POWERFACTORY

PowerFactory 2021

Technical Reference

DIgSILENT Library - DSL Macros Documentation

POWER SYSTEM SOLUTIONS
MADE IN GERMANY

PF2021

Publisher:
DIgSILENT GmbH
Heinrich-Hertz-Straße 9
72810 Gomaringen / Germany
Tel.: +49 (0) 7072-9168-0
Fax: +49 (0) 7072-9168-88
info@digsilent.de

Please visit our homepage at:
https://www.digsilent.de

Copyright © 2021 DIgSILENT GmbH
All rights reserved. No part of this
publication may be reproduced or
distributed in any form without written
permission