^Xs, y: 'T ^Ys, z: 'T ^Zs, and w: 'T ^Zs^Ys^Xs where 'T is float
Outputs	OutW: 'T where 'T is float
Comment	An "assume" assertion in all LutXXX operators checks that the X vector in 1D case, and X and Y vectors in 2D case, and X, Y, and Z vectors in 3D case increase strictly monotonically.

# Pre-LUT

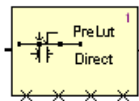
Symbol 7.30:



SCADE name	PreLut
Package	lut
Function	Performs index search and interval fraction calculation from vector X for input InX. X values must increase strictly monotonically. For out-of-bounds input, the computation of interval fraction corresponds to linear extrapolation or to end-point values depending on parameter Extrapol.
Size parameters	Xs
Inputs	InX: 'T where 'T is float
Hidden inputs	X: 'T ^Xs and Extrapol: bool where 'T is float
Outputs	Out: LutIndex = {k:'T, f:int32} where 'T is float

# Pre-LUT (Direct)

Symbol 7.31:



SCADE name	PreLutDirect
Package	lut
Function	Performs index search and interval fraction calculation for input InX when X points are evenly spaced. X points are defined by the first point (Orig), interval between 2 points (step), number of elements (NbElem). For out-of-bounds input, computation of interval fraction corresponds to linear extrapolation or to end-point values depending on parameter Extrapol.
Inputs	InX: 'T where 'T is float
Hidden inputs	Orig: 'T, Step: 'T, NbElem: int32, and Extrapol: bool where 'T is float
Outputs	OutIndex: LutIndex = {k:int32, f:'T} where 'T is float



# 8 / Library libsmk

This library contains operators designed to reproduce the behavior of Simulink blocks with respect to SCADE formalism when translating Simulink models.

- ["Discrete Filter Operators"](#)
- ["Discrete-Time Integrator Operators"](#)
- ["Arithmetic Operators"](#)
- ["Time Operators"](#)
- ["Bitwise Logical Operators"](#)
- ["Conversion and Transformation Operators"](#)
- ["Flip-Flop Operators"](#)
- ["Logical Operators"](#)
- ["Model Verification Operators"](#)
- ["Signal Routing Operators"](#)
- ["Subsystem Operators"](#)
- ["Signal Generator Operators"](#)
- ["Discontinuity Operators"](#)
- ["LUT, Pre-LUT and Interpolation Operators"](#)
- ["Miscellaneous Operators"](#)

Access the description of library operators in alphabetical order:

<a href="#"><u>"AND"</u></a>	<a href="#"><u>"Detect Falling Edge (non-positive)"</u></a>	<a href="#"><u>"Modulo (real/float)"</u></a>
<a href="#"><u>"Assertion"</u></a>	<a href="#"><u>"Detect Increase"</u></a>	<a href="#"><u>"NOT"</u></a>
<a href="#"><u>"Assertion (Inverse)"</u></a>	<a href="#"><u>"Detect Rising Edge (non-negative)"</u></a>	<a href="#"><u>"OR"</u></a>
<a href="#"><u>"Backlash"</u></a>	<a href="#"><u>"Detect Rising Edge (positive)"</u></a>	<a href="#"><u>"Polar to Cartesian"</u></a>
<a href="#"><u>"Bias"</u></a>	<a href="#"><u>"Discrete Filter"</u></a>	<a href="#"><u>"Polynomial"</u></a>
<a href="#"><u>"Bit Clear"</u></a>	<a href="#"><u>"Discrete Filter (Normalized)"</u></a>	<a href="#"><u>"Pre-LUT2"</u></a>
<a href="#"><u>"Bit Set"</u></a>	<a href="#"><u>"Discrete-Time Integrator (Backward)"</u></a>	<a href="#"><u>"Pre-LUT2 (Direct)"</u></a>
<a href="#"><u>"Bitwise AND"</u></a>	<a href="#"><u>"Discrete-Time Integrator (Forward)"</u></a>	<a href="#"><u>"Quantizer"</u></a>
<a href="#"><u>"Bitwise Exclusive OR"</u></a>	<a href="#"><u>"Discrete-Time Integrator (Trapezoidal)"</u></a>	<a href="#"><u>"Radians to Degrees"</u></a>
<a href="#"><u>"Bitwise Nand"</u></a>	<a href="#"><u>"DLatch"</u></a>	<a href="#"><u>"Rate Limiter"</u></a>
<a href="#"><u>"Bitwise Nor"</u></a>	<a href="#"><u>"Entry Detection"</u></a>	<a href="#"><u>"Rate Limiter Dynamic"</u></a>
<a href="#"><u>"Bitwise NOT"</u></a>	<a href="#"><u>"Extract Bit Range"</u></a>	<a href="#"><u>"Real to Boolean"</u></a>
<a href="#"><u>"Bitwise OR"</u></a>	<a href="#"><u>"Enumeration to Integer"</u></a>	<a href="#"><u>"Relay"</u></a>
<a href="#"><u>"Bitwise Shift"</u></a>	<a href="#"><u>"Fahrenheit to Celsius"</u></a>	<a href="#"><u>"Remainder (real/float)"</u></a>
<a href="#"><u>"Bitwise Shift Left"</u></a>	<a href="#"><u>"Flip-Flop D"</u></a>	<a href="#"><u>"Sample Time (all math operations)"</u></a>
<a href="#"><u>"Bitwise Shift Right"</u></a>	<a href="#"><u>"Flip-Flop JK"</u></a>	<a href="#"><u>"Saturate Dynamic"</u></a>
<a href="#"><u>"Boolean to Boolean"</u></a>	<a href="#"><u>"Flip-Flop SR"</u></a>	<a href="#"><u>"Scalar to Boolean"</u></a>
<a href="#"><u>"Border Crossing Detection"</u></a>	<a href="#"><u>"Ground (Boolean)"</u></a>	<a href="#"><u>"Sign (int/int32)"</u></a>
<a href="#"><u>"Cartesian to Polar"</u></a>	<a href="#"><u>"Ground (int/int32)"</u></a>	<a href="#"><u>"Spherical to Cartesian"</u></a>
<a href="#"><u>"Cartesian to Spherical"</u></a>	<a href="#"><u>"Ground (real/float)"</u></a>	<a href="#"><u>"Switch"</u></a>
<a href="#"><u>"Celsius to Fahrenheit"</u></a>	<a href="#"><u>"Identity"</u></a>	<a href="#"><u>"Switch (GTT)"</u></a>
<a href="#"><u>"Check Bounds"</u></a>	<a href="#"><u>"If Then Else"</u></a>	<a href="#"><u>"Switch (NEZ)"</u></a>
<a href="#"><u>"Check Bounds (Extended)"</u></a>	<a href="#"><u>"Ignore First Input"</u></a>	<a href="#"><u>"Switch Enumeration"</u></a>
<a href="#"><u>"Check Gap"</u></a>	<a href="#"><u>"Inactive Cycles"</u></a>	<a href="#"><u>"Switch Enumeration (GTT)"</u></a>
<a href="#"><u>"Check Gap (Extended)"</u></a>	<a href="#"><u>"Inactive Time"</u></a>	<a href="#"><u>"Transfer Function (First Order)"</u></a>
<a href="#"><u>"Check Gradient"</u></a>	<a href="#"><u>"Integer to Boolean"</u></a>	<a href="#"><u>"Transfer Function (Lead or Lag)"</u></a>
<a href="#"><u>"Check Gradient (Extended)"</u></a>	<a href="#"><u>"Integer to Enumeration"</u></a>	<a href="#"><u>"Transfer Function (Real Zero)"</u></a>
<a href="#"><u>"Check Lower Bound"</u></a>	<a href="#"><u>"In Range (In-In)"</u></a>	<a href="#"><u>"Trigger Either Edge"</u></a>
<a href="#"><u>"Check Lower Bound (Extended)"</u></a>	<a href="#"><u>"In Range (In-Out)"</u></a>	<a href="#"><u>"Trigger Either Edge (Extended)"</u></a>
<a href="#"><u>"Check Upper Bound"</u></a>	<a href="#"><u>"In Range (Out-In)"</u></a>	<a href="#"><u>"Trigger Falling Edge"</u></a>
<a href="#"><u>"Check Upper Bound (Extended)"</u></a>	<a href="#"><u>"In Range (Out-Out)"</u></a>	<a href="#"><u>"Trigger Falling Edge (Extended)"</u></a>
<a href="#"><u>"Clock"</u></a>	<a href="#"><u>"Interpolation1D2"</u></a>	<a href="#"><u>"Trigger Rising Edge"</u></a>
<a href="#"><u>"Clock Generator"</u></a>	<a href="#"><u>"Interpolation1D (Floor)"</u></a>	<a href="#"><u>"Trigger Rising Edge (Extended)"</u></a>
<a href="#"><u>"Compare (all operations)"</u></a>	<a href="#"><u>"Interpolation2D2"</u></a>	<a href="#"><u>"Unit Delay (Helper)"</u></a>
<a href="#"><u>"Compare Enumeration"</u></a>	<a href="#"><u>"Interpolation2D (Floor)"</u></a>	<a href="#"><u>"Unit Delay Enabled"</u></a>
<a href="#"><u>"Counter (limited)"</u></a>	<a href="#"><u>"Inverse (int/int32)"</u></a>	<a href="#"><u>"Unit Delay Enabled (Init)"</u></a>
<a href="#"><u>"Dead Zone Dynamic"</u></a>	<a href="#"><u>"LUT 1D"</u></a>	<a href="#"><u>"Unit Delay Enabled Resetable"</u></a>
<a href="#"><u>"Decrement Time to Zero (int/int32)"</u></a>	<a href="#"><u>"Maximum (Resetable)"</u></a>	<a href="#"><u>"Unit Delay Enabled Resetable (Init)"</u></a>
<a href="#"><u>"Decrement Time to Zero (real/float)"</u></a>	<a href="#"><u>"Merge (2 inputs)"</u></a>	<a href="#"><u>"Unit Delay (Init)"</u></a>
<a href="#"><u>"Decrement to Zero"</u></a>	<a href="#"><u>"Merge (3 inputs)"</u></a>	<a href="#"><u>"Unit Delay Resetable"</u></a>
<a href="#"><u>"Degrees to Radians"</u></a>	<a href="#"><u>"Minimum (Resetable)"</u></a>	<a href="#"><u>"Unit Delay Resetable (Init)"</u></a>
<a href="#"><u>"Detect Change"</u></a>	<a href="#"><u>"Modulo (int/int32)"</u></a>	<a href="#"><u>"Width (Matrix)"</u></a>
<a href="#"><u>"Detect Decrease"</u></a>		<a href="#"><u>"Wrap To Zero"</u></a>
<a href="#"><u>"Detect Falling Edge (negative)"</u></a>		

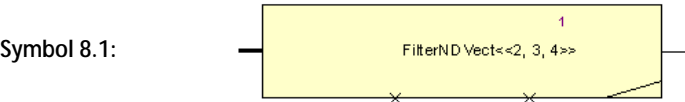
# Discrete Filter Operators

Read the description of the following operators:

[“Discrete Filter”](#)

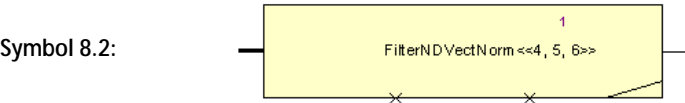
[“Discrete Filter \(Normalized\)”](#)

## Discrete Filter



SCADE names	FilterNDVect
Package	smlk
Function	Vectorized version of filters::FilterND (see <a href="#">“Filter and Transfer Function Operators”</a> ).
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns^Vect and Den : 'T ^Ds where 'T is float
Outputs	Out : 'T ^Vect where 'T is float

## Discrete Filter (Normalized)



SCADE names	FilterNDVectNorm
Package	smlk
Function	Vectorized version of filters::FilterNDNorm (see <a href="#">“Filter and Transfer Function Operators”</a> ).
Inputs	In: 'T where 'T is float
Hidden inputs	Num: 'T ^Ns^Vect and Den : 'T ^Ds where 'T is float
Outputs	Out : 'T ^Vect where 'T is float

# Discrete-Time Integrator Operators

Read the description of the following operators:

["Discrete-Time Integrator \(Backward\)"](#)

["Discrete-Time Integrator \(Forward\)"](#)

["Discrete-Time Integrator \(Trapezoidal\)"](#)

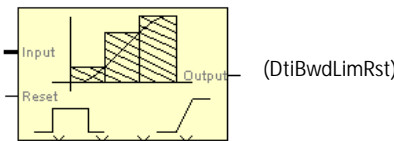
["Transfer Function \(First Order\)"](#)

["Transfer Function \(Lead or Lag\)"](#)

["Transfer Function \(Real Zero\)"](#)

