Ansys

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.

Contact one of our Sales representatives at scade-sales@ansys.com.

Direct general questions about SCADE products to scade-info@ansys.com.

Discover latest news on our products at www.ansys.com/products/embedded-software.

LEGAL INFORMATION

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.

RELATED DOCUMENTS

- *SCADE Suite User Manual*

TYPOGRAPHICAL CONVENTIONS

Bold	GUI buttons, options, tabs, radio buttons, drop-down lists, window titles, panels.
Italics	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

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

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
- libsmlk
- 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 libsmlk, libmath, libdigital, liblinear, libpwlinear or libmathext libraries.

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

Note

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

WHAT IS THE GENERIC FUNCTION OF OPERATORS IN THIS LIBRARY?

- **libsmlk** contains operators designed to reproduce the behavior of Simulink blocks with respect to SCADE Suite formalism. For details, see "Library libsmlk".

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:

Digital Operators

Read the description of the following operators:

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.e.,* (BoolVect2Int([true,false])=2.

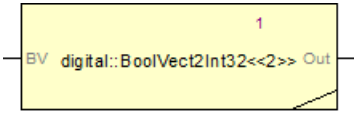
Boolean Vector to Int16

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.e.,* (BoolVect2Int([true,false])=2.

Boolean Vector to Int32

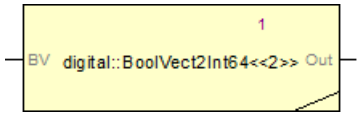
Symbol 2.3:



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.e.,* (BoolVect2Int([true,false])=2.

Boolean Vector to Int64

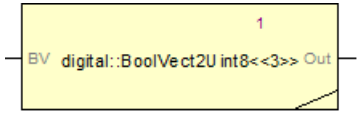
Symbol 2.4:



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.e.,* (BoolVect2Int([true,false])=2.

Boolean Vector to Uint8

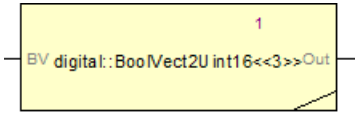
Symbol 2.5:



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.e.,* (BoolVect2Int([true,false])=2.

Boolean Vector to Uint16

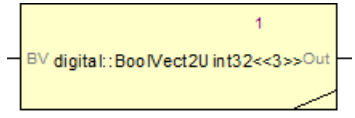
Symbol 2.6:



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.e.,* (BoolVect2Int([true,false])=2.

Boolean Vector to Uint32

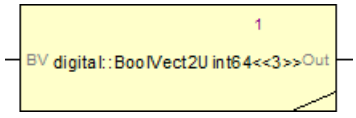
Symbol 2.7:



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.e.,* (BoolVect2Int([true,false])=2.

Boolean Vector to Uint64

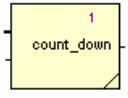
Symbol 2.8:



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.e.,* (BoolVect2Int([true,false])=2.

Count-Down

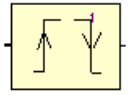
Symbol 2.9:



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

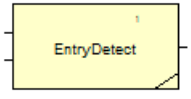
Symbol 2.10:



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`.

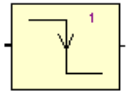
EntryDetect

Symbol 2.11:



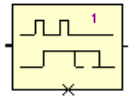
SCADE name	**EntryDetect**
Package	digital
Function	Detects entry into a range based on two Boolean inputs: one input indicates whether the signal is outside the border and the other one whether the signal is at the border.
Inputs	`Outside`: bool and `OnTheBorder`: bool
Outputs	`Entry`: bool

Falling Edge

Symbol 2.12: 

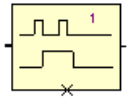
SCADE name	**FallingEdge**
Package	digital
Function	Detects a falling edge (i.e., `true` to `false` transition of Boolean input).
Inputs	`FE_Input`: bool
Outputs	`FE_Output`: bool
Comment	Output is true during transition clock cycle. Output is initialized to `false`.

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.

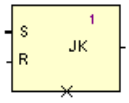
Falling Edge (without Re-Trigger)

Symbol 2.14: 

SCADE name	**FallingEdgeNoRetrigger**
Package	digital
Function	Detects a falling edge (i.e., `true` to `false` transition of Boolean input).
Inputs	`FENR_Input`: bool
Hidden inputs	`NumberOfCycle`: 'T
Outputs	`FENR_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 not detected.

Flip-Flop JK

Symbol 2.15:



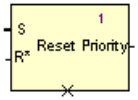
SCADE name	**FlipFlopJK**
Package	digital
Function	Basic JK flip-flop of Boolean values. See Table 2.1 below.
Inputs	`Set` and `Reset`: bool
Hidden inputs	`Init`: bool
Outputs	`FFJK_Output`: bool
Comment	At the first step, the Output takes the same value as Init, regardless of the values of Set and Reset.

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:



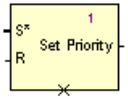
SCADE name	**FlipFlopReset**
Package	digital
Function	Basic flip-flop of Boolean values with priority for reset. See Table 2.2 below.
Inputs	`Set` and `Reset`: bool
Hidden inputs	`Init`: bool
Outputs	`FFR_Output`: bool
KCG pragma	Operator expansion
Comment	At the first step, the Output takes the same 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:



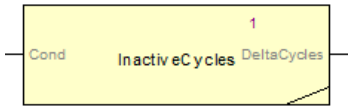
SCADE name	**FlipFlopSet**
Package	digital
Function	Basic flip-flop of Boolean values with priority for set. See Table 2.3 below.
Inputs	`Set` and `Reset`: bool
Hidden inputs	`Init`: bool
Outputs	`FFS_Output`: bool
KCG pragma	Operator expansion
Comment	At the first step, the Output takes the same value 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

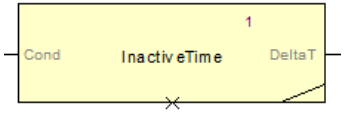
Symbol 2.18:



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

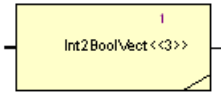
Symbol 2.19:



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

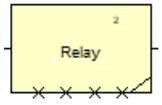
Symbol 2.20:



SCADE name	**Int2BoolVect**
Package	digital
Function	Outputs a vector of booleans BV that is the boolean representation of the integer input, modulo 2^Nbool.
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.

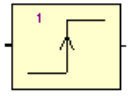
Relay

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`. • when off (*i.e.,* output = `OutputWhenOff`), relay remains off unless the input reaches or exceeds the `UpperLimit_On` (*i.e.,* input >= `UpperLimit_On`) • when on (*i.e.,* output = `OutputWhenOn`), relay remains on unless the input reaches or drops below `LowerLimit_Off` (*i.e.,* input <= `LowerLimit_Off`).

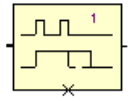
Rising Edge

Symbol 2.22: 

SCADE name	**RisingEdge**
Package	digital
Function	Detects a rising edge (i.e., `false` to `true` 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 `false`.

Rising Edge (Re-Trigger)

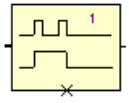
Symbol 2.23:



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

Rising Edge (without Re-Trigger)

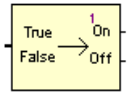
Symbol 2.24:



SCADE name	**RisingEdgeNoRetrigger**
Package	digital
Function	Detects a rising edge (i.e., `false` to `true` 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 `true` as soon as transition is detected and remains unchanged during NumberOfCycle. If another rising edge occurs while the output is `true`, it is not detected. Output is initialized to `false`.

Toggle

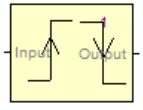
Symbol 2.25:



SCADE name	**Toggle**
Package	digital
Function	Stores the input value into On and Off states. When the input is `true`, On is `true` and Off is `false`. When the input is `false`, On is `false` and Off is `true`.
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)

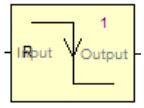
Symbol 2.26:



SCADE name	**TriggerEither**
Package	digital
Function	Output is true if any of the following conditions is true: • Input is 0 and was greater than 0 at previous cycle • Input lower than 0 and was greater than 0 at previous cycle • Input is lower than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle • Input is 0 and was lower than 0 at previous cycle • Input greater than 0 and was lower than 0 at previous cycle • Input is greater than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle
Inputs	`Input`: 'T where 'T is numeric
Outputs	`Output`: bool

Trigger (Fall)

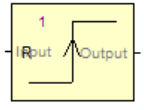
Symbol 2.27:



SCADE name	**TriggerFall**
Package	digital
Function	Output is true if any of the following conditions is true: • Input is 0 and was greater than 0 at previous cycle • Input lower than 0 and was greater than 0 at previous cycle • Input is lower than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle
Inputs	`Input`: 'T where 'T is numeric
Outputs	`Output`: bool

Trigger (Rise)

Symbol 2.28:



SCADE name	**TriggerRise**
Package	digital
Function	Output is true if any of the following conditions is true: • Input is 0 and was lower than 0 at previous cycle • Input greater than 0 and was lower than 0 at previous cycle • Input is greater than 0 and was equal to 0 on previous cycle, and Output was not already true at previous cycle
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)"

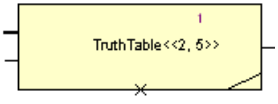
TYPES

This package of library operators includes the following type definition:

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

Truth Table

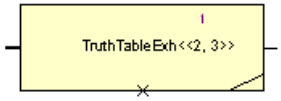
Symbol 2.29:



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)

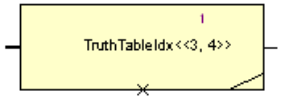
Symbol 2.30:



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).

Access the description of library operators in alphabetical order:

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, mod(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 **`libimpl`** 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)

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

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.

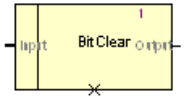
2. Same type as the first operand.

Read the description of the following operators:

“Bit Clear”
“Bit Set”
“Decrease LSB”
“Extract Bit Range”
“Increase LSB to Ceiling”
“Increase LSB to Fix”
“Increase LSB to Floor”
“Shift Left”
“Shift Right”

Bit Clear

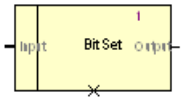
Symbol 3.20:



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

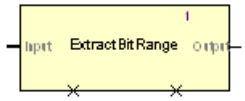
Decrease LSB

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 "Shift Left".
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 (*e.g.,* 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 `Input` from index `Start` to index `End` and returns a value that has exactly those bits (filling the beginning with zeros)
Inputs	`Input`: uint32
Hidden inputs	`Start`: uint32 `End`: uint32
Outputs	`Output`: 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	`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 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 "Shift Right".
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 "Increase LSB to Ceiling", "Increase LSB to Fix", or "Increase LSB to Floor".

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

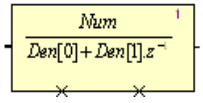
Access the description of library operators in alphabetical order:

Filter and Transfer Function Operators

Read the description of the following operators:

Discrete Filter (Numerator scalar, Denominator deg. 1)

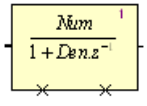
Symbol 4.1:



SCADE name	**Filter01**
Package	filters
Function	Discrete filter expressed as a Z-transform function: `Num / (Den[0] + Den[1]*z^-1)` 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*InX)/Den[0].

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

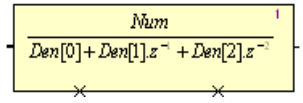
Symbol 4.2:



SCADE name	**Filter01Norm**
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: `Num / (1 + Den*z^-1)` 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*InX.

Discrete Filter (Numerator scalar, Denominator deg. 2)

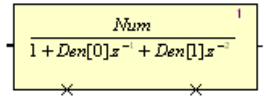
Symbol 4.3:



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)

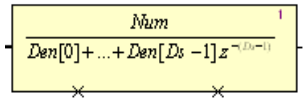
Symbol 4.4:



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)

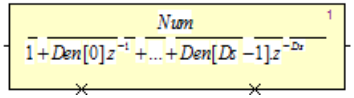
Symbol 4.5:



SCADE name	**Filter0D**
Package	filters
Function	Discrete filter expressed as a Z-transform function: `Num / (Den[0] + Den[1]*z^-1 + ... + Den[Ds - 1]*z^-(Ds - 1))` 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*InX)/Den[0].

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

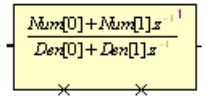
Symbol 4.6:



SCADE name	**Filter0DNorm**
Package	filters
Function	Normalized discrete filter expressed as a Z-transform function: `Num / (1 + Den[0]*z^-1 + ... + Den[Ds - 1]*z^-(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)

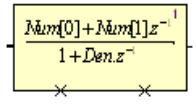
Symbol 4.7:



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)

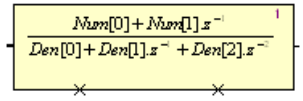
Symbol 4.8:



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)

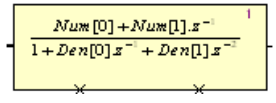
Symbol 4.9:



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)

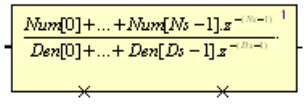
Symbol 4.10:



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)

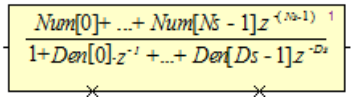
Symbol 4.11:



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)

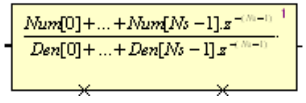
Symbol 4.12:



SCADE name	**FilterNDNorm**
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[Ds - 1]*z^-Ds)`
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)

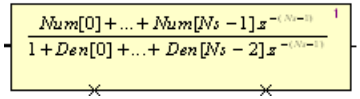
Symbol 4.13:



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)

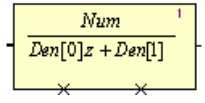
Symbol 4.14:



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)

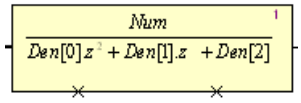
Symbol 4.15:



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)

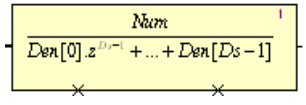
Symbol 4.16:



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)

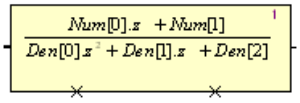
Symbol 4.17:



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)

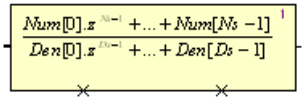
Symbol 4.18:



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:



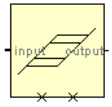
SCADE name	**TransferFcnND**
Package	filters
Function	Discrete transfer function expressed as a Z-transform function: `(Num[0]*z^(Ns - 1) + ... + Num[Ns - 2]*z + Num[Ns - 1]) / (Den[0]*z^(Ds - 1) + ... + Den[Ds - 2]*z + Den[Ds - 1])`
Size parameters	`Ns` is the numerator array size and `Ds` is the denominator 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:

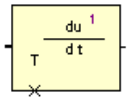
Backlash

Symbol 4.20:



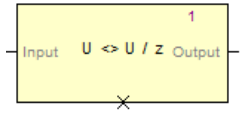
SCADE name	**Backlash**
Package	linear
Function	The behavior is as follows: if the input was decreasing before and is now increasing, the output remains constant until it has crossed `deadband_width`, and then it increases at the same rhythm as the input. And vice versa.
Inputs	`input`: 'T where 'T is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

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

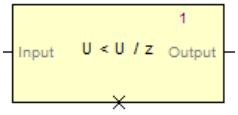
Detect (Change)

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)

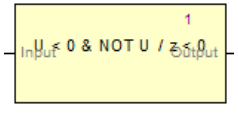
Symbol 4.23:



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)

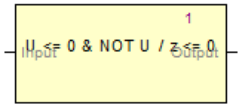
Symbol 4.24:



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)

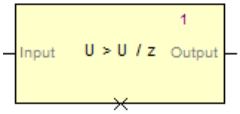
Symbol 4.25:



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

Detect (Increase)

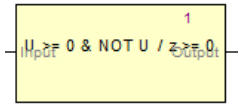
Symbol 4.26:



SCADE name	**DetectIncrease**
Package	linear
Function	The output is `true` 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)

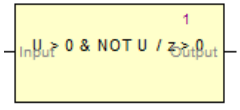
Symbol 4.27:



SCADE name	**DetectRiseNonNegative**
Package	linear
Function	The output is `true` 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)

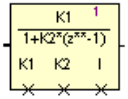
Symbol 4.28:



SCADE name	**DetectRisePositive**
Package	linear
Function	The output is `true` 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)

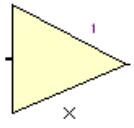
Symbol 4.29:



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^{-1}).