## Discrete-Time Integrator (Backward)

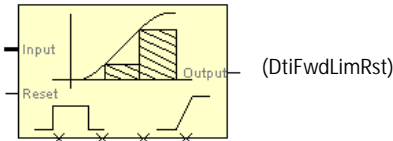
Symbol 8.3:



SCADE names	Several backward types are available in the library: <b>DtiBwdLimRst</b> ; <b>DtiBwdLimRstE</b> ; <b>DtiBwdLimRstF</b> ; <b>DtiBwdLimRstL</b> ; <b>DtiBwdLimRstN</b> ; <b>DtiBwdLimRstR</b> ; <b>DtiBwdLimSRst</b> ; <b>DtiBwdLimSRstE</b> ; <b>DtiBwdLimSRstF</b> ; <b>DtiBwdLimSRstL</b> ; <b>DtiBwdLimSRstN</b> ; <b>DtiBwdLimSRstR</b> ; <b>DtiBwdRst</b> ; <b>DtiBwdRstE</b> ; <b>DtiBwdRstF</b> ; <b>DtiBwdRstL</b> ; <b>DtiBwdRstN</b> ; <b>DtiBwdRstR</b> Naming convention: <b>Bwd</b> indicates Backward Euler Integrator method <b>Lim</b> indicates operators with limits on outputs <b>S</b> indicates saturated operators <b>RstF</b> indicates falling external reset <b>RstR</b> for rising external reset, <b>RstE</b> for either external reset, <b>RstN</b> for none, <b>RstL</b> for external reset level with type inferred for second reset input as int32 or 'T' where 'T' is float, and <b>Rst</b> for external reset level with type inferred for second reset input as bool.
Package	dti
Function	Signal discrete-time integration or accumulation based on Backward Euler method.
Inputs	Input: 'T where 'T is float Reset: 'T or bool (except <b>DtiBwdRstN</b> , <b>DtiBwdLimRstN</b> , <b>DtiBwdLimSRstN</b> )
Hidden inputs	init: 'T and deltaT: 'T HighLimit: 'T and LowLimit: 'T where 'T is float
Outputs	Output: 'T where 'T is float Saturation: int32 (for saturated operators)
KCG pragma	Operator expansion of <b>DtiBwdRstN</b> only

# Discrete-Time Integrator (Forward)

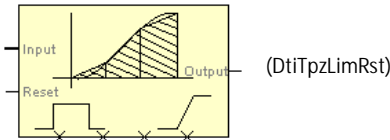
Symbol 8.4:



SCADE name	<p>Several forward types are available in the library:  <b>DtiFwdLimRst</b>; <b>DtiFwdLimRstE</b>; <b>DtiFwdLimRstF</b>; <b>DtiFwdLimRstL</b>; <b>DtiFwdLimRstN</b>;  <b>DtiFwdLimRstR</b>; <b>DtiFwdLimSRst</b>; <b>DtiFwdLimSRstE</b>; <b>DtiFwdLimSRstF</b>; <b>DtiFwdLimSRstL</b>;  <b>DtiFwdLimSRstN</b>; <b>DtiFwdLimSRstR</b>; <b>DtiFwdRst</b>; <b>DtiFwdRstE</b>; <b>DtiFwdRstF</b>; <b>DtiFwdRstL</b>;  <b>DtiFwdRstN</b>; <b>DtiFwdRstR</b></p> <p>Naming convention:  <b>Fwd</b> indicates Forward Euler Integrator method  <b>Lim</b> indicates operators with limits on outputs  <b>S</b> indicates saturated operators  <b>RstF</b> indicates falling external reset, <b>RstR</b> for rising external reset, <b>RstE</b> for either external reset, <b>RstN</b> for none, <b>RstL</b> for external reset level with type inferred for second reset input as int32 or 'T, and <b>Rst</b> for external reset level with type inferred for second reset input as bool.</p>
Package	dti
Function	Signal discrete-time integration or accumulation based on Forward Euler method.
Dependencies	n/a
Inputs	<p>Input: 'T where 'T is float  Reset: 'T or bool (except <b>DtiFwdRstN</b>, <b>DtiFwdLimRstN</b>, <b>DtiFwdLimSRstN</b>)</p>
Hidden inputs	<p>init: 'T and deltaT: 'T  HighLimit: 'T and LowLimit: 'T where 'T is float</p>
Outputs	<p>Output: 'T where 'T is float  Saturation: int32 (for saturated operators)</p>

# Discrete-Time Integrator (Trapezoidal)

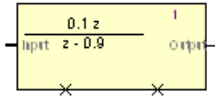
Symbol 8.5:



SCADE name	<p>Several trapezoidal types are available in the library:  <b>DtiTpzLimRst</b>; <b>DtiTpzLimRstE</b>; <b>DtiTpzLimRstF</b>; <b>DtiTpzLimRstL</b>; <b>DtiTpzLimRstN</b>;  <b>DtiTpzLimRstR</b>; <b>DtiTpzLimSRst</b>; <b>DtiTpzLimSRstE</b>; <b>DtiTpzLimSRstF</b>; <b>DtiTpzLimSRstL</b>;  <b>DtiTpzLimSRstN</b>; <b>DtiTpzLimSRstR</b>; <b>DtiTpzRst</b>; <b>DtiTpzRstE</b>; <b>DtiTpzRstF</b>; <b>DtiTpzRstL</b>;  <b>DtiTpzRstN</b>; <b>DtiTpzRstR</b></p> <p>Naming convention:  <b>Tpz</b> indicates Trapezoidal Integrator method  <b>Lim</b> indicates operators with limits on outputs  <b>S</b> indicates saturated operators  <b>RstF</b> indicates falling external reset, <b>RstR</b> for rising external reset, <b>RstE</b> for either external reset, <b>RstN</b> for none, <b>RstL</b> for external reset level with type inferred for second reset input as int32 or 'T, and <b>Rst</b> for external reset level with type inferred for second reset input as bool.</p>
Package	dti
Function	Signal discrete-time integration or accumulation based on Trapezoidal method.
Inputs	<p>Input: 'T where 'T is float  Reset: 'T or bool (except <b>DtiTpzRstN</b>, <b>DtiTpzLimRstN</b>, <b>DtiTpzLimSRstN</b>)</p>
Hidden inputs	<p>init: 'T and deltaT: 'T  HighLimit: 'T and LowLimit: 'T where 'T is float</p>
Outputs	<p>Output: 'T where 'T is float  Saturation: int32 (for saturated operators)</p>

# Transfer Function (First Order)

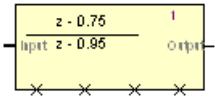
Symbol 8.6:



SCADE name	TransferFcnFirstOrder
Package	smlk
Function	Discrete-time first order transfer function.
Inputs	Input: 'T Pole: 'T where 'T is float
Hidden inputs	IC: 'T where 'T is float
Outputs	Output: 'T where 'T is float

# Transfer Function (Lead or Lag)

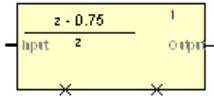
Symbol 8.7:



SCADE name	TransferFcnLeadOrLag
Package	smlk
Function	Discrete-time lead or lag compensator.
Inputs	Input: 'T where 'T is float
Hidden inputs	Pole: 'T Zero: 'T IC_Input: 'T IC_Output: 'T where 'T is float
Outputs	Output: 'T where 'T is float

# Transfer Function (Real Zero)

Symbol 8.8:



SCADE name	TransferFcnRealZero
Package	smlk
Function	Discrete-time transfer function that has real zero and no pole.
Inputs	Input: 'T where 'T is float
Hidden inputs	Zero: 'T IC: 'T where 'T is float
Outputs	Output: 'T where 'T is float



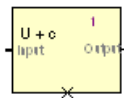
# Arithmetic Operators

Read the description of the following operators:

- ["Bias"](#)  
["Inverse \(int/int32\)"](#)  
["Maximum \(Resettable\)"](#)  
["Minimum \(Resettable\)"](#)  
["Modulo \(int/int32\)"](#)
- ["Modulo \(real/float\)"](#)  
["Polynomial"](#)  
["Remainder \(real/float\)"](#)  
["Sign \(int/int32\)"](#)

## Bias

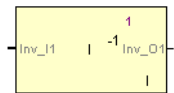
Symbol 8.9:



SCADE name	Bias
Package	smlk
Function	Adds bias to the input.
Inputs	Input: 'T
Hidden inputs	Bias: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion

# Inverse (int/int32)

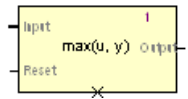
Symbol 8.10:



SCADE name	InvI
Package	smlk
Function	Returns the inverse of an integer input.
Inputs	Inv_I1: int32
Outputs	Inv_O1: int32
KCG pragma	Operator expansion

# Maximum (Resettable)

Symbol 8.11:



SCADE name	MaxReset
Package	smlk
Function	Outputs the maximum of all past inputs. State can be reset.
Inputs	Input: 'T Reset: 'T
Hidden inputs	IC: 'T2
Outputs	Output: 'T where 'T numeric

# Minimum (Resettable)

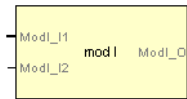
Symbol 8.12:



SCADE name	MinReset
Package	smk
Function	Outputs the minimum of all past inputs. State can be reset.
Inputs	Input: 'T Reset: 'T
Hidden inputs	IC: 'T2
Outputs	Output: 'T where 'T numeric

# Modulo (int/int32)

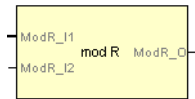
Symbol 8.13:



SCADE name	ModI
Package	smk
Function	Computes the modulus of two integer inputs.
Inputs	ModI_I1: int32 and ModI_I2: int32
Outputs	ModI_O: int32

# Modulo (real/float)

Symbol 8.14:



SCADE name	ModR
Package	smlk
Function	Computes the modulus of two real/float inputs.
Inputs	ModR_I1: 'T and ModR_I2: 'T where 'T is float
Outputs	ModR_O: 'T where 'T is float

# Polynomial

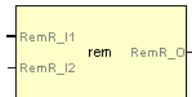
Symbol 8.15:



SCADE name	Polynomial <<N>>
Package	smlk
Function	Performs the evaluation of polynomial coefficients on input values.
Inputs	x: 'T where 'T is numeric
Hidden inputs	c: 'T ^N where 'T is numeric
Outputs	y: 'T where 'T is numeric
KCG pragma	Operator expansion

# Remainder (real/float)

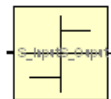
Symbol 8.16:



SCADE name	RemR
Package	smlk
Function	Computes the remainder of a division.
Inputs	RemR_O: 'T and RemR_I 2: 'T where 'T is float
Outputs	RemR_O: 'T where 'T is float

# Sign (int/int32)

Symbol 8.17:



SCADE name	SignInt
Package	smlk
Function	Returns an integer sign value based on the value of a numeric input.
Inputs	S_Input: 'T where 'T numeric
Outputs	S_Output: int32

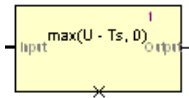
# Time Operators

Read the description of the following operators:

- ["Decrement Time to Zero \(int/int32\)"](#)  
["Decrement Time to Zero \(real/float\)"](#)  
["Decrement to Zero"](#)  
["Sample Time \(all math operations\)"](#)  
["Unit Delay Enabled"](#)  
["Unit Delay Enabled \(Init\)"](#)
- ["Unit Delay Enabled Resettable"](#)  
["Unit Delay Enabled Resettable \(Init\)"](#)  
["Unit Delay \(Init\)"](#)  
["Unit Delay Resettable"](#)  
["Unit Delay Resettable \(Init\)"](#)

## Decrement Time to Zero (int/int32)

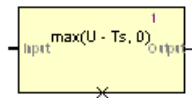
Symbol 8.18:



SCADE name	DecrementTimeToZeroI
Package	smlk
Function	Decreases real-world value of signal by sample time, but only to zero.
Inputs	Input: 'T' where 'T' is integer
Hidden inputs	Ts: float64
Outputs	Output: int32
KCG pragma	Operator expansion

# Decrement Time to Zero (real/float)

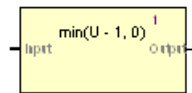
Symbol 8.19:



SCADE name	DecrementTimeToZeroR
Package	smlk
Function	Decreases real-world value of signal by sample time, but only to zero.
Inputs	Input: 'T where 'T is float
Hidden inputs	Ts: 'T where 'T is float
Outputs	Output: 'T where 'T is float
KCG pragma	Operator expansion

# Decrement to Zero

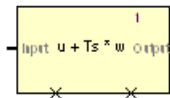
Symbol 8.20:



SCADE name	DecrementToZero
Package	smlk
Function	Decreases real-world value of signal by sample time, but only to zero.
Inputs	Input: 'T
Outputs	Output: 'T where 'T numeric

# Sample Time (all math operations)

Symbol 8.21:



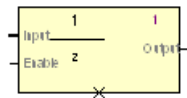
SampleTimeAdd

SCADE name	Several operations on sample time are available in the library: <ul style="list-style-type: none"><li>• Addition: <b>SampleTimeAdd</b></li><li>• Division: <b>SampleTimeDiv</b></li><li>• Inverse: <b>SampleTimeInv</b></li><li>• Multiplication: <b>SampleTimeMult</b></li><li>• Subtraction: <b>SampleTimeSub</b></li></ul>
Package	smk
Function	Computes operations involving sample time.
Inputs	For <b>SampleTimeAdd</b> , <b>SampleTimeMult</b> , <b>SampleTimeSub</b> : Input: 'T' where 'T' is numeric For <b>SampleTimeInv</b> : Input: 'T' where 'T' is float For <b>SampleTimeDiv</b> : Input: 'T' where 'T' is float
Hidden inputs	For <b>SampleTimeAdd</b> , <b>SampleTimeMult</b> , <b>SampleTimeSub</b> : w: 'T' dT: 'T' For <b>SampleTimeDiv</b> : w: 'T' dT: 'T' For <b>SampleTimeInv</b> : w: 'U' dT: 'U' where 'U' is float
Outputs	Output: 'T' where 'T' numeric Output: 'T' (for <b>SampleTimeDiv</b> ) or 'U' (for <b>SampleTimeInv</b> ) where 'T' and 'U' are float
KCG pragma	Operator expansion



# Unit Delay Enabled

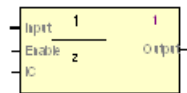
Symbol 8.22:



SCADE name	UDEnabled
Package	smlk
Function	Delays input signal by one tick when Enable input is on, otherwise holds the current state and outputs that value.
Inputs	Input: 'T Enable: 'T1
Hidden inputs	IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Needed for graphical layout purposes of initial condition (IC) input.