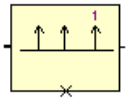
Gain

Symbol 4.30:



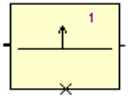
SCADE name	**Gain**
Package	linear
Function	Computes the value of an input multiplied by a set gain value. G_Output equals G_Input multiplied by Gain.
Inputs	`G_Input`: 'T
Hidden inputs	`Gain`: 'T
KCG pragma	Operator expansion
Outputs	`G_Output`: 'T where 'T is numeric

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 `true` 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

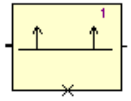
Hit Crossing (Falling Direction)

Symbol 4.32: 

SCADE name	**HitCrossingFalling**
Package	linear
Function	Detects crossing of a specified value by a falling signal. Output is `true` 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)

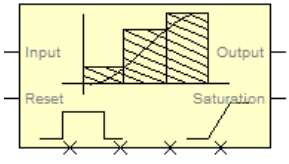
Symbol 4.33:



SCADE name	**HitCrossingRising**
Package	linear
Function	Detects crossing of a specified value by a rising signal. Output is `true` 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)

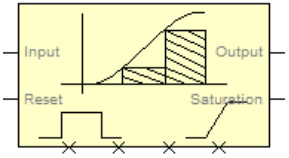
Symbol 4.34:



SCADE name	**IntegralBackward**
Package	linear
Function	Backward Euler discrete-time integrator. Computes the integral of `Input` using the backward rule. If the integral is outside of the `]LowLimit, Highlimit[` range, it is saturated to this range. If saturation is active, the `Saturation` output is `true`.
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)

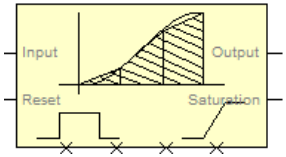
Symbol 4.35:



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

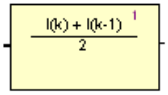
Integrator (Trapezoidal)

Symbol 4.36:



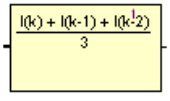
SCADE name	**IntegralTrapezoid**
Package	linear
Function	Trapezoidal discrete-time integrator. The **IntegralTrapezoid** operator computes the integral of `Input` using the trapezoid rule. If the integral is outside of the `]LowLimit, HighLimit[` range, it is saturated to this range. If saturation is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

Mean Cycle (2 values)

Symbol 4.37: 

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

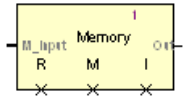
Mean Cycle (3 values)

Symbol 4.38: 

SCADE name	**MeanCycle3**
Package	linear
Function	Computes the mean of three iterative values of the signal input. MeanOn3Steps (0) = $\frac{(U(0) + U(0) + U(0))}{3.0}$ MeanOn3Steps (1) = $\frac{(U(1) + U(0) + U(0))}{3.0}$ 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

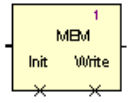
Symbol 4.39:



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

Memory (Basic)

Symbol 4.40:



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

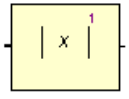
Arithmetic Operators

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

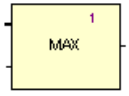
Symbol 5.1:



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)

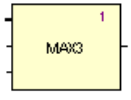
Symbol 5.2:



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)

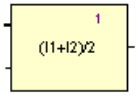
Symbol 5.3:



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)

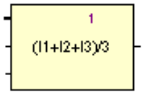
Symbol 5.4:



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)

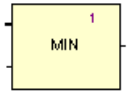
Symbol 5.5:



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)

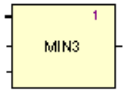
Symbol 5.6:



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

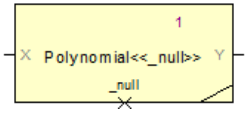
Mod

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

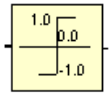
Polynomial

Symbol 5.9:



SCADE name	**Polynomial**
Package	math
Function	If C is the list [$C_{(N-1)}$, $C_{(N-2)}$, ..., C_0,], the operator computes the following: $C_{(N-1)} \cdot X^{(N-1)} + C_{(N-2)} \cdot X^{(N-2)} + ... + C_0$
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.

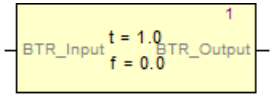
Conversion Operators

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

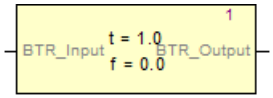
Symbol 5.11:



SCADE name	**BoolToFloat32**
Package	math
Function	Converts a Boolean value into a real value. When the input is `true`, 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

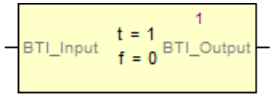
Symbol 5.12:



SCADE name	**BoolToFloat64**
Package	math
Function	Converts a Boolean value into a real value. When the input is `true`, 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

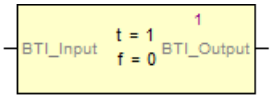
Symbol 5.13:



SCADE name	**BoolToInt8**
Package	math
Function	Converts a Boolean input into an integer value. When the input is `true`, 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

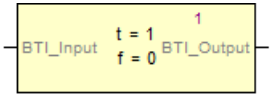
Symbol 5.14:



SCADE name	**BoolToInt16**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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

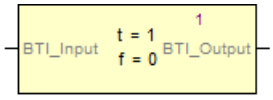
Symbol 5.15:



SCADE name	**BoolToInt32**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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

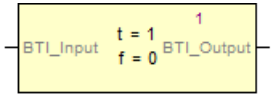
Symbol 5.16:



SCADE name	**BoolToInt64**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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

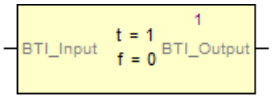
Symbol 5.17:



SCADE name	**BoolToUint8**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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

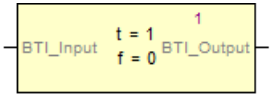
Symbol 5.18:



SCADE name	**BoolToUint16**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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

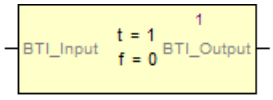
Symbol 5.19:



SCADE name	**BoolToUint32**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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

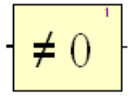
Symbol 5.20:



SCADE name	**BoolToUint64**
Package	math
Function	Converts a Boolean value into an integer value. When the input is `true`, 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:



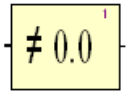
SCADE name	**IntToBool**
Package	math
Function	Converts an integer value into a Boolean value. When the input is 0, the output is equal to `false`, otherwise the output is equal to `true`.
Inputs	`ITB_Input`: 'T where 'T is integer
Outputs	`ITB_Output`: bool
KCG pragma	Operator expansion

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: 

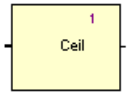
SCADE name	**RealToBool**
Package	math
Function	Converts a real value into a Boolean value. When the input is 0.0, the output is equal to `false`, otherwise the output is equal to `true`.
Inputs	`RTB_Input`: 'T where 'T is float
Outputs	`RTB_Output`: bool
KCG pragma	Operator expansion
Caution	Relies 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

Symbol 5.26: 

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

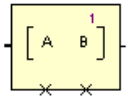
Interval Operators

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

Read the description of the following operators:

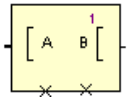
In Range (In-In)

Symbol 5.27:



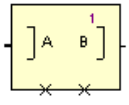
SCADE name	**InRangeInIn**
Package	math
Function	Returns a `true` output if the input belongs to the [A , B] range; otherwise it is `false`.
Inputs	`IRII_Input`: 'T where 'T numeric
Hidden inputs	A: 'T (lower border) and B: 'T (upper border) where 'T numeric
Outputs	`IRII_Output`: bool
KCG pragma	Operator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

In Range (In-Out)

Symbol 5.28: 

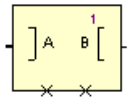
SCADE name	**InRangeInOut**
Package	math
Function	Returns a `true` output if the input belongs to the [A , B [range; otherwise it is `false`.
Inputs	`IRIO_Input`: 'T where 'T numeric Value in the [-MaximumValue ; MaximumValue] range.
Hidden inputs	A: 'T (lower border) and B: 'T (upper border) where 'T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

In Range (Out-In)

Symbol 5.29: 

SCADE name	**InRangeOutIn**
Package	math
Function	Returns a `true` output if the input belongs to the] A , B] range; otherwise it is `false`.
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)

Symbol 5.30: 

SCADE name	**InRangeOutOut**
Package	math
Function	Returns a `true` output if the input belongs to the] A , B [range; otherwise it is `false`.
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

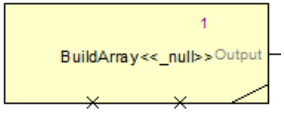
Vector and Matrix Operators

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

Read the description of the following operators:

BuildArray

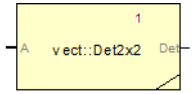
Symbol 5.31:



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

Determinant of Square Matrix (2x2)

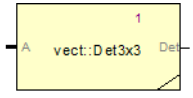
Symbol 5.32:



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)

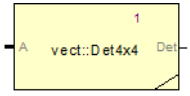
Symbol 5.33:



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)

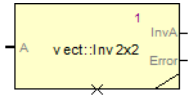
Symbol 5.34:



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)

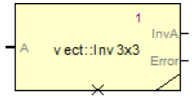
Symbol 5.35:



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)

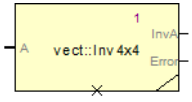
Symbol 5.36:



SCADE name	**Inv3x3**
Package	vect
Function	Inverses a square matrix of type 'T^3^3
Inputs	A: 'T^3^3 where 'T is float
Hidden Inputs	`Epsilon`: 'T where 'T is float
Outputs	`InvA`: 'T ^3^3 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)

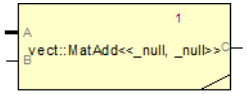
Symbol 5.37:



SCADE name	**Inv4x4**
Package	vect
Function	Inverses a square matrix of type 'T^4^4
Inputs	A: 'T^4^4 where 'T is float
Hidden Inputs	`Epsilon`: 'T where 'T is float
Outputs	`InvA`: 'T ^3^3 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.

Matrix Addition

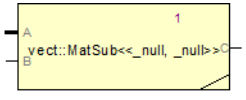
Symbol 5.38:



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

Matrix Difference

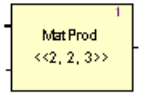
Symbol 5.39:



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

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

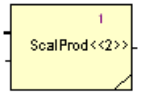
Matrix Product by Vector

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

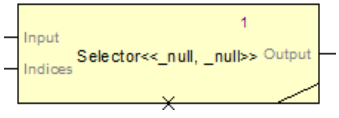
Symbol 5.42:



SCADE name	**ScalProd**
Package	vect
Function	Computes the scalar product of vectors V and W: `SP = V[0]*W[0]+...+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

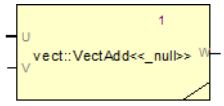
Symbol 5.43:



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

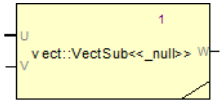
Symbol 5.44:



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`.

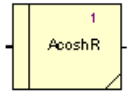
Note

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

Trigonometric and Power-Based Functions

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^x	**ExpR**
	acos (x)	**AcosR**	log (x)	**LogR**
	asin (x)	**AsinR**	$\sqrt{x}$	**SqrtR**
	atan (x)	**AtanR**	10^x	**TenPowR**
		HypotR	(sin(x), cos(x))	**SinCosR**
Inputs	`Input1`: 'T `Input1` and `Input2`: 'T for **Atan2R** `HypotR_I1` and `HypotR_I2`: 'T for **HypotR** where 'T is float			
Outputs	`Output1`: 'T `Output1` and `Output2`: 'T for **SinCosR** `HypotR_O`: 'T for **HypotR** where 'T is float			
Comment	Behavior may vary depending on compiler math libraries.			

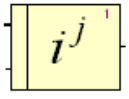
Advanced Mathematical Operators

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

Read the description of the following operators:

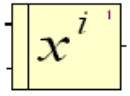
Integer Power for Integers

Symbol 6.2:



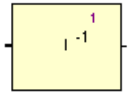
SCADE name	**PowerI**
Package	mathext
Function	`P_Ouput` equals `P_Input` powered to `Power`.
Dependencies	C macro defined in macro_libmathext32.h and macro_libmathext64.h files under SCADE/libraries/SC65/libmathext
Inputs	`P_Input`: int32 and `Power`: int32
Outputs	`P_Output`: int32

Integer Power for Reals

Symbol 6.3: 

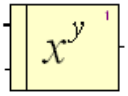
SCADE name	**PowerR**
Package	mathext
Function	`P_Ouput` equals `P_Input` powered to `Power`.
Dependencies	C macro defined in macro_libmathext32.h and macro_libmathext64.h files under SCADE/libraries/SC65/libmathext
Inputs	`P_Input`: 'T and `Power`: int32 where 'T is float
Outputs	`P_Output`: 'T where 'T is float

Inverse

Symbol 6.4: 

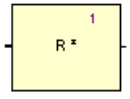
SCADE name	**Inv**
Package	mathext
Function	Computes the reciprocal of a real or int number to return a real output
Inputs	`Inv_In`: 'T where 'T is float
Outputs	`Inv_Out`: 'T where 'T is float
Caution	If Inv_In equals 0, division by zero.

Real Power for Reals

Symbol 6.5: 

SCADE name	**PowerRR**
Package	mathext
Function	`P_Ouput` 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.e.,* 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

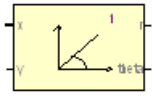
Conversion Operators

The **`libmathext`** library contains conversion functions.

Read the description of the following operators:

Cartesian to Polar

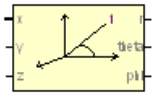
Symbol 6.7:



SCADE name	**CartesianToPolar**
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:

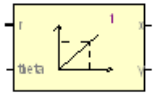
SCADE name	**DegreesToRadians**
Package	libmathext
Function	Conversion from degrees to radians
Inputs	`Degrees`: 'T where 'T is float
Outputs	`Radians`: 'T where 'T is float
KCG pragma	Operator expansion

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	`r`: 'T and `theta`: 'T where 'T is float
Outputs	`x`: 'T and `y`: 'T where 'T is float
KCG pragma	Operator expansion

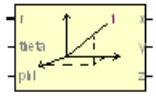
Radians to Degrees

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,θ, φ) into 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

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:

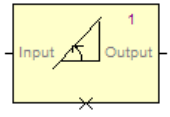
Piecewise Linear Functions

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

Read the description of the following operators:

CheckSlope

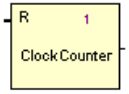
Symbol 7.1:



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

Clock Counter

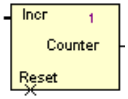
Symbol 7.2:



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

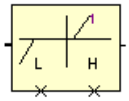
Counter

Symbol 7.3:



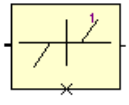
SCADE name	**Counter**
Package	pwlinear
Function	The output increments at each cycle by Incr. At initialization or if Reset equals `true`, the output is set to 0.
Inputs	`Incr`: 'T where 'T numeric
Hidden inputs	`Reset`: bool
Outputs	`Count`: 'T where 'T numeric

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`.

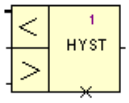
Dead Band (Symmetrical)

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)

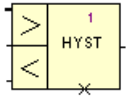
Symbol 7.6:



SCADE name	**FallingHysteresis**
Package	pwlinear
Function	Detects with hystererisis 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)

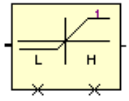
Symbol 7.7:



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)

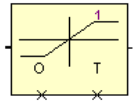
Symbol 7.8:



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)

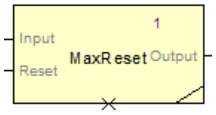
Symbol 7.9:



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:



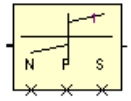
SCADE name	**MaxReset**
Package	pwlinear
Function	When `Reset` is true and before `Init`, `Output` is set to `Init`. Whenever `Input` is greater than the previous `Output`, `Output` is set to `Input`.
Inputs	`Input`: 'T where 'T numeric `Reset`: bool
Hidden inputs	`Init`: 'T where 'T numeric
Outputs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

MinReset

Symbol 7.11:

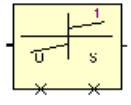
SCADE name	**MinReset**
Package	pwlinear
Function	When `Reset` is true and before `Init`, `Output` is set to `Init`. Whenever `Input` is lower than the previous `Output`, `Output` is set to `Input`.
Inputs	`Input`: 'T where 'T numeric `Reset`: bool
Hidden inputs	`Init`: 'T where 'T numeric
Outputs	`Output`: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

Pre-load (Asymmetrical)

Symbol 7.12: 

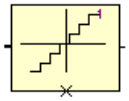
SCADE name	**PreLoadUnSymmetrical**
Package	pwlinear
Function	Asymmetrical preload with specified offset when negative or positive, and slope. The output is equal to (input*Slope)+PosOffset if the input is positive or zero, otherwise equal to (input*Slope)+NegOffset if the input is negative.
Inputs	`PLUS_Input`: '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: 

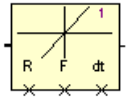
SCADE name	**PreLoadSymmetrical**
Package	pwlinear
Function	Basic symmetrical preload with specified offset and slope. The output is equal to (input*slope) + offset if the input is strictly positive, otherwise the output is equal to (input * slope) - offset.
Inputs	`PLS_Input`: 'T where 'T numeric
Hidden 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))/DeltaT. If Rate > Rising, then output equals (Rising* DeltaT + pre(RL_Output)), else if Rate < Falling, then output equals (Falling* DeltaT + 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 `deltaT`: 'T where 'T is float
Outputs	`RL_Output`: 'T where 'T is float
Comment	For a correct behavior, `deltaT` must be strictly positive. If it equals 0, there is a division by 0.