# Unit Delay Enabled (Init)

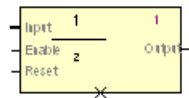
Symbol 8.23:



SCADE name	UDEnabledInit
Package	smlk
Function	Delays input signal by one tick when Enable input is on, otherwise holds the current state and outputs that value.
Inputs	Input: 'T, Enable: 'T1, and IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Initial condition (IC) input is not an hidden input.

# Unit Delay Enabled Resettable

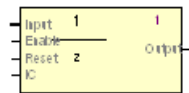
Symbol 8.24:



SCADE name	UDEnabledReset
Package	smlk
Function	Delays input signal by one tick when Enable input is on, otherwise holds the current state and outputs that value. State can be reset based on reset input.
Inputs	Input: 'T, Enable: 'T1, and Reset: 'T2
Hidden inputs	IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Needed for graphical layout purposes of initial condition (IC) input.

# Unit Delay Enabled Resettable (Init)

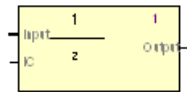
Symbol 8.25:



SCADE name	UDEnabledResetInit
Package	smlk
Function	Delays input signal by one tick when Enable input is on, otherwise holds the current state and outputs that value. State can be reset based on reset input.
Inputs	Input: 'T, Enable: 'T1, Reset: 'T2, and IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Initial condition (IC) input is not an hidden input.

# Unit Delay (Init)

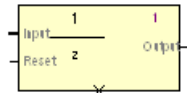
Symbol 8.26:



SCADE name	UDInit
Package	smlk
Function	Delays input signal by one tick.
Inputs	Input: 'T IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Initial condition (IC) input is not an hidden input.

# Unit Delay Resettable

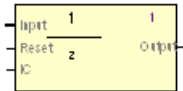
Symbol 8.27:



SCADE name	UDReset
Package	smlk
Function	Delays input signal by one tick. State can be reset based on reset input.
Inputs	Input: 'T Reset: 'T1
Hidden inputs	IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Needed for graphical layout purposes of initial condition (IC) input.

# Unit Delay Resettable (Init)

Symbol 8.28:



SCADE name	UDResetInit
Package	smlk
Function	Delays input signal by one tick. State can be reset based on reset input.
Inputs	Input: 'T Reset: 'T1 IC: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Initial condition (IC) input is not an hidden input.

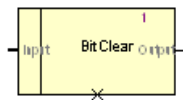
# Bitwise Logical Operators

Read the description of the following operators:

- [“Bit Clear”](#)  
[“Bit Set”](#)  
[“Bitwise AND”](#)  
[“Bitwise Nand”](#)
- [“Bitwise Nor”](#)  
[“Bitwise NOT”](#)  
[“Bitwise OR”](#)  
[“Bitwise Exclusive OR”](#)
- [“Bitwise Shift”](#)  
[“Bitwise Shift Left”](#)  
[“Bitwise Shift Right”](#)  
[“Extract Bit Range”](#)

## Bit Clear

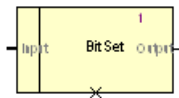
Symbol 8.29:



SCADE name	BitClear (imported function)
Package	bits
Function	Sets specified bit of input value to false.
Inputs	Input: int32
Hidden inputs	Index: int32
Outputs	Output: int32

# Bit Set

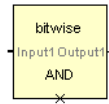
Symbol 8.30:



SCADE name	BitSet (imported function)
Package	bits
Function	Sets specified bit of input value to true.
Inputs	Input: int32
Hidden inputs	Index: int32
Outputs	Output: int32

# Bitwise AND

Symbol 8.31:



SCADE name	BitwiseAnd
Package	bits
Function	Performs a bitwise logical operation AND.
Inputs	Input 1: int32
Hidden inputs	Operand2: int32
Outputs	Output 1: int32

# Bitwise Nand

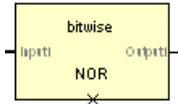
Symbol 8.32:



SCADE name	BitwiseNand
Package	bits
Function	Performs a bitwise logical operation NOT AND.
Inputs	Input1: int32
Hidden inputs	Operand2: int32
Outputs	Output1: int32

# Bitwise Nor

Symbol 8.33:



SCADE name	BitwiseNor
Package	bits
Function	Performs a bitwise logical operation NOT OR.
Inputs	Input1: int32
Hidden inputs	Operand2: int32
Outputs	Output1: int32

# Bitwise NOT

Symbol 8.34:



SCADE name	BitwiseNot
Package	bits
Function	Performs a bitwise logical operation NOT.
Inputs	Input1: int32
Outputs	Output1: int32

# Bitwise OR

Symbol 8.35:

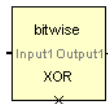


SCADE name	BitwiseOr
Package	bits
Function	Performs a bitwise logical operation OR.
Inputs	Input1: int32
Hidden inputs	Operand2: int32
Outputs	Output1: int32



# Bitwise Exclusive OR

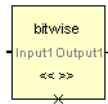
Symbol 8.36:



SCADE name	BitwiseXor
Package	bits
Function	Performs a bitwise exclusive logical operation OR.
Inputs	Input1: int32
Hidden inputs	Operand2: int32
Outputs	Output1: int32

# Bitwise Shift

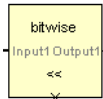
Symbol 8.37:



SCADE name	BitwiseShift
Package	bits
Function	Shifts a specified number of bits.
Inputs	Input1: int32
Hidden inputs	BitShiftRight: int32
Outputs	Output1: int32

# Bitwise Shift Left

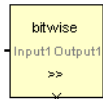
Symbol 8.38:



SCADE name	BitwiseShiftLeft
Package	bits
Function	Shifts left a specified number of bits.
Inputs	Input 1: int32
Hidden inputs	Operand2: int32
Outputs	Output 1: int32

# Bitwise Shift Right

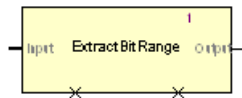
Symbol 8.39:



SCADE name	BitwiseShiftRight
Package	bits
Function	Shifts right a specified number of bits.
Inputs	Input 1: int32
Hidden inputs	Operand2: int32
Outputs	Output 1: int32

# Extract Bit Range

Symbol 8.40:



SCADE name	ExtractBitRange (imported function)
Package	bits
Function	Outputs a selection of contiguous bits from the input signal.
Inputs	Input: int32
Hidden inputs	Start: int32 End: int32
Outputs	Output: int32

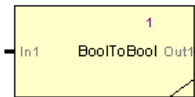
# Conversion and Transformation Operators

Read the description of the following operators:

- ["Boolean to Boolean"](#)  
["Cartesian to Polar"](#)  
["Cartesian to Spherical"](#)  
["Celsius to Fahrenheit"](#)  
["Degrees to Radians"](#)  
["Enumeration to Integer"](#)  
["Fahrenheit to Celsius"](#)
- ["Integer to Boolean"](#)  
["Integer to Enumeration"](#)  
["Polar to Cartesian"](#)  
["Radians to Degrees"](#)  
["Real to Boolean"](#)  
["Scalar to Boolean"](#)  
["Spherical to Cartesian"](#)

## Boolean to Boolean

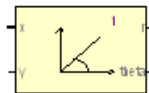
Symbol 8.41:



SCADE name	<b>BoolToBool</b> (specialization of <code>ScalarToBool</code> )
Package	smlkutils
Function	Conversion of scalar values into Boolean values.
Inputs	In1: <code>bool</code>
Outputs	Out1: <code>bool</code>
KCG pragma	Operator expansion

# Cartesian to Polar

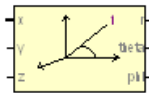
Symbol 8.42:



SCADE name	Cartesian2Polar
Package	conv
Function	Transformation from cartesian to polar coordinates.
Inputs	x: 'T y: 'T where 'T is float
Outputs	r: 'T theta: 'T where 'T is float
KCG pragma	Operator expansion

# Cartesian to Spherical

Symbol 8.43:



SCADE name	Cartesian2Spherical
Package	conv
Function	Transformation from cartesian to spherical coordinates.
Inputs	x: 'T y: 'T z: 'T where 'T is float
Outputs	r: 'T theta: 'T phi: 'T where 'T is float
KCG pragma	Operator expansion

# Celsius to Fahrenheit

Symbol 8.44:



SCADE name	Celsius2Fahrenheit
Package	conv
Function	Conversion from Celsius degrees to Fahrenheit degrees.
Inputs	Celsius: 'T where 'T is float
Outputs	Fahrenheit: 'T where 'T is float
KCG pragma	Operator expansion

# Degrees to Radians

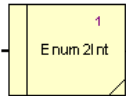
Symbol 8.45:



SCADE name	Degrees2Radians
Package	conv
Function	Conversion from degrees to radians.
Inputs	Degrees: 'T where 'T is float
Outputs	Radians: 'T where 'T is float
KCG pragma	Operator expansion

# Enumeration to Integer

Symbol 8.46:



SCADE name	Enum2Int
Package	conv
Function	Conversion of an enumerated type to its underlying integer value.
Inputs	Input1 : 'T
Outputs	Output1 : int32
Comment	Specialization operators shall be provided by users.

# Fahrenheit to Celsius

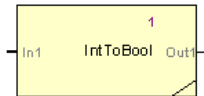
Symbol 8.47:



SCADE name	Fahrenheit2Celsius
Package	conv
Function	Conversion from Fahrenheit degrees to Celsius degrees.
Inputs	Fahrenheit: 'T where 'T is float
Outputs	Celsius: 'T where 'T is float
KCG pragma	Operator expansion

# Integer to Boolean

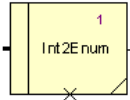
Symbol 8.48:



SCADE name	IntToBool (specialization of ScalarToBool)
Package	smlkutils
Function	Conversion of scalar values into Boolean values.
Inputs	In1: 'T where 'T is integer
Outputs	Out1: bool
KCG pragma	Operator expansion

# Integer to Enumeration

Symbol 8.49:

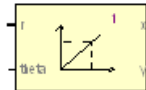


SCADE name	Int2Enum
Package	conv
Function	Conversion of an integer value to the corresponding enumerated value if any, or to Default otherwise.
Inputs	Input1: int32
Hidden inputs	Default: 'T
Outputs	Output1: 'T
Comment	Specialization operators shall be provided by users.