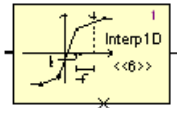
Look-Up Table Operators

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

Read the description of the following operators:

Interpolation 1D

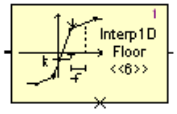
Symbol 7.16:



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)

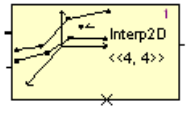
Symbol 7.17:



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	`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 2D

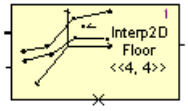
Symbol 7.18:



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	`IdxX`: LutIndex = {k:int32, f:'T} and `IdxY`: LutIndex = {k:int32, f:'T}
Hidden inputs	Z: 'T ^Ys^Xs where 'T is float
Outputs	`OutZ`: 'T where 'T is float

Interpolation 2D (Floor)

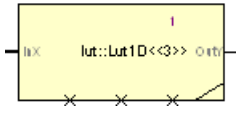
Symbol 7.19:



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	`IdxX`: LutIndex = {k:int32, f:'T} and `IdxY`: LutIndex = {k:int32, f:'T}
Hidden inputs	Z: 'T ^Ys^Xs where 'T is float
Outputs	`OutZ`: 'T where 'T is float

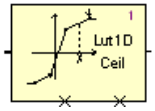
LUT 1D

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

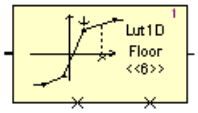
LUT 1D (Value above)

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)

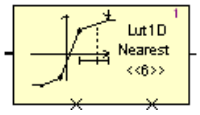
Symbol 7.22:



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)

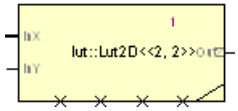
Symbol 7.23:



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

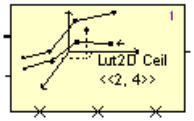
Symbol 7.24:



SCADE name	**Lut2D**
Package	lut
Function	Performs piecewise linear interpolation to approximate z = f(x,y) with <Xs>, <Ys> points set as vectors X, Y, and <Xs>*<Ys> 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)

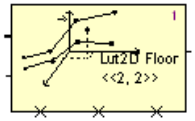
Symbol 7.25:



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)

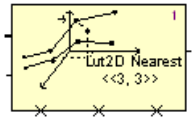
Symbol 7.26:



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)

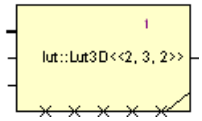
Symbol 7.27:



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

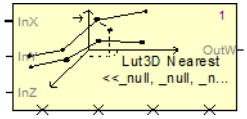
Symbol 7.28:



SCADE name	**Lut3D**
Package	lut
Function	Performs piecewise linear interpolation to approximate w = f(x,y,z) with <Xs>, <Ys>, <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. 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)

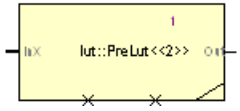
Symbol 7.29:



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

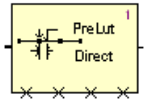
Symbol 7.30:



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 libsmlk

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

Access the description of library operators in alphabetical order:

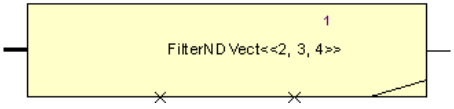
Discrete Filter Operators

Read the description of the following operators:

"Discrete Filter" "Discrete Filter (Normalized)"

Discrete Filter

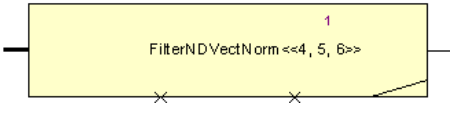
Symbol 8.1:



SCADE names	**FilterNDVect**
Package	smlk
Function	Vectorized version of filters::FilterND (see "Filter and Transfer Function Operators").
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)

Symbol 8.2:

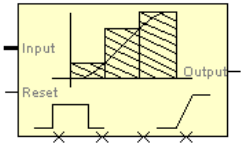


SCADE names	**FilterNDVectNorm**
Package	smlk
Function	Vectorized version of filters::FilterNDNorm (see "Filter and Transfer Function Operators").
Inputs	`In`: 'T where 'T is float
Hidden inputs	`Num`: 'T^Ns^Vect and `Den:` 'T^Ds where 'T is float
Outputs	`Out:` 'T^Vect where 'T is float

Discrete-Time Integrator Operators

Read the description of the following operators:

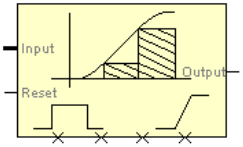
Discrete-Time Integrator (Backward)

Symbol 8.3: 

(DtiBwdLimRst)

SCADE names	Several backward types are available in the library: **DtiBwdLimRst**; **DtiBwdLimRstE**; **DtiBwdLimRstF**; **DtiBwdLimRstL**; **DtiBwdLimRstN**; **DtiBwdLimRstR**; **DtiBwdLimSRst**; **DtiBwdLimSRstE**; **DtiBwdLimSRstF**; **DtiBwdLimSRstL**; **DtiBwdLimSRstN**; **DtiBwdLimSRstR**; **DtiBwdRst**; **DtiBwdRstE**; **DtiBwdRstF**; **DtiBwdRstL**; **DtiBwdRstN**; **DtiBwdRstR** Naming convention: **Bwd** indicates Backward Euler Integrator method **Lim** indicates operators with limits on outputs **S** indicates saturared operators **RstF** indicates falling external reset **RstR** for rising external reset, **RstE** for either external reset, **RstN** for none, **RstL** for external reset level with type inferred for second reset input as int32 or 'T where 'T is float, and **Rst** 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 **DtiBwdRstN**, **DtiBwdLimRstN**, **DtiBwdLimSRstN**)
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 **DtiBwdRstN** only

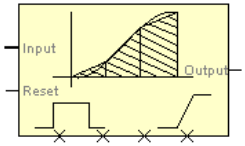
Discrete-Time Integrator (Forward)

Symbol 8.4: 

(DtiFwdLimRst)

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

Discrete-Time Integrator (Trapezoidal)

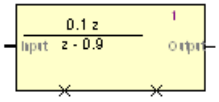
Symbol 8.5: 

(DtiTpzLimRst)

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

Transfer Function (First Order)

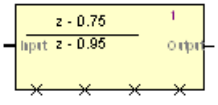
Symbol 8.6:



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)

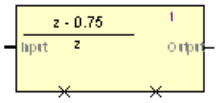
Symbol 8.7:



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:



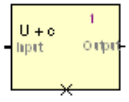
SCADE name	**TransferFcnRealZero**
Package	smlk
Function	Discrete-time transfer function that has real zero and no pole.
Inputs	`Input`: 'T where 'T is float
Hidden inputs	`Zero`: 'T `IC`: 'T where 'T is float
Outputs	`Output`: 'T where 'T is float

Arithmetic Operators

Read the description of the following operators:

Bias

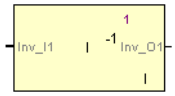
Symbol 8.9:



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)

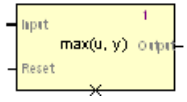
Symbol 8.10:



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)

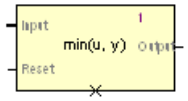
Symbol 8.11:



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)

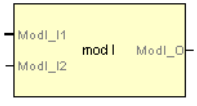
Symbol 8.12:



SCADE name	**MinReset**
Package	smlk
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)

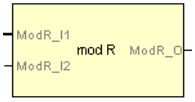
Symbol 8.13:



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

Modulo (real/float)

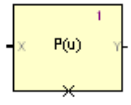
Symbol 8.14:



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

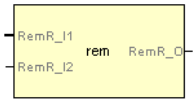
Symbol 8.15:



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)

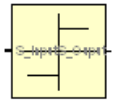
Symbol 8.16:



SCADE name	**RemR**
Package	smlk
Function	Computes the remainder of a division.
Inputs	`RemR_O`: 'T and `RemR_I2`: 'T where 'T is float
Outputs	`RemR_O`: 'T where 'T is float

Sign (int/int32)

Symbol 8.17:



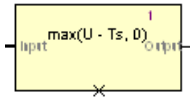
SCADE name	**SignInt**
Package	smlk
Function	Returns an integer sign value based on the value of a numeric input.
Inputs	`S_Input`: 'T where 'T numeric
Outputs	`S_Output`: int32

Time Operators

Read the description of the following operators:

Decrement Time to Zero (int/int32)

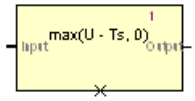
Symbol 8.18:



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)

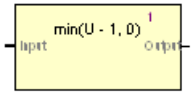
Symbol 8.19:



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

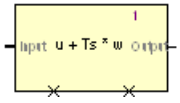
Decrement to Zero

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

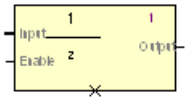
Sample Time (all math operations)

Symbol 8.21:  SampleTimeAdd

SCADE name	Several operations on sample time are available in the library: • Addition: **SampleTimeAdd** • Division: **SampleTimeDiv** • Inverse: **SampleTimeInv** • Multiplication: **SampleTimeMult** • Subtraction: **SampleTimeSub**
Package	smlk
Function	Computes operations involving sample time.
Inputs	For **SampleTimeAdd**, **SampleTimeMult**, **SampleTimeSub**: `Input`: 'T where 'T is numeric For **SampleTimeInv** `Input`: 'T where 'T is float For **SampleTimeDiv**: `Input`: 'T where 'T is float
Hidden inputs	For **SampleTimeAdd**, **SampleTimeMult**, **SampleTimeSub**: `W`: 'T `dT`: 'T For **SampleTimeDiv**: `W`: 'T `dT`: 'T For **SampleTimeInv**: `W`: 'U `dT`: 'U where 'U is float
Outputs	`Output`: 'T where 'T numeric `Output`: 'T (for **SampleTimeDiv**) or 'U (for **SampleTimeInv**) where 'T and 'U are float
KCG pragma	Operator expansion

Unit Delay Enabled

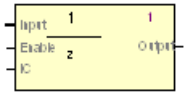
Symbol 8.22:



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)

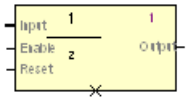
Symbol 8.23:



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

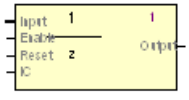
Symbol 8.24:



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)

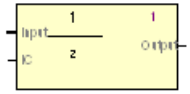
Symbol 8.25:



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)

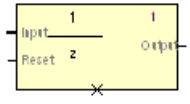
Symbol 8.26:



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

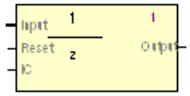
Symbol 8.27:



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:



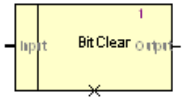
SCADE name	**UDResetInit**
Package	smlk
Function	Delays input signal by one tick. State can be reset based on reset input.
Inputs	`Input`: 'T `Reset`: 'T1 `IC`: 'T
Outputs	`Output`: 'T where 'T numeric
KCG pragma	Operator expansion
Comment	Initial condition (IC) input is not an hidden input.

Bitwise Logical Operators

Read the description of the following operators:

Bit Clear

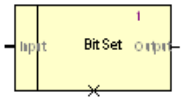
Symbol 8.29:



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

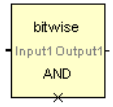
Symbol 8.30:



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

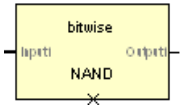
Symbol 8.31:



SCADE name	**BitwiseAnd**
Package	bits
Function	Performs a bitwise logical operation AND.
Inputs	`Input1`: int32
Hidden inputs	`Operand2`: int32
Outputs	`Output1`: int32

Bitwise Nand

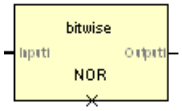
Symbol 8.32:



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

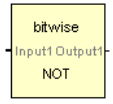
Symbol 8.33:



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

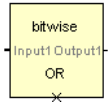
Symbol 8.34:



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

Bitwise OR

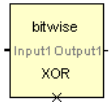
Symbol 8.35:



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

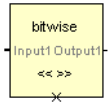
Symbol 8.36:



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

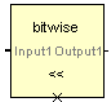
Symbol 8.37:



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

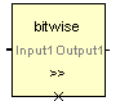
Symbol 8.38:



SCADE name	**BitwiseShiftLeft**
Package	bits
Function	Shifts left a specified number of bits.
Inputs	`Input1`: int32
Hidden inputs	`Operand2`: int32
Outputs	`Output1`: int32

Bitwise Shift Right

Symbol 8.39:



SCADE name	**BitwiseShiftRight**
Package	bits
Function	Shifts right a specified number of bits.
Inputs	`Input1`: int32
Hidden inputs	`Operand2`: int32
Outputs	`Output1`: int32

Extract Bit Range

Symbol 8.40:

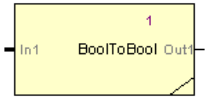
SCADE name	**ExtractBitRange** (imported function)
Package	bits
Function	Outputs a selection of contiguous bits from the input signal.
Inputs	`Input`: int32
Hidden inputs	`Start`: int32 `End`: int32
Outputs	`Output`: int32

Conversion and Transformation Operators

Read the description of the following operators:

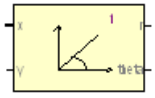
Boolean to Boolean

Symbol 8.41:



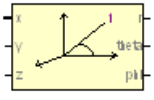
SCADE name	**BoolToBool** (specialization of `ScalarToBool`)
Package	smlkutils
Function	Conversion of scalar values into Boolean values.
Inputs	In1: `bool`
Outputs	Out1: `bool`
KCG pragma	Operator expansion

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

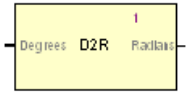
Celsius to Fahrenheit

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

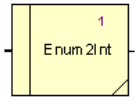
Symbol 8.45:



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

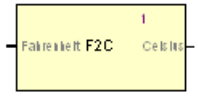
Symbol 8.46:



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

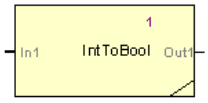
Symbol 8.47:



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

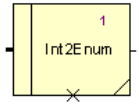
Symbol 8.48:



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

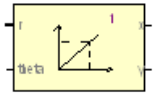
Symbol 8.49:



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

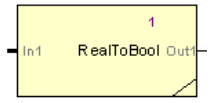
Radians to Degrees

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

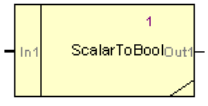
Symbol 8.52:



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

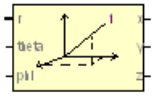
Symbol 8.53:



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:



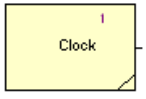
SCADE name	**Spherical2Cartesian**
Package	conv
Function	Transformation from spherical to cartesian coordinates.
Inputs	`r`: 'T , `theta`: 'T, and `phi`: 'T where 'T is float
Outputs	`x`: 'T , `y`: 'T, and `z`: 'T where 'T is float
KCG pragma	Operator expansion

Flip-Flop Operators

Read the description of the following operators:

Clock

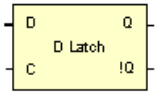
Symbol 8.55:



SCADE name	**Clock**
Package	smlk
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).

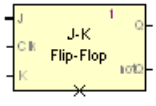
Flip-Flop D

Symbol 8.57:

SCADE name	**FlipFlopD**
Package	flipflop
Function	Basic D flip-flop of Boolean values.
Inputs	`D`: bool, `Clk`: bool, and `notClr`: 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 notClr input is true, otherwise Output Q is false. Output notQ is equal to (not Q).

Flip-Flop JK

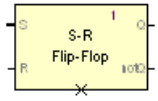
Symbol 8.58:



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: • if not J and not K, Qn= Qn-1 • if not J and K, Qn = false • if J and not K, Qn = true • if J and K, Qn = not Qn-1 Otherwise, Qn = Qn-1 Output notQ is equal to (not Q).

Flip-Flop SR

Symbol 8.59:



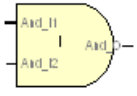
SCADE name	**FlipFlopSR**
Package	flipflop
Function	Basic Set-Reset flip-flop of Boolean values.
Inputs	S: bool and R: bool
Hidden inputs	`IC`: bool
Outputs	`Q`: bool and `notQ`: bool
Comment	On falling edge of Clk input, the following computation is done: • if not S and not R, Qn= Qn-1 • if not S and R, Qn = false • if S and not R, Qn = true • if S and R, Qn = false Output notQ = if (R and notR) false else (not Qn).

Logical Operators

Read the description of the following operators:

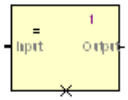
AND

Symbol 8.60:



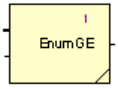
SCADE name	**AndNum**
Package	smlk
Function	Performs logical AND operation with numeric inputs.
Inputs	`And_I1` and `And_I2`: 'T where 'T numeric
Outputs	`And_O`: bool
KCG pragma	Operator expansion

Compare (all operations)

Symbol 8.61:  Equal

SCADE name	Several comparison operations are available in the library: • Equal: **Equal** • Greater or equal: **GreaterEqual** • Greater than: **GreaterThan** • Less or equal: **LessEqual** • Less than: **LessThan** • Not equal: **NotEqual**
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

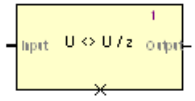
Compare Enumeration

Symbol 8.62: 

(EnumGE)

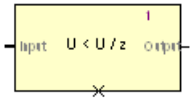
SCADE name	Several comparison operations are available in the library: • Greater or equal: **EnumGE** • Greater than: **EnumGT** • Less or equal: **EnumLE** • Less than: **EnumLT**
Package	cmp
Function	Comparison of enumerated values to their conversion into integers (using `smlk::EnumToInt` specialization)
Inputs	`Input1: 'T` `Input2: 'T`
Outputs	`Output1: bool`
Comment	cmp::EnumLT: Output1 = smlk::EnumToInt(Input1) < smlk::EnumToInt(Input2) cmp::EnumLE: Output1 = smlk::EnumToInt(Input1) <= smlk::EnumToInt(Input2) cmp::EnumGT: Output1 = smlk::EnumToInt(Input1) > smlk::EnumToInt(Input2) cmp::EnumGE: Output1 = smlk::EnumToInt(Input1) >= smlk::EnumToInt(Input2)

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

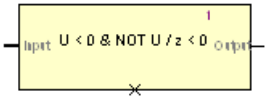
Detect Decrease

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)

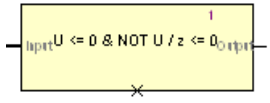
Symbol 8.65:



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)

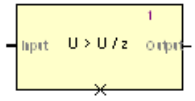
Symbol 8.66:



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

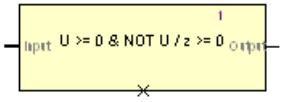
Symbol 8.67:



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)

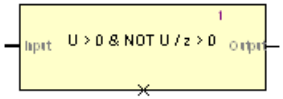
Symbol 8.68:



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)

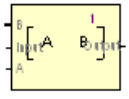
Symbol 8.69:



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

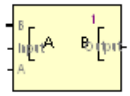
In Range (In-In)

Symbol 8.70:



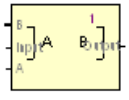
SCADE name	**InRangeInIn**
Package	smlk
Function	Returns a `true` output if the input belongs to the [A , B] range; otherwise it is `false`.
Inputs	B: 'T `Input`: 'T A: 'T where 'T numeric
Outputs	`Output`: bool
KCG pragma	Operator expansion

In Range (In-Out)

Symbol 8.71: 

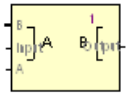
SCADE name	**InRangeInOut**
Package	smlk
Function	Returns a `true` output if the input belongs to the [A , B[range; otherwise it is `false`.
Inputs	B: 'T `Input`: 'T A: 'T where 'T numeric
Outputs	`Output`: bool
KCG pragma	Operator expansion

In Range (Out-In)

Symbol 8.72: 

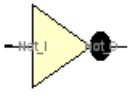
SCADE name	**InRangeIOutIn**
Package	smlk
Function	Returns a `true` output if the input belongs to the]A , B] range; otherwise it is `false`.
Inputs	B: 'T `Input`: 'T A: 'T where 'T numeric
Outputs	`Output`: bool
KCG pragma	Operator expansion

In Range (Out-Out)

Symbol 8.73: 

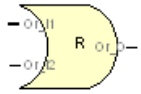
SCADE name	**InRangeOutOut**
Package	smlk
Function	Returns a `true` output if the input belongs to the]A , B[range; otherwise it is `false`.
Inputs	B: 'T `Input`: 'T A: 'T where 'T numeric
Outputs	`Output`: bool
KCG pragma	Operator expansion

NOT

Symbol 8.74: 

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

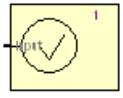
Symbol 8.75: 

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

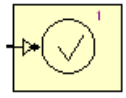
Read the description of the following operators:

Assertion

Symbol 8.76: 

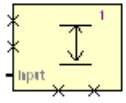
SCADE name	**Assertion**
Package	chk
Function	Checks whether signal is nonzero.
Inputs	Input: 'T
Outputs	none
KCG pragma	Operator expansion

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

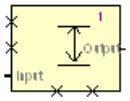
Check Bounds

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)

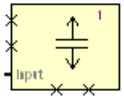
Symbol 8.79:



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

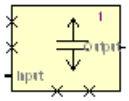
Symbol 8.80:



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)

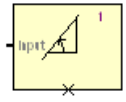
Symbol 8.81:



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

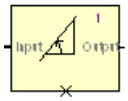
Check Gradient

Symbol 8.82:



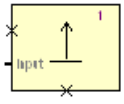
SCADE name	**CheckGradient**
Package	chk
Function	Checks that the expression Input - fby(Input;1;0) is strictly lower than Max.
Inputs	`Input`: 'T
Hidden inputs	`Max`: 'T where 'T numeric
Outputs	none
KCG pragma	Operator expansion

Check Gradient (Extended)

Symbol 8.83: 

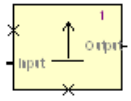
SCADE name	**CheckGradientEx**
Package	chk
Function	Checks that the expression Input - fby(Input;1;0) is strictly lower than Max and returns test results.
Inputs	`Input`: 'T
Hidden inputs	`Max`: 'T where 'T numeric
Outputs	`Output`: bool

Check Lower Bound

Symbol 8.84: 

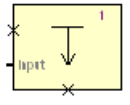
SCADE name	**CheckLowerBound**
Package	chk
Function	Checks that an input is strictly higher than Min.
Inputs	`Input`: 'T
Hidden inputs	`Min`: 'T where 'T numeric `MinOpen`: bool
Outputs	none
KCG pragma	Operator expansion

Check Lower Bound (Extended)

Symbol 8.85: 

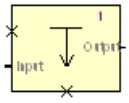
SCADE name	**CheckLowerBoundEx**
Package	chk
Function	Checks that an input is strictly higher than Min and returns test results.
Inputs	`Input`: 'T
Hidden inputs	`Min`: 'T where 'T numeric `MinOpen`: bool
Outputs	`Output`: bool

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: 

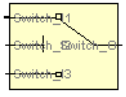
SCADE name	**CheckUpperBoundEx**
Package	chk
Function	Checks that an input is strictly lower than Max and eturns test results.
Inputs	`Input`: 'T
Hidden inputs	`Max`: 'T where 'T numeric `MaxOpen`: boo
Outputs	`Output`: bool

Signal Routing Operators

Read the description of the following operators:

If Then Else

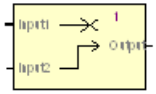
Symbol 8.88:



SCADE name	**IfThenElse**
Package	smlk
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

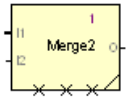
Symbol 8.89:



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

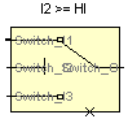
Merge (3 inputs)

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

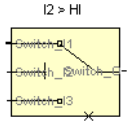
Symbol 8.92:



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)

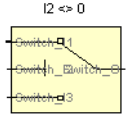
Symbol 8.93:



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)

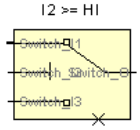
Symbol 8.94:



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

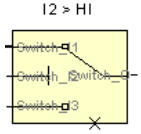
Symbol 8.95:



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:

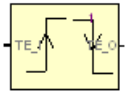


SCADE name	**SwitchGTTEnum**
Package	smlk
Function	Switches output between the first and the third input based on criteria applied to the value of the second input (if I2 > threshold, then O=I1, else O=I3).
Inputs	Switch_I1: 'T, Switch_I2: 'T2, and Switch_I3: 'T
Hidden inputs	Switch_HI: 'T2 (threshold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

Subsystem Operators

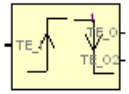
Read the description of the following operators:

Trigger Either Edge

Symbol 8.97: 

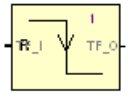
SCADE name	**TriggerEither**
Package	smlk
Function	Detects rising or falling edge (compared to zero) of an input.
Inputs	`TE_I`: 'T where 'T numeric
Outputs	`TE_O`: bool
KCG pragma	Operator expansion

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).