# Polar to Cartesian

Symbol 8.50:



SCADE name	Polar2Cartesian
Package	conv
Function	Transformation from polar to cartesian coordinates.
Inputs	r: 'T theta: 'T where 'T is float
Outputs	x: 'T y: 'T where 'T is float
KCG pragma	Operator expansion

# Radians to Degrees

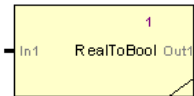
Symbol 8.51:



SCADE name	Radians2Degrees
Package	conv
Function	Conversion from radians to degrees.
Inputs	Radians: 'T where 'T is float
Outputs	Degrees: 'T where 'T is float
KCG pragma	Operator expansion

# Real to Boolean

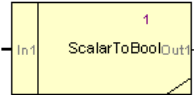
Symbol 8.52:



SCADE name	RealToBool (specialization of ScalarToBool)
Package	smlkutils
Function	Conversion of scalar values into Boolean values.
Inputs	In1: 'T where 'T is float
Outputs	Out1: bool
KCG pragma	Operator expansion

# Scalar to Boolean

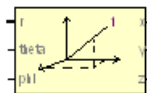
Symbol 8.53:



SCADE name	ScalarToBool (specialized operator)
Package	smlkutils
Function	Conversion of scalar values into Boolean values.
Inputs	In1: 'T where 'T is float
Outputs	Out1: bool

# Spherical to Cartesian

Symbol 8.54:



SCADE name	Spherical2Cartesian
Package	conv
Function	Transformation from spherical to cartesian coordinates.
Inputs	r: 'T , theta: 'T, and phi: 'T where 'T is float
Outputs	x: 'T , y: 'T, and z: 'T where 'T is float
KCG pragma	Operator expansion

# Flip-Flop Operators

Read the description of the following operators:

["Clock"](#)

["DLatch"](#)

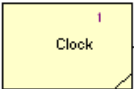
["Flip-Flop D"](#)

["Flip-Flop JK"](#)

["Flip-Flop SR"](#)

## Clock

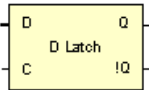
Symbol 8.55:



SCADE name	Clock
Package	smk
Function	Boolean output toggles at each tick.
Outputs	Output1: bool

## DLatch

Symbol 8.56:



SCADE name	DLatch
Package	flipflop
Function	Latches the data input when the clock input is true.
Inputs	D: bool and C: bool
Outputs	Q: bool and notQ: bool
Comment	Output Q equals input D if the input C changes its value. Output notQ is equal to (not Q).

# Flip-Flop D

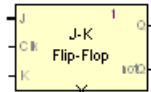
Symbol 8.57:



SCADE name	FlipFlopD
Package	flipflop
Function	Basic D flip-flop of Boolean values.
Inputs	D: bool, Clk: bool, and notClk: bool
Outputs	Q: bool and notQ: bool
Comment	Output Q is equal to Input D if there is a rising edge on input Clk and if the notClk input is true, otherwise Output Q is false. Output notQ is equal to (not Q).

# Flip-Flop JK

Symbol 8.58:



SCADE name	FlipFlopJK
Package	flipflop
Function	Basic JK flip-flop of Boolean values.
Inputs	J: bool, Clk: bool, and K: bool
Hidden inputs	IC: bool
Outputs	Q: bool and notQ: bool
Comment	On falling edge of Clk input, the following computation is done: <ul style="list-style-type: none"><li>• if not J and not K, Qn= Qn-1</li><li>• if not J and K, Qn = false</li><li>• if J and not K, Qn = true</li><li>• if J and K, Qn = not Qn-1</li></ul> Otherwise, Qn = Qn-1 Output notQ is equal to (not Q).

# Flip-Flop SR

Symbol 8.59:



SCADE name	FlipFlopSR
Package	flipflop
Function	Basic Set-Reset flip-flop of Boolean values.
Inputs	S: bool and R: bool
Hidden inputs	IC: bool
Outputs	Q: bool and notQ: bool
Comment	<p>On falling edge of Clk input, the following computation is done:</p> <ul style="list-style-type: none"><li>• if not S and not R, Qn= Qn-1</li><li>• if not S and R, Qn = false</li><li>• if S and not R, Qn = true</li><li>• if S and R, Qn = false</li></ul> <p>Output notQ = if (R and notR) false else (not Qn).</p>

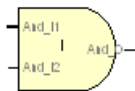
# Logical Operators

Read the description of the following operators:

- ["AND"](#)  
["Compare \(all operations\)"](#)  
["Compare Enumeration"](#)  
["Detect Change"](#)  
["Detect Decrease"](#)  
["Detect Falling Edge \(negative\)"](#)  
["Detect Falling Edge \(non-positive\)"](#)  
["Detect Increase"](#)
- ["Detect Rising Edge \(non-negative\)"](#)  
["Detect Rising Edge \(positive\)"](#)  
["In Range \(In-In\)"](#)  
["In Range \(In-Out\)"](#)  
["In Range \(Out-In\)"](#)  
["In Range \(Out-Out\)"](#)  
["NOT"](#)  
["OR"](#)

## AND

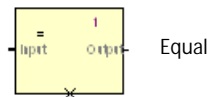
Symbol 8.60:



SCADE name	AndNum
Package	smlk
Function	Performs logical AND operation with numeric inputs.
Inputs	And_I1 and And_I2: 'T where 'T numeric
Outputs	And_O: bool
KCG pragma	Operator expansion

# Compare (all operations)

Symbol 8.61:

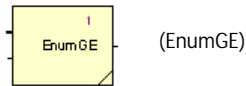


SCADE name	Several comparison operations are available in the library: <ul style="list-style-type: none"><li>• Equal: <b>Equal</b></li><li>• Greater or equal: <b>GreaterEqual</b></li><li>• Greater than: <b>GreaterThan</b></li><li>• Less or equal: <b>LessEqual</b></li><li>• Less than: <b>LessThan</b></li><li>• Not equal: <b>NotEqual</b></li></ul>
Package	cmp
Function	Compares a signal to a specified constant.
Inputs	Input: 'T
Hidden inputs	Value: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion



# Compare Enumeration

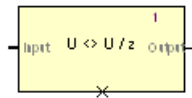
Symbol 8.62:



SCADE name	Several comparison operations are available in the library: <ul style="list-style-type: none"><li>• Greater or equal: <b>EnumGE</b></li><li>• Greater than: <b>EnumGT</b></li><li>• Less or equal: <b>EnumLE</b></li><li>• Less than: <b>EnumLT</b></li></ul>
Package	cmp
Function	Comparison of enumerated values to their conversion into integers (using <code>smlk::EnumToInt</code> specialization)
Inputs	Input1: 'T Input2: 'T
Outputs	Output1: bool
Comment	cmp::EnumLT: Output1 = <code>smlk::EnumToInt(Input1) &lt; smlk::EnumToInt(Input2)</code> cmp::EnumLE: Output1 = <code>smlk::EnumToInt(Input1) &lt;= smlk::EnumToInt(Input2)</code> cmp::EnumGT: Output1 = <code>smlk::EnumToInt(Input1) &gt; smlk::EnumToInt(Input2)</code> cmp::EnumGE: Output1 = <code>smlk::EnumToInt(Input1) &gt;= smlk::EnumToInt(Input2)</code>

# Detect Change

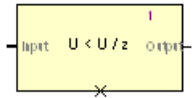
Symbol 8.63:



SCADE name	DetectChange
Package	smlk
Function	Detects if the input value has changed with respect to previous step.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# Detect Decrease

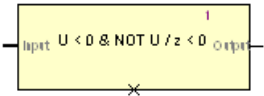
Symbol 8.64:



SCADE name	DetectDecrease
Package	smlk
Function	Detects decrease in signal value.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# Detect Falling Edge (negative)

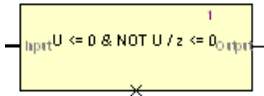
Symbol 8.65:



SCADE name	DetectFallNeg
Package	smlk
Function	Detects falling edge when signal value decreases to strictly negative value and its previous value was non-negative.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# Detect Falling Edge (non-positive)

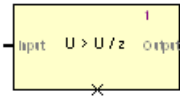
Symbol 8.66:



SCADE name	DetectFallNonPos
Package	smlk
Function	Detects falling edge when signal value decreases to non-positive value and its previous value was strictly positive.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# Detect Increase

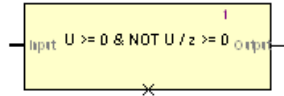
Symbol 8.67:



SCADE name	DetectIncrease
Package	smlk
Function	Detects increase in signal value.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# Detect Rising Edge (non-negative)

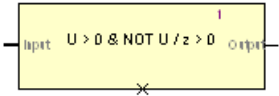
Symbol 8.68:



SCADE name	DetectRiseNonNeg
Package	smlk
Function	Detects rising edge when signal value increases to non-negative value and its previous value was strictly negative.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# Detect Rising Edge (positive)

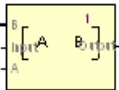
Symbol 8.69:



SCADE name	DetectRisePositive
Package	smlk
Function	Detects rising edge when signal value increases to strictly positive value and its previous value was non-positive.
Inputs	Input: 'T
Hidden inputs	IC: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# In Range (In-In)

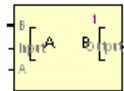
Symbol 8.70:



SCADE name	InRangeInIn
Package	smlk
Function	Returns a true output if the input belongs to the [A , B] range; otherwise it is false.
Inputs	B: 'T Input: 'T A: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# In Range (In-Out)

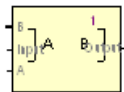
Symbol 8.71:



SCADE name	InRangeInOut
Package	smlk
Function	Returns a <code>true</code> output if the input belongs to the <code>[A , B[</code> range; otherwise it is <code>false</code> .
Inputs	B: 'T Input: 'T A: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# In Range (Out-In)

Symbol 8.72:



SCADE name	InRangeOutIn
Package	smlk
Function	Returns a <code>true</code> output if the input belongs to the <code>]A , B]</code> range; otherwise it is <code>false</code> .
Inputs	B: 'T Input: 'T A: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

# In Range (Out-Out)



SCADE name	InRangeOutOut
Package	smlk
Function	Returns a <code>true</code> output if the input belongs to the ]A , B[ range; otherwise it is <code>false</code> .
Inputs	B: 'T Input: 'T A: 'T where 'T numeric
Outputs	Output: bool
KCG pragma	Operator expansion

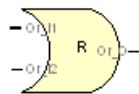
# NOT



SCADE name	NotNum
Package	smlk
Function	Performs logical NOT operation with numeric inputs.
Inputs	Not_I1 and Not_I2: 'T where 'T numeric
Outputs	Not_O: bool
KCG pragma	Operator expansion

# OR

Symbol 8.75:



SCADE name	OrNum
Package	smlk
Function	Performs logical OR operation with numeric inputs.
Inputs	Or_I1 and Or_I2: 'T where 'T numeric
Outputs	Or_O: bool
KCG pragma	Operator expansion



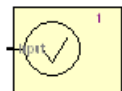
# Model Verification Operators

Read the description of the following operators:

- ["Assertion"](#)  
["Assertion \(Inverse\)"](#)  
["Check Bounds"](#)  
["Check Bounds \(Extended\)"](#)  
["Check Gap"](#)  
["Check Gap \(Extended\)"](#)
- ["Check Gradient"](#)  
["Check Gradient \(Extended\)"](#)  
["Check Lower Bound"](#)  
["Check Lower Bound \(Extended\)"](#)  
["Check Upper Bound"](#)  
["Check Upper Bound \(Extended\)"](#)

## Assertion

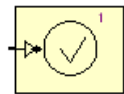
Symbol 8.76:



SCADE name	Assertion
Package	chk
Function	Checks whether signal is nonzero.
Inputs	Input: 'T
Outputs	none
KCG pragma	Operator expansion

# Assertion (Inverse)

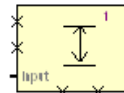
Symbol 8.77:



SCADE name	AssertionInv
Package	chk
Function	Checks whether signal is zero.
Inputs	Input: 'T
Outputs	none
KCG pragma	Operator expansion

# Check Bounds

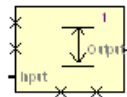
Symbol 8.78:



SCADE name	CheckBounds
Package	chk
Function	Checks that an input is strictly lower than Min and strictly higher than Max.
Inputs	Input: 'T
Hidden inputs	Max: 'T and Min: 'T where 'T numeric MaxOpen: bool MinOpen: bool
Outputs	none
KCG pragma	Operator expansion

# Check Bounds (Extended)

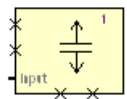
Symbol 8.79:



SCADE name	CheckBoundsEx
Package	chk
Function	Checks that an input is strictly lower than Min and strictly higher than Max and returns test results.
Inputs	Input: 'T
Hidden inputs	Max: 'T and Min: 'T where 'T numeric MaxOpen: bool MinOpen: bool
Outputs	Output: bool

# Check Gap

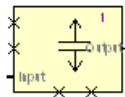
Symbol 8.80:



SCADE name	CheckGap
Package	chk
Function	Checks that an input is strictly higher than Min and strictly lower than Max.
Inputs	Input: 'T
Hidden inputs	Max: 'T and Min: 'T where 'T numeric MaxOpen: bool MinOpen: bool
Outputs	none
KCG pragma	Operator expansion

# Check Gap (Extended)

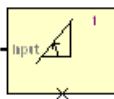
Symbol 8.81:



SCADE name	CheckGapEx
Package	chk
Function	Checks that an input is strictly higher than Min and strictly lower than Max and returns test results.
Inputs	Input: 'T
Hidden inputs	Max: 'T and Min: 'T where 'T numeric MaxOpen: bool MinOpen: bool
Outputs	Output: bool

# Check Gradient

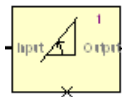
Symbol 8.82:



SCADE name	CheckGradient
Package	chk
Function	Checks that the expression $\text{Input} - \text{fby}(\text{Input};1;0)$ is strictly lower than Max.
Inputs	Input: 'T
Hidden inputs	Max: 'T where 'T numeric
Outputs	none
KCG pragma	Operator expansion

# Check Gradient (Extended)

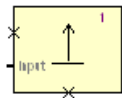
Symbol 8.83:



SCADE name	CheckGradientEx
Package	chk
Function	Checks that the expression Input - fby(Input;1;0) is strictly lower than Max and returns test results.
Inputs	Input: 'T
Hidden inputs	Max: 'T where 'T numeric
Outputs	Output: bool

# Check Lower Bound

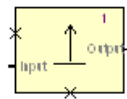
Symbol 8.84:



SCADE name	CheckLowerBound
Package	chk
Function	Checks that an input is strictly higher than Min.
Inputs	Input: 'T
Hidden inputs	Min: 'T where 'T numeric MinOpen: bool
Outputs	none
KCG pragma	Operator expansion

# Check Lower Bound (Extended)

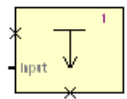
Symbol 8.85:



SCADE name	CheckLowerBoundEx
Package	chk
Function	Checks that an input is strictly higher than Min and returns test results.
Inputs	Input: 'T
Hidden inputs	Min: 'T where 'T numeric MinOpen: bool
Outputs	Output: bool

# Check Upper Bound

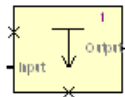
Symbol 8.86:



SCADE name	CheckUpperBound
Package	chk
Function	Checks that an input is strictly lower than Max.
Inputs	Input: 'T
Hidden inputs	Max: 'T where 'T numeric MaxOpen: boo
Outputs	none
KCG pragma	Operator expansion

# Check Upper Bound (Extended)

Symbol 8.87:



SCADE name	CheckUpperBoundEx
Package	chk
Function	Checks that an input is strictly lower than Max and eturns test results.
Inputs	Input: 'T
Hidden inputs	Max: 'T where 'T numeric MaxOpen: boo
Outputs	Output: bool

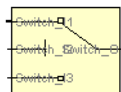
# Signal Routing Operators

Read the description of the following operators:

- [“If Then Else”](#)  
[“Ignore First Input”](#)  
[“Merge \(2 inputs\)”](#)
- [“Merge \(3 inputs\)”](#)  
[“Switch”](#)  
[“Switch \(GTT\)”](#)
- [“Switch \(NEZ\)”](#)  
[“Switch Enumeration”](#)  
[“Switch Enumeration \(GTT\)”](#)

## If Then Else

Symbol 8.88:

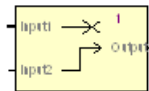


SCADE name	IfThenElse
Package	smk
Function	Switches output between the first and third input based on the criteria applied to the value of the second input when Boolean (if I2 >= threshold, then O=I1, else O=I3).
Inputs	Switch_I1: 'T Switch_I2: bool Switch_I3: 'T where 'T numeric
Outputs	Switch_O: 'T
KCG pragma	Operator expansion



# Ignore First Input

Symbol 8.89:



SCADE name	IgnoreFirst
Package	smlk
Function	Connects the first input to a terminator and returns the second input to the output.
Inputs	Input1: 'T Input2: 'T1
Outputs	Output: 'T1
KCG pragma	Operator expansion

# Merge (2 inputs)

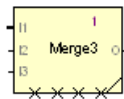
Symbol 8.90:



SCADE name	Merge2
Package	smlk
Function	Merges two inputs into one output based on the Boolean conditions specified as hidden inputs.
Inputs	I1 and I2: 'T where 'T numeric
Hidden inputs	InitVal: 'T C1 and C2: bool
Outputs	O: 'T

# Merge (3 inputs)

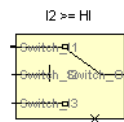
Symbol 8.91:



SCADE name	Merge3
Package	smlk
Function	Merges three inputs into one output based on the Boolean conditions specified as hidden inputs.
Inputs	I1, I2 and I3: 'T where 'T numeric
Hidden inputs	InitVal: 'T C1, C2 and C3: bool
Outputs	O: 'T

# Switch

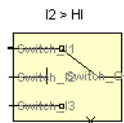
Symbol 8.92:



SCADE name	Switch
Package	smlk
Function	Switches output between the first and the third input based on criteria applied to the value of the second input (if I2 >= threshold, then O=I1, else O=I3).
Inputs	Switch_I1: 'T Switch_I2: 'T2 where 'T2 is numeric Switch_I3: 'T
Hidden inputs	Switch_HI: 'T2 (threshold value) where 'T2 is numeric
Outputs	Switch_O: 'T
KCG pragma	Operator expansion

# Switch (GTT)

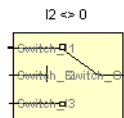
Symbol 8.93:



SCADE name	SwitchGTT
Package	smlk
Function	Switches output between the first and the third input based on criteria applied to the value of the second input (if I2 > threshold, then O=I1, else O=I3).
Inputs	Switch_I1: 'T Switch_I2: 'T2 where 'T2 is numeric Switch_I3: 'T
Hidden inputs	Switch_HI: 'T2 (threshold value) where 'T2 is numeric
Outputs	Switch_O: 'T
KCG pragma	Operator expansion

# Switch (NEZ)

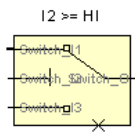
Symbol 8.94:



SCADE name	SwitchNEZ
Package	smlk
Function	Switches output between the first and the third input based on criteria applied to the value of the second input (if I2 not equal to 0, then O=I1, else O=I3).
Inputs	Switch_I1: 'T Switch_I2: 'T2 where 'T2 is numeric Switch_I3: 'T
Outputs	Switch_O: 'T
KCG pragma	Operator expansion

# Switch Enumeration

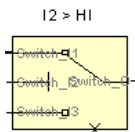
Symbol 8.95:



SCADE name	SwitchEnum
Package	smlk
Function	Switches output between the first and the third input based on criteria applied to the value of the second input (if I2 >= threshold, then O=I1, else O=I3).
Inputs	Switch_I1: 'T, Switch_I2: 'T2, and Switch_I3: 'T
Hidden inputs	Switch_HI: 'T2 (threshold value)
Outputs	Switch_O: 'T
KCG pragma	Operator expansion

# Switch Enumeration (GTT)

Symbol 8.96:



SCADE name	SwitchGTEnum
Package	smlk
Function	Switches output between the first and the third input based on criteria applied to the value of the second input (if I2 > threshold, then O=I1, else O=I3).
Inputs	Switch_I1: 'T, Switch_I2: 'T2, and Switch_I3: 'T
Hidden inputs	Switch_HI: 'T2 (threshold value)
Outputs	Switch_O: 'T
KCG pragma	Operator expansion

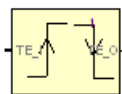
# Subsystem Operators

Read the description of the following operators:

- ["Trigger Either Edge"](#)
- ["Trigger Either Edge \(Extended\)"](#)
- ["Trigger Falling Edge"](#)
- ["Trigger Falling Edge \(Extended\)"](#)
- ["Trigger Rising Edge"](#)
- ["Trigger Rising Edge \(Extended\)"](#)

## Trigger Either Edge

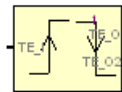
Symbol 8.97:



SCADE name	TriggerEither
Package	smlk
Function	Detects rising or falling edge (compared to zero) of an input.
Inputs	TE_I: 'T where 'T numeric
Outputs	TE_O: bool
KCG pragma	Operator expansion

# Trigger Either Edge (Extended)

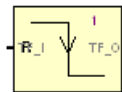
Symbol 8.98:



SCADE name	TriggerEitherEx
Package	smlk
Function	Detects rising or falling edge (compared to zero) of an input.
Inputs	TE_I: 'T where 'T numeric
Outputs	TE_O: bool and TE_O2: int8
Comment	An additional output indicates the kind of edge that is activated (-1 for falling, 1 for rising, and 0 for no edge).

# Trigger Falling Edge

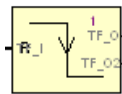
Symbol 8.99:



SCADE name	TriggerFall
Package	smlk
Function	Detects falling edge (compared to zero) of an integer input.
Inputs	TF_I: 'T where 'T numeric
Outputs	TF_O: bool
KCG pragma	Operator expansion

# Trigger Falling Edge (Extended)

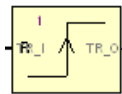
Symbol 8.100:



SCADE name	TriggerFallEx
Package	smlk
Function	Detects falling edge (compared to zero) of a real/float input.
Inputs	TF_I: 'T where 'T numeric
Outputs	TF_O: bool TF_O2: int8
Comment	An additional output indicates if the edge is activated (-1 for falling or 0 for no edge).

# Trigger Rising Edge

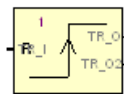
Symbol 8.101:



SCADE name	TriggerRise
Package	smlk
Function	Detects rising edge (compared to zero) of an integer input.
Inputs	TR_I: 'T where 'T numeric
Outputs	TR_O: bool
KCG pragma	Operator expansion

# Trigger Rising Edge (Extended)

Symbol 8.102:



SCADE name	TriggerRiseEx
Package	smlk
Function	Detects rising edge (compared to zero) of a real/float input.
Inputs	TR_I: 'T where 'T numeric
Outputs	TR_O: bool TR_O2: int8
Comment	An additional output indicates if the edge is activated (1 for rising or 0 for no edge).



# Signal Generator Operators

Read the description of the following operators:

["Clock Generator"](#)

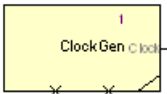
["Ground \(Boolean\)"](#)

["Ground \(int/int32\)"](#)

["Ground \(real/float\)"](#)

## Clock Generator

Symbol 8.103:



SCADE name	ClockGen
Package	smlk
Function	Generates a Boolean clock signal based on two hidden integer inputs.
Inputs	None
Hidden inputs	Period: int32 and Offset: int32
Outputs	Clock: bool

## Ground (Boolean)

Symbol 8.104:



SCADE name	GroundB
Package	smlk
Function	Generates a Boolean output set as <code>false</code> .
Inputs	None
Outputs	GB_Out: bool
KCG pragma	Operator expansion

# Ground (int/int32)

Symbol 8.105:



SCADE name	GroundI
Package	smlk
Function	Generates an integer output set to 0.
Inputs	None
Outputs	GI_Out: int32
KCG pragma	Operator expansion

# Ground (real/float)

Symbol 8.106:



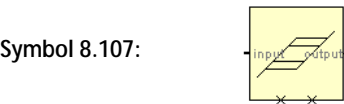
SCADE name	GroundR
Package	smlk
Function	Generates a real/float64 output set to 0.0.
Inputs	None
Outputs	GR_Out: float64
KCG pragma	Operator expansion

# Discontinuity Operators

Read the description of the following operators:

- ["Backlash"](#)  
["Dead Zone Dynamic"](#)  
["Rate Limiter"](#)  
["Rate Limiter Dynamic"](#)
- ["Relay"](#)  
["Saturate Dynamic"](#)  
["Wrap To Zero"](#)

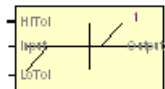
## Backlash



SCADE name	Backlash
Package	smlk
Function	Returns a change in output equal to a change in input with a deadband in direction changes.
Inputs	input: 'T where 'T is float
Hidden inputs	deadband_width: 'T and initial_output: 'T where 'T is float
Outputs	output: 'T where 'T is float

# Dead Zone Dynamic

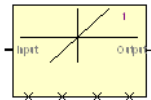
Symbol 8.108:



SCADE name	DeadZoneDynamic
Package	smlk
Function	Keeps the output equal to 0.0 as long as the input is within the [lower tolerance, upper tolerance] range. When the input reaches beyond one limit, the output is equal to the input minus the limit.
Inputs	HiTol: 'T, Input: 'T, and LoTol: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion

# Rate Limiter

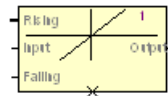
Symbol 8.109:



SCADE name	RateLimiter
Package	smlk
Function	Limits the first derivative of the input signal. The output does not change faster than the specified limit.
Inputs	input: 'T where 'T is float
Hidden inputs	RisingSlewLimit: 'T, FallingSlewLimit: 'T, TimeStep: 'T, and InitialOutput: 'T where 'T is float
Outputs	output: 'T where 'T is float

# Rate Limiter Dynamic

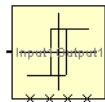
Symbol 8.110:



SCADE name	RateLimiterDynamic
Package	smlk
Function	Limits the first derivative of the signal passing through. The output does not change faster than the specified limit.
Inputs	Rising: 'T, Input: 'T, and Falling: 'T where 'T is float
Hidden inputs	deltaT: 'T where 'T is float
Outputs	Output: 'T where 'T is float
KCG pragma	Operator expansion
Comment	Let us call Rate the derivative $((RL\_Input - pre(RL\_Output)) / \Delta T$ . If $Rate > Rising$ , then $RL\_Output$ equals $(Rising * \Delta T + pre(RL\_Output))$ If $Rate < Falling$ , then $RL\_Output$ equals $(Falling * \Delta T + pre(RL\_Output))$ Otherwise, $RL\_Output = RL\_Input$ .