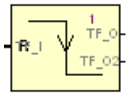
Trigger Falling Edge

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)

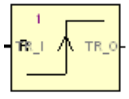
Symbol 8.100:



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

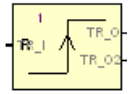
Symbol 8.101:



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:



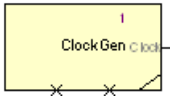
SCADE name	**TriggerRiseEx**
Package	smlk
Function	Detects rising edge (compared to zero) of a real/float input.
Inputs	`TR_I`: 'T where 'T numeric
Outputs	`TR_O`: bool `TR_O2`: int8
Comment	An additional output indicates if the edge is activated (1 for rising or 0 for no edge).

Signal Generator Operators

Read the description of the following operators:

Clock Generator

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 `false`.
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: 

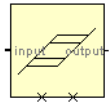
SCADE name	**GroundR**
Package	smlk
Function	Generates a real/float64 output set to **0.0**.
Inputs	None
Outputs	`GR_Out`: float64
KCG pragma	Operator expansion

Discontinuity Operators

Read the description of the following operators:

Backlash

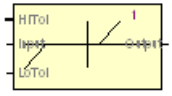
Symbol 8.107:



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

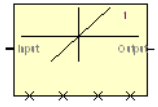
Symbol 8.108:



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

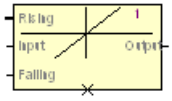
Symbol 8.109:



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

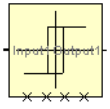
Symbol 8.110:



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))/DeltaT. If Rate > Rising, then RL_Output equals (Rising* DeltaT + pre(RL_Output)) If Rate < Falling, then RL_Output equals (Falling* DeltaT + pre(RL_Output)) Otherwise, RL_Output=RL_Input.

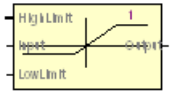
Relay

Symbol 8.111:



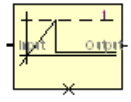
SCADE name	**Relay**
Package	smlk
Function	Basic relay that generates discontinuous output values based on input.
Inputs	`Input1`: 'T where 'T numeric
Hidden inputs	`SwitchOn`: 'T and `SwitchOff`: 'T `OutputOn`: 'T and `OutputOff`: 'T
Outputs	`Output1`: 'T

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

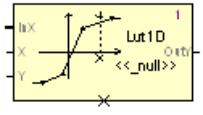
Read the description of the following operators:

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

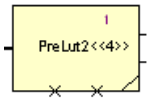
Symbol 8.114:



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

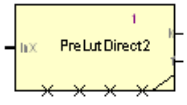
Symbol 8.115:



SCADE name	**PreLut2**
Package	smlk
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)

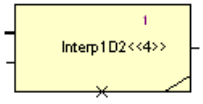
Symbol 8.116:



SCADE name	**PreLutDirect2**
Package	smlk
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

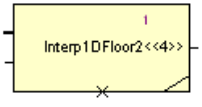
Symbol 8.117:



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	OutY: 'T where 'T is float

Interpolation1D (Floor)

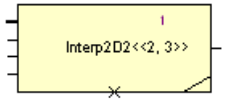
Symbol 8.118:



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	OutY: 'T where 'T is float

Interpolation2D2

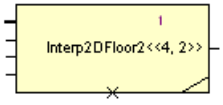
Symbol 8.119:



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	`OutZ`: 'F where 'F is float

Interpolation2D (Floor)

Symbol 8.120:



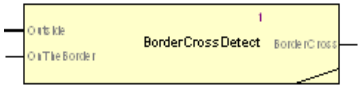
SCADE name	**Interp2DFloor2**
Package	smlk
Function	See introductory note above
Size parameters	Xs and Ys
Inputs	`k1`: int32, `f1`: 'T, `k2`: int32, and `f2`: 'T where 'T is float
Hidden inputs	`Z`: 'T ^Ys^Xs where 'T is float
Outputs	`OutZ`: 'T where 'T is float

Miscellaneous Operators

Read the description of the following operators:

Border Crossing Detection

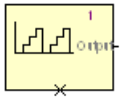
Symbol 8.121:



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)

Symbol 8.122:



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

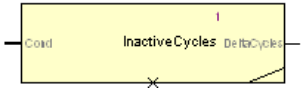
Identity

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:



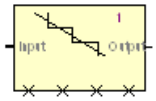
SCADE name	**InactiveCycles**
Package	smlk
Function	Counter that increments an integer output in steps equal to a hidden integer value when Boolean input is `false`, else it returns a constant equal to its hidden value.
Inputs	`Cond`: bool
Hidden inputs	`NbCycles`: 'T where 'T is integer
Outputs	`DeltaCycles`: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

Inactive Time

Symbol 8.126:

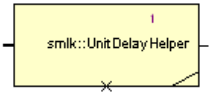
SCADE name	**InactiveTime**
Package	smlk
Function	Counter that increments a real /float output in steps equal to the hidden value when the Boolean input is `false`, else it returns a constant equal to its hidden value.
Inputs	`Cond`: bool
Hidden inputs	`DTInit`: 'T where 'T is numeric
Outputs	`DeltaT`: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

Quantizer

Symbol 8.127: 

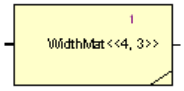
SCADE name	**Quantizer**
Package	smlk
Function	Basic quantizer based on specified interval. Discretizes the input at a specified interval and produces a stair-step output.
Inputs	`Input`: 'T where 'T is float
Hidden inputs	`Bits`: int32, `Umin` and `Umax`: 'T where 'T is float `OutputNegativeValues`: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

Unit Delay (Helper)

Symbol 8.128: 

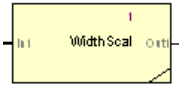
SCADE name	**UnitDelayHelper**
Package	smlk
Function	Performs a unit delay based on init value.
Inputs	Input: 'T
Hidden inputs	`x0`: 'T
Outputs	Output: 'T
KCG pragma	Operator expansion

Width (Matrix)

Symbol 8.129: 

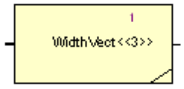
SCADE name	**WidthMat**
Package	smlk
Function	Outputs the number of elements from the input matrix.
Size parameters	N1 and N2
Inputs	`In1`: 'T ^N1^N2 where 'T numeric
Outputs	`Out1`: int32
KCG pragma	Operator expansion
Comment	Simulink block Width mapped to operators WidthScal, WidthVect, or WidthMat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

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 `true` (*i.e.*, 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

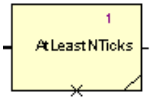
Always After First Condition

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

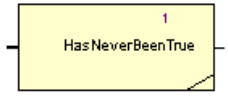
Symbol 9.3: 

SCADE name	**AtLeastNTicks**
Function	Used to express that the output equals the input as soon as the input is `true` for N cycles. Before this N^{th} cycle, the output is `false`.
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:

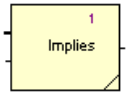


SCADE name	**HasNeverBeenTrue**
Function	Used to express that the output becomes `false` as soon as its input becomes `true` for the first time. After this cycle, the output remains `false`.
Inputs	`Input1`: bool
Outputs	`Output1`: bool
KCG pragma	Operator expansion
Comment	Check the operator timing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

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

Implies

Symbol 9.5:



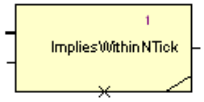
SCADE name	**Implies**
Function	Implements the "implies" logical operator. "If A is `true`, then B is `true`": Not(A) or B.
Inputs	A: bool and B: bool
Outputs	C: bool
KCG pragma	Operator expansion
Comment	Check the operator timing diagram below.

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

Index

Index

E

Index

Index

L

Index

Index

M

Index

Index

Index

Index