# Relay

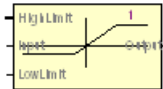
Symbol 8.111:



SCADE name	Relay
Package	smlk
Function	Basic relay that generates discontinuous output values based on input.
Inputs	Input1: 'T where 'T numeric
Hidden inputs	SwitchOn: 'T and SwitchOff: 'T OutputOn: 'T and OutputOff: 'T
Outputs	Output1: 'T

# Saturate Dynamic

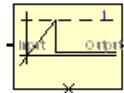
Symbol 8.112:



SCADE name	SaturateDynamic
Package	smlk
Function	If the input is lower than a lower limit, the output is the lower limit; if the input is greater than a higher limit, the output is the higher limit; otherwise the output is equal to the input.
Inputs	HighLimit: 'T, Input: 'T, and LowLimit: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	If Input is lower than LowLimit, Output is equal to LowLimit; if Input is greater than HighLimit, Output is equal to HighLimit; otherwise Output is equal to Input.

# Wrap To Zero

Symbol 8.113:



SCADE name	WrapToZero
Package	smlk
Function	Sets the output to zero if the input is above threshold.
Inputs	Input: 'T
Hidden inputs	Threshold: 'T
Outputs	Output: 'T where 'T numeric
KCG pragma	Operator expansion

# LUT, Pre-LUT and Interpolation Operators

Read the description of the following operators:

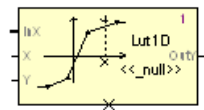
- [“LUT 1D”](#)
- [“Pre-LUT2”](#)
- [“Pre-LUT2 \(Direct\)”](#)
- [“Interpolation1D2”](#)
- [“Interpolation1D \(Floor\)”](#)
- [“Interpolation2D2”](#)
- [“Interpolation2D \(Floor\)”](#)

## Note

The Pre-Look-up and Interpolation operators from the `Lut` package use a structured type, `LutIndex`. These operators map directly to Simulink blocks until release R2006a. From release R2006b, these Simulink operators output the index and factor in two outputs instead of a structure. The following operators are used in SCADE Suite Gateway for Simulink to map directly Simulink blocks.

## LUT 1D

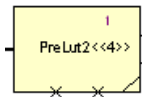
Symbol 8.114:



SCADE name	Lut1D
Package	smlk
Size parameters	Xs
Inputs	InX: 'T where 'T is float x: 'T^Xs and y: 'T^Xs
Hidden inputs	Extrapol: bool
Outputs	OutY: 'T where 'T is float
KCG pragma	Operator expansion

# Pre-LUT2

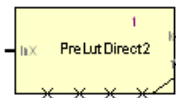
Symbol 8.115:



SCADE name	PreLut2
Package	smk
Function	See introductory note above
Size parameters	Xs
Inputs	InX: 'T where 'T float
Hidden inputs	x: 'T ^Xs where 'T float Extrapol: bool
Outputs	k: int32 f: 'T where 'T is float

# Pre-LUT2 (Direct)

Symbol 8.116:

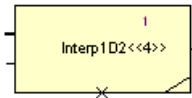


SCADE name	PreLutDirect2
Package	smk
Function	See introductory note above
Inputs	InX: 'T where 'T is float
Hidden inputs	Orig: 'T Step: 'T NbElem: int32 Extrapol: bool where 'T is float
Outputs	k: int32 f: 'T where 'T is float



# Interpolation1D2

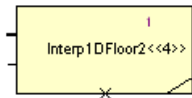
Symbol 8.117:



SCADE name	Interp1D2
Package	smlk
Function	See introductory note above
Size parameters	Xs
Inputs	k: int32 f: 'T where 'T is float
Hidden inputs	Y: 'T ^Xs where 'T is float
Outputs	Out Y: 'T where 'T is float

# Interpolation1D (Floor)

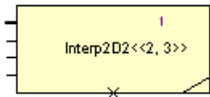
Symbol 8.118:



SCADE name	Interp1DFloor2
Package	smlk
Function	See introductory note above
Size parameters	Xs
Inputs	k: int32 f: 'T where 'T is float
Hidden inputs	Y: 'T ^Xs where 'T is float
Outputs	Out Y: 'T where 'T is float

# Interpolation2D2

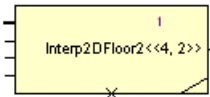
Symbol 8.119:



SCADE name	Interp2D2
Package	smlk
Function	See introductory note above
Size parameters	Xs and Ys
Inputs	k1: int32, f1: 'F, k2: int32, and f2: 'F where 'F is float
Hidden inputs	z: 'F^Ys^Xs where 'F is float
Outputs	Out z: 'F where 'F is float

# Interpolation2D (Floor)

Symbol 8.120:



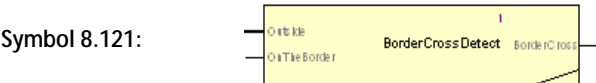
SCADE name	Interp2DFloor2
Package	smlk
Function	See introductory note above
Size parameters	Xs and Ys
Inputs	k1: int32, f1: 'T, k2: int32, and f2: 'T where 'T is float
Hidden inputs	z: 'T ^Ys^Xs where 'T is float
Outputs	Out z: 'T where 'T is float

# Miscellaneous Operators

Read the description of the following operators:

- ["Border Crossing Detection"](#)  
["Counter \(limited\)"](#)  
["Entry Detection"](#)  
["Identity"](#)
- ["Inactive Cycles"](#)  
["Inactive Time"](#)  
["Quantizer"](#)  
["Unit Delay \(Helper\)"](#)
- ["Width \(Matrix\)"](#)  
["Width \(Scalar\)"](#)  
["Width \(Vector\)"](#)

## Border Crossing Detection



SCADE name	BorderCrossDetect
Package	smlk
Function	Detects border crossing based on two Boolean inputs: one input indicates whether the signal is outside the border and the other one whether the signal is at the border.
Inputs	Outside: bool and OnTheBorder: bool
Outputs	BorderCross: bool

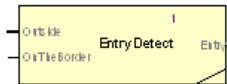
## Counter (limited)



SCADE name	CounterLimited
Package	smlk
Function	Counts from zero to a specified limit, then wraps.
Hidden inputs	Limit: int32
Outputs	Output: int32

# Entry Detection

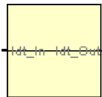
Symbol 8.123:



SCADE name	EntryDetect
Package	smlk
Function	Detects entry into a range based on two Boolean inputs: one input indicates whether the signal is outside the border and the other one whether the signal is at the border.
Inputs	Outside: bool and OnTheBorder: bool
Outputs	BorderCross: bool

# Identity

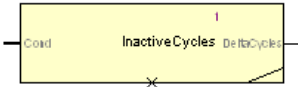
Symbol 8.124:



SCADE name	Idt
Package	smlk
Function	Passes signal through (Output=Input)
Inputs	Idt_In: 'T where 'T is numeric
Outputs	Idt_Out: 'T
KCG pragma	Operator expansion

# Inactive Cycles

Symbol 8.125:



SCADE name	InactiveCycles
Package	smlk
Function	Counter that increments an integer output in steps equal to a hidden integer value when Boolean input is <code>false</code> , else it returns a constant equal to its hidden value.
Inputs	Cond: bool
Hidden inputs	NbCycles: 'T where 'T is integer
Outputs	DeltaCycles: 'T where 'T is integer

# Inactive Time

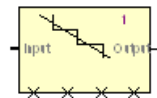
Symbol 8.126:



SCADE name	InactiveTime
Package	smlk
Function	Counter that increments a real /float output in steps equal to the hidden value when the Boolean input is <code>false</code> , else it returns a constant equal to its hidden value.
Inputs	Cond: bool
Hidden inputs	DTInit: 'T where 'T is numeric
Outputs	DeltaT: 'T where 'T is numeric

# Quantizer

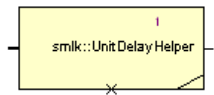
Symbol 8.127:



SCADE name	Quantizer
Package	smlk
Function	Basic quantizer based on specified interval. Discretizes the input at a specified interval and produces a stair-step output.
Inputs	Input: 'T where 'T is float
Hidden inputs	Bits: int32, Umin and Umax: 'T where 'T is float OutputNegativeValues: bool
Outputs	Output: int32

# Unit Delay (Helper)

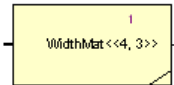
Symbol 8.128:



SCADE name	UnitDelayHelper
Package	smlk
Function	Performs a unit delay based on init value.
Inputs	Input: 'T
Hidden inputs	x0: 'T
Outputs	Output: 'T
KCG pragma	Operator expansion

# Width (Matrix)

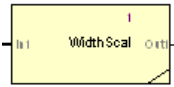
Symbol 8.129:



SCADE name	WidthMat
Package	smlk
Function	Outputs the number of elements from the input matrix.
Size parameters	N1 and N2
Inputs	In1: 'T ^N1^N2 where 'T numeric
Outputs	Out1: int32
KCG pragma	Operator expansion
Comment	Simulink block Width mapped to operators WidthScal, WidthVect, or WidthMat depending on the kind of block input.

# Width (Scalar)

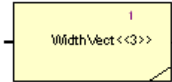
Symbol 8.130:



SCADE name	WidthScal
Package	smlk
Function	Outputs the number of elements from the input scalar, i.e. value 1.
Inputs	In1: 'T where 'T numeric
Outputs	Out1: int32
KCG pragma	Operator expansion
Comment	Simulink block Width mapped to operators WidthScal, WidthVect, or WidthMat depending on the kind of block input.

# Width (Vector)

Symbol 8.131:



SCADE name	WidthVect
Package	smlk
Function	Outputs the number of elements from the input vector.
Size parameters	N
Inputs	In1: 'T ^N where 'T numeric
Outputs	Out1: int32
KCG pragma	Operator expansion
Comment	Simulink block Width mapped to operators WidthScal, WidthVect, or WidthMat depending on the kind of block input.



# 9 / Library libverif

This library contains operators designed to help expressing verification properties. Such operators are expressed as regular SCADE nodes. The timing diagram of each operator is provided when it applies.

Access the description of library operators in alphabetical order:

- [“After Nth Tick”](#)
- [“Always After First Condition”](#)
- [“At Least N Ticks”](#)
- [“Has Never Been True”](#)
- [“Implies”](#)
- [“Implies Within N Ticks”](#)

## After Nth Tick

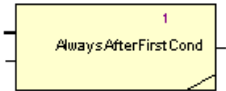
Symbol 9.1:



SCADE name	AfterNthTick						
Function	Used to express that the output equals the input, except for the first N cycles during which the output is <code>true</code> ( <i>i.e.</i> , input on the first N cycles is ignored).						
Size parameters	N specifies the number of cycles						
Inputs	Input1: bool						
Outputs	Output1: bool						
Comment	Check the operator timing diagram below.						
Timing Diagram	Cycle	1	2	...	10	11	12
	Output1 (N=10 ticks)	true	true	true	true	Input1	Input1

# Always After First Condition

Symbol 9.2:

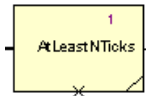


SCADE name	AlwaysAfterFirstCond
Function	Used to express that the output equals the input once a condition is true. Before that and since the initial cycle, the output is true.
Inputs	Input1: bool and Cond: bool
Outputs	Output1: bool
Comment	Check the operator timing diagram below.

Timing Diagram	Cycle	1	2	3	4	5	6
	Cond	false	false	true	*	*	*
	Output1	true	true	Input1	Input1	Input1	Input1

# At Least N Ticks

Symbol 9.3:

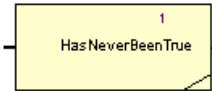


SCADE name	AtLeastNTicks
Function	Used to express that the output equals the input as soon as the input is <code>true</code> for N cycles. Before this N <sup>th</sup> cycle, the output is <code>false</code> .
Inputs	Input1: bool
Hidden inputs	N: int32
Outputs	Output1: bool
Comment	Check the operator timing diagram below.

Timing Diagram	Cycle (N)	1	2	...	5	6	...
	Input1	true	true	true	true	*	*
	Output1	false	false	false	Input1	Input1	Input1

# Has Never Been True

Symbol 9.4:

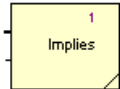


SCADE name	HasNeverBeenTrue
Function	Used to express that the output becomes <code>false</code> as soon as its input becomes <code>true</code> for the first time. After this cycle, the output remains <code>false</code> .
Inputs	Input1: bool
Outputs	Output1: bool
KCG pragma	Operator expansion
Comment	Check the operator timing diagram below.

Timing Diagram	Cycle	1	2	3	4	5	...
	Input1	false	false	true	true	*	*
	Output1	true	true	false	false	*	*

# Implies

Symbol 9.5:

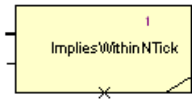


SCADE name	<b>Implies</b>
Function	Implements the "implies" logical operator. "If A is <code>true</code> , then B is <code>true</code> ": <code>Not(A) or B</code> .
Inputs	A: bool and B: bool
Outputs	C: bool
KCG pragma	Operator expansion
Comment	Check the operator timing diagram below.

Truth Table	A	B	C
	false	false	true
	false	true	true
	true	false	false
	true	true	true

# Implies Within N Ticks

Symbol 9.6:



SCADE name	ImpliesWithinNTick
Function	Output equal to "Input1 implies Input2" as soon as Input1 is true for "N" ticks.
Inputs	Input1: bool and Input2: bool
Hidden inputs	N: int32
Outputs	Output1: bool

Timing Diagram	Cycle (N)	1	2	3	4	5	6	...
	N=3	*	*	*	*	*		*
	Input1	false	false	true	true	true	true	false
	Input2	*	*	*	*	false	true	*
	Output1	true	true	true	true	false	true	true

# Index

---

## A

### Absolute value

libmath 73

### Advanced arithmetic (libraries)

See list 103

integer power for integers 103

integer power for reals 104

inverse 104

real power for reals 105

square 105

### AfterNthTick

libverif 213

### AlwaysAfterFirstCond

libverif 214

### AND (numeric)

libsmk operator 171

### Arithmetic (libraries)

See list 73

absolute value 73

maximum (2 inputs) 74

maximum (3 inputs) 74

mean (2 inputs) 75

mean (3 inputs) 75

minimum (2 inputs) 76

minimum (3 inputs) 76

modulo 77

polynomial 77

sign 78

### Arithmetic from Simulink (libraries)

See list 141

bias 141

inverse (int) 142

maximum (reset) 142

minimum (reset) 143

modulo (int) 143

modulo (real) 144

polynomial 144

remainder (real) 145

sign (int) 145

### Assertion

libsmk operator 181

### Assertion (inv)

libsmk operator 182

### Asymmetrical dead band

libpwnlinear 114

### Asymmetrical limiter

libpwnlinear 116

### Asymmetrical pre-load

libpwnlinear 118

### AtLeastNTicks

libverif 215

## B

### Backlash

liblinear operator 59

libsmk operator 199

### Backward integrator

liblinear operator 66

### Basic memory

liblinear operator 70

### Bias

libsmk operator 141

### Bit clear

libsmk operator 153

### Bit set

libsmk operator 154

### BitClear

libimpl operator 40

### BitSet

libimpl operator 41

### Bitshift on sized integer (libraries)

See list 40

decrease LSB 41

increase LSB to ceiling 42

increase LSB to fix 43

increase LSB to floor 43

shift left 44

shift right 44

### Bitwise AND

libsmk operator 154

### Bitwise logic from Simulink (libraries)

See list 153

AND 154

bit clear 153

bit set 154

extract bit range 159

Nand 155

Nor 155

NOT 156

OR 156

shift 157

shift left 158

shift right 158

XOR 157

### Bitwise logic on sized integer (libraries)

See list 40

### Bitwise Nand

# Index

---

- libsmkl operator 155
- Bitwise Nor
  - libsmkl operator 155
- Bitwise NOT
  - libsmkl operator 156
- Bitwise OR
  - libsmkl operator 156
- Bitwise shift
  - libsmkl operator 157
- Bitwise shift left
  - libsmkl operator 158
- Bitwise shift right
  - libsmkl operator 158
- Bitwise XOR
  - libsmkl operator 157
- Boolean to Boolean
  - libsmkl operator 160
- Border crossing detection
  - libsmkl operator 207
- BuildArray
  - libmath operator 91

## C

- Cartesian to polar
  - libmathext operator 106
  - libsmkl operator 161
- Cartesian to spherical
  - libmathext operator 107
  - libsmkl operator 161
- Celsius to Fahrenheit
  - libmathext operator 107
  - libsmkl operator 162
- Check bounds

- libsmkl operator 182
- Check gap
  - libsmkl operator 183
- Check gradient
  - libsmkl operator 184
- Check lower bound
  - libsmkl operator 185
- Check slope
  - libpwnlinear 112
- Check upper bound
  - libsmkl operator 186
- Clock
  - libsmkl operator 168
- Clock counter
  - libpwnlinear 113
- Clock generator
  - libsmkl operator 197
- Compare (enumeration)
  - libsmkl operator 173
- Compare (numeric)
  - libsmkl operator 172
- Comparison operators
  - in libimpl 36
- Conversion (libraries)
  - See list 79
  - cartesian to polar 106
  - cartesian to spherical 107
  - Celsius to Fahrenheit 107
  - degrees to radians 108
  - Fahrenheit to Celsius 108
  - integer to Boolean 84
  - polar to cartesian 109
  - radians to degrees 109

- real to Boolean 85
- rounding function 86
- rounding to ceil function 86
- rounding to floor function 87
- spherical to cartesian 110
- Conversion from Simulink (libraries)
  - See list 160
  - Boolean to Boolean 160
  - Celsius to Fahrenheit 162
  - degrees to radians 162
  - Fahrenheit to Celsius 163
  - integer to Boolean 164
  - radians to degrees 165
  - real to Boolean 166
  - scalar to Boolean 166
- Count-down
  - libdigital operator 12
- Counter
  - libpwnlinear 113
- Counter (limited)
  - libsmkl operator 207

## D

- Dead zone dynamic
  - libsmkl operator 200
- Decrease LSB
  - libimpl operator 41
- Decrement time to zero (int)
  - libsmkl operator 146
- Decrement time to zero (real)
  - libsmkl operator 147
- Decrement to zero
  - libsmkl operator 147



# Index

---

- Degrees to radians
  - libmathext operator 108
  - libsmk operator 162
- Derivative
  - liblinear operator 60
- Detect (change)
  - liblinear operator 60
- Detect (decrease)
  - liblinear operator 61
- Detect (fall negative)
  - liblinear operator 61
- Detect (fall non positive)
  - liblinear operator 62
- Detect (increase)
  - liblinear operator 62
- Detect (rise non negative)
  - liblinear operator 63
- Detect (rise positive)
  - liblinear operator 63
- Detect change (numeric)
  - libsmk operator 174
- Detect decrease (numeric)
  - libsmk operator 174
- Detect falling edge (to negative)
  - libsmk operator 175
- Detect falling edge (to non-positive)
  - libsmk operator 175
- Detect increase (numeric)
  - libsmk operator 176
- Detect rising edge
  - libsmk operator 176
- Detect rising edge (to positive)
  - libsmk operator 177
- Different (integers)
  - libimpl operator 38
- Digital operators (libraries)
  - See list 8
  - count-down 12
  - either edge 13
  - falling edge 14
  - falling edge (no re-trigger) 15
  - falling edge (re-trigger) 14
  - flip-flop (reset priority) 17
  - flip-flop (set priority) 18
  - flip-flop JK 16
  - inactive cycles 19
  - integer to Boolean vector 20
  - rising edge 21
  - rising edge (no re-trigger) 22
  - rising edge (re-trigger) 22
  - toggle 23
  - trigger (either) 23
  - trigger (fall) 24
  - trigger (rise) 24
- Discontinuity from Simulink (libraries)
  - See list 199
  - backlash 199
  - dead zone dynamic 200
  - rate limiter 200
  - rate limiter dynamic 201
  - relay 201
  - saturate dynamic 202
  - wrap to zero 202
- Discrete filter
  - libsmk operator 135
- Discrete filter (normalized)
  - libsmk operator 135
- Discrete-time integrator (backward)
  - libsmk operator 136
- Discrete-time integrator (forward)
  - libsmk operator 137
- Discrete-time integrator (trapeze)
  - libsmk operator 138
- Discrete-time integrators (libraries)
  - See list 136
  - transfer function (first order) 139
- DLatch
  - libsmk operator 168
- E**
- Either edge
  - libdigital operator 13
- Entry detection
  - libsmk operator 208
- Enumeration to Integer
  - libsmk operatorConversion from Simulink (libraries) enumeration to integer 163
- Equal (integers)
  - libimpl operator 37
- Extract bit range
  - libsmk operator 159
- ExtractBitRange

# Index

---

libimpl operator 42

## F

Fahrenheit to Celsius

libmathext operator 108

libsmk operator 163

Falling edge

libdigital operator 14

Falling edge (no re-trigger)

libdigital operator 15

Falling edge (re-trigger)

libdigital operator 14

Falling hysteresis

libpwnlinear 115

Filter (first order loop)

liblinear operator 64

Filter functions (libraries)

See list 46

Flip-flop (reset priority)

libdigital operator 17

Flip-flop (set priority)

libdigital operator 18

Flip-flop D

libsmk operator 169

Flip-flop from Simulink  
(libraries)

See list 168

clock 168

DLatch 168

flip-flop D 169

flip-flop JK 169

flip-flop SR 170

Flip-flop JK

libdigital operator 16

libsmk operator 169

Flip-flop SR

libsmk operator 170

Forward integrator

liblinear operator 67

## G

Gain

liblinear operator 64

Greater than or equal  
(integers)

libimpl operator 39

Ground (bool)

libsmk operator 197

Ground (int)

libsmk operator 198

Ground (real)

libsmk operator 198

## H

HasNeverBeenTrue

libverif 216

Hit crossing (either direction)

liblinear operator 65

Hit crossing (falling direction)

liblinear operator 65

Hit crossing (rising direction)

liblinear operator 66

## I

Identity

libsmk operator 208

if then else

libsmk operator 188

Ignore first input

libsmk operator 189

Implies

libverif 217

ImpliesWithinNTick

libverif 218

In range (In-In)

libmath 88

libsmk operator 177

In range (In-Out)

libmath 89

libsmk operator 178

In range (Out-In)

libmath 89

libsmk operator 178

In range (Out-Out)

libmath 90

libsmk operator 179

Inactive cycles

libsmk operator 209

Inactive time

libsmk operator 209

InactiveCycles

libdigital operator 19

Increase LSB to ceiling

libimpl operator 42

Increase LSB to fix

libimpl operator 43

Increase LSB to floor

libimpl operator 43

Integer division (ceiling  
rounding)

libimpl operator 30

# Index

---

Integer division (fix rounding)  
    libimpl operator 30  
Integer division (floor rounding)  
    libimpl operator 31  
Integer division (saturated)  
    libimpl operator 31  
Integer multiplication  
    libimpl operator 32  
Integer multiplication (int16)  
    libimpl operator 32  
Integer multiplication (int32)  
    libimpl operator 33  
Integer multiplication (saturated)  
    libimpl operator 34  
Integer multiplication (uint16)  
    libimpl operator 33  
Integer multiplication (uint32)  
    libimpl operator 33  
Integer negation (saturated)  
    libimpl operator 34  
Integer power (for integers)  
    libmathext 103  
Integer power (for reals)  
    libmathext 104  
Integer subtraction (saturated)  
    libimpl operator 35  
Integer sum (saturated)  
    libimpl operator 35  
Integer to Boolean  
    libmath 84  
    libsmkl operator 164

Integer to Boolean vector  
    libdigital operator 20  
Integer to enumeration  
    libsmkl operatorConversion  
    from Simulink (libraries)  
        integer to enumeration 164  
Interpolation 1D  
    libpwllinear 120  
Interpolation 1D (floor)  
    libpwllinear 121  
Interpolation 2D  
    libpwllinear 121  
Interpolation 2D (floor)  
    libpwllinear 122  
Interpolation from Simulink (libraries)  
    See list 203  
    interpolation1D (floor) 205  
    interpolation1D2 205  
    interpolation2D (Floor) 206  
    interpolation2D2 206  
Interpolation1D (floor)  
    libsmkl operator 205  
Interpolation1D2  
    libsmkl operator 205  
Interpolation2D (Floor)  
    libsmkl operator 206  
Interpolation2D2  
    libsmkl operator 206  
Interval calculus (libraries)  
    See list 88  
    In range (In-In) 88  
    In range (In-Out) 89  
    In range (Out-In) 89

    In range (Out-Out) 90  
Inverse  
    libmathext 104  
Inverse (int)  
    libsmkl operator 142

## L

Less than or equal (integers)  
    libimpl operator 37  
libdigital  
    See list of operators 7  
    digital operators 8  
    truth table operators 25  
libimpl (sied integer)  
    saturation operators 28  
libimpl (sized integer)  
    See list of operators 27  
    arithmetic operators 29  
    bitshift operators 40  
    bitwise logic operators 40  
    comparison operators 36  
liblinear  
    See list of operators 45  
    discrete filters 46  
    linear functions 59  
    transfer functions 46  
libmath  
    See list of operators 71  
    arithmetic operators 73  
    conversion operators 79  
    interval operators 88  
    matrix and vector operators 91  
libmathext  
    See list of operators 101

# Index

---

- advanced arithmetic operators 103
- power function operators 102
- trigonometric function operators 102
- libpwnlinear**
  - See list of operators 111
  - LUT operators 120
  - piecewise linear functions 112
- Libraries**
  - libdigital 7
  - libimpl 27
  - liblinear 45
  - libmath 71
  - libmathext 101
  - libpwnlinear 111
  - libsmk 133
  - libverif 213
- Library operators**
  - advanced arithmetic 103
  - arithmetic from Simulink 141
  - basic arithmetic 73
  - basic vector/matrix calculus 91
  - bitshift (sized integer) 40
  - bitwise logic (Simulink) 153
  - bitwise logic (sized integer) 40
  - conversion 79
  - conversion (Simulink) 160
  - digital operators 8
  - discontinuity from Simulink 199
  - discrete filters 46
  - flip-flop from Simulink 168
  - integer comparison 36
  - interpolation from Simulink 203
  - interval calculus 88
  - linear functions 59
  - logical operation from Simulink 171
  - LUT (piecewise) 120
  - LUT and Pre-LUT from Simulink 203
  - miscellaneous from Simulink 207
  - model verification from Simulink 181
  - piecewise linear functions 112
  - power functions 102
  - signal generator from Simulink 197
  - signal routing from Simulink 188
  - Simulink discrete-time integrators 136
  - sized integer arithmetic 29
  - sized integer comparison 36
  - subsystem from Simulink 193
  - time operators (Simulink) 146
  - transfer functions 46
  - transformation (Simulink) 160
  - trigonometric functions 102
  - truth table operators 25
- libsmk**
  - See list of operators 133
  - arithmetic operators 141
  - bitwise logical operators 153
  - conversion operators 160
- discontinuity operators 199
- discrete-time integrators 136
- flip-flop operators 168
- interpolation operators 203
- logical operators 171
- LUT and Pre-LUT operators 203
- miscellaneous operators 207
- model verification operators 181
- signal generator operators 197
- signal routing operators 188
- subsystem operators 193
- time operators 146
- transformation operators 160
- libverif**
  - See list of operators 213
- Linear functions (libraries)**
  - See list 59
  - backlash 59
  - backward integrator 66
  - basic memory 70
  - derivative 60
  - detect (change) 60
  - detect (decrease) 61
  - detect (fall negative) 61
  - detect (fall non positive) 62
  - detect (increase) 62
  - detect (rise non negative) 63
  - detect (rise positive) 63
  - filter (first order loop) 64
  - forward integrator 67
  - gain 64
  - hit crossing (either direction) 65

# Index

---

- hit crossing (falling direction) 65
  - hit crossing (rising direction) 66
  - mean cycle (2 values) 69
  - mean cycle (3 values) 69
  - memory 70
  - trapeze integrator 68
  - Logical operations from Simulink (libraries)
    - See list 171
    - AND (numeric) 171
    - compare 172
    - compare enumerated 173
    - detect change 174
    - detect decrease 174
    - detect falling edge (to negative) 175
    - detect falling edge (to non-positive) 175
    - detect increase 176
    - detect rising edge (to non-negative) 176
    - detect rising edge (to positive) 177
    - in range (In-In) 177
    - in range (In-Out) 178
    - in range (Out-In) 178
    - in range (Out-Out) 179
    - NOT 179
    - OR 180
  - LUT 1D
    - libpwnlinear 122
    - libsmk operator 203
  - LUT 1D (above)
    - libpwnlinear 123
  - LUT 1D (below)
    - libpwnlinear 124
  - LUT 1D (nearest)
    - libpwnlinear 125
  - LUT 2D
    - libpwnlinear 126
  - LUT 2D (above)
    - libpwnlinear 127
  - LUT 2D (below)
    - libpwnlinear 128
  - LUT 2D (nearest)
    - libpwnlinear 129
  - LUT 3D
    - libpwnlinear 130
  - LUT 3D (nearest)
    - libpwnlinear 131
  - LUT and Pre-LUT from Simulink (libraries)
    - See list 203
    - LUT 1D 203
    - Pre-LUT2 204
    - Pre-LUT2 (direct) 204
  - LUT operations (libraries)
    - See list 120
    - interpolation 1D 120
    - interpolation 1D (floor) 121
    - interpolation 2D 121
    - interpolation 2D (floor) 122
    - LUT 1D 122
    - LUT 1D (above) 123
    - LUT 1D (below) 124
    - LUT 1D (nearest) 125
    - LUT 2D 126
    - LUT 2D (above) 127
    - LUT 2D (below) 128
    - LUT 2D (nearest) 129
    - LUT 3D 130
    - LUT 3D (nearest) 131
    - Pre-LUT 132
    - Pre-LUT (direct) 132
- ## M
- Matrix product
    - libmath 95, 96
  - Matrix product by vector
    - libmath 96
  - Max reset
    - libpwnlinear 117
  - Maximum (2 inputs)
    - libmath 74
  - Maximum (3 inputs)
    - libmath 74
  - Maximum (reset)
    - libsmk operator 142
  - Mean (2 inputs)
    - libmath 75
  - Mean (3 inputs)
    - libmath 75
  - Mean cycle (2 values)
    - liblinear operator 69
  - Mean cycle (3 values)
    - liblinear operator 69
  - Memory
    - liblinear operator 70
  - Merge (2 inputs)
    - libsmk operator 189
  - Merge (3 inputs)
    - libsmk operator 190

# Index

---

## Minimum (2 inputs)

libmath 76

## Minimum (3 inputs)

libmath 76

## Minimum (reset)

libsmk operator 143

## Model verification from Simulink (libraries)

See list 181

assertion 181

assertion (inv) 182

check bounds 182

check gap 183

check gradient 184

check lower bound 185

check upper bound 186

## Modulo

libmath 77

libsmk operator 143

## Modulo (real)

libsmk operator 144

## N

## NOT (numeric)

libsmk operator 179

## O

## OR (numeric)

libsmk operator 180

## Overflow check

decrease LSB operator 41

## Overflow detection

in libimpl 28

## P

## Piecewise linear functions (libraries)

See list 112

asymmetrical dead band 114

asymmetrical limiter 116

asymmetrical pre-load 118

check slope 112

clock counter 113

counter 113

falling hysteresis 115

max reset 117

quantizer 119

rate limiter 119

rising hysteresis 115

symmetrical dead band 114

symmetrical limiter 116

symmetrical pre-load 118

## Polar to cartesian

libmathext operator 109

libsmk operator 165

## Polynomial

libmath 77

libsmk operator 144

## Power functions (libraries)

See list 102

## Pre-LUT

libpwnlinear 132

## Pre-LUT (direct)

libpwnlinear 132

## Pre-LUT2

libsmk operator 204

## Pre-LUT2 (direct)

libsmk operator 204

## Q

## Quantizer

libpwnlinear 119

libsmk operator 210

## R

## Radians to degrees

libmathext operator 109

libsmk operator 165

## Rate limiter

libpwnlinear 119

libsmk operator 200

## Rate limiter dynamic

libsmk operator 201

## Real power (for reals)

libmathext 105

## Real to Boolean

libmath 85

libsmk operator 166

## Relay

libsmk operator 201

## Remainder (real)

libsmk operator 145

## Rising edge

libdigital operator 21

## Rising edge (no re-trigger)

libdigital operator 22

## Rising edge (re-trigger)

libdigital operator 22

## Rising hysteresis

libpwnlinear 115

## Rounding function

# Index

---

- libmath 86
- Rounding to ceil function
  - libmath 86
- Rounding to floor function
  - libmath 87
- S**
- Sample time (math operations)
  - libsmk operator 148
- Saturate dynamic
  - libsmk operator 202
- Saturation operators
  - and overflows 28
- Scalar product
  - libmath 97
- Scalar to Boolean
  - libsmk operator 166
- Selector
  - libmath operator 97
- Shift left
  - libimpl operator 44
- Shift right
  - libimpl operator 44
- Sign
  - libmath 78
- Sign (int)
  - libsmk operator 145
- Signal generator from Simulink (libraries)
  - See list 197
  - clock generator 197
  - ground (bool) 197
  - ground (int) 198
  - ground (real) 198

- Signal routing from Simulink (libraries)
  - See list 188
  - if then else 188
  - ignore first input 189
  - merge (2 inputs) 189
  - merge (3 inputs) 190
  - switch 190
  - switch (GTT) 191
  - switch (NEZ) 191
  - switch enumeration 192
  - switch enumeration (GTT) 192
- Sized integer arithmetic (libraries)
  - See list 29
  - division (ceiling rounding) 30
  - division (fix rounding) 30
  - division (floor rounding) 31
  - division (saturated) 31
  - multiplication 32
  - multiplication (int16) 32
  - multiplication (int32) 33
  - multiplication (saturated) 34
  - multiplication (uint16) 33
  - multiplication (uint32) 33
  - negation (saturated) 34
  - subtraction (saturated) 35
  - sum (saturated) 35
- Sized integer comparison (libraries)
  - See list 36
  - different 38
  - equal 37
  - greater than or equal 39

- less than or equal 37
- strictly greater than 38
- strictly less than 36
- Spherical to cartesian
  - libmathext operator 110
  - libsmk operator 167
- Square
  - libmathext 105
- Strictly greater than (integers)
  - libimpl operator 38
- Strictly less than (integers)
  - libimpl operator 36
- Subsystem from Simulink (libraries)
  - See list 193
  - trigger either edge 193
  - trigger falling edge 194
  - trigger rising edge 195
- Switch
  - libsmk operator 190
- Switch (GTT)
  - libsmk operator 191
- Switch (NEZ)
  - libsmk operator 191
- Switch enumeration
  - libsmk operator 192
- Switch enumeration (GTT)
  - libsmk operator 192
- Symmetrical dead band
  - libpwnlinear 114
- Symmetrical limiter
  - libpwnlinear 116
- Symmetrical pre-load

# Index

---

libpwlinear 118

## T

Time operators from Simulink (libraries)

See list 146

decrement time to zero (int) 146

decrement time to zero (real) 147

decrement to zero 147

sample time 148

unit delay 151

unit delay (reset) 151, 152

unit delay enabled 149

unit delay enabled (reset) 150

Timing diagram 213, 214, 215, 216, 217

Toggle

libdigital operator 23

Transfer function (first order)

libsmk operator 139

Transfer function (lead or lag)

libsmk operator 139

Transfer function (real zero)

libsmk operator 140

Transfer functions (libraries)

See list 46

Transformation from Simulink (libraries)

See list 160

cartesian to polar 161

cartesian to spherical 161

polar to cartesian 165

spherical to cartesian 167

Trapeze integrator

liblinear operator 68

Trigger (either)

libdigital operator 23

Trigger (fall)

libdigital operator 24

Trigger (rise)

libdigital operator 24

Trigger either edge

libsmk operator 193

Trigger falling edge

libsmk operator 194

Trigger rising edge

libsmk operator 195

Trigonometric functions (libraries)

See list 102

Truth table operators (libraries)

See list 25

## U

Unit delay

libsmk operator 151

Unit delay (Helper)

libsmk operator 210

Unit delay (reset)

libsmk operator 151, 152

Unit delay enabled

libsmk operator 149

Unit delay enabled (reset)

libsmk operator 150

## V

Vector addition

libmath 98

Vector calculus (libraries)

vector addition 98

vector difference 98

Vector difference

libmath 98

Vector product by matrix

libmath 99

Vector/matrix calculus (libraries)

See list 91

matrix product 95, 96

matrix product by vector 96

scalar product 97

vector by matrix 99

Verification operators (libraries)

AfterNthTick 213

AlwaysAfterFirstCond 214

AtLeastNTicks 215

HasNeverBeenTrue 216

Implies 217

ImpliesWithinNTick 218

## W

Width (matrix)

libsmk operator 211

Width (scalar)

libsmk operator 211

Width (vector)

libsmk operator 212



# Index

---

Wrap to zero

libsmk operator 202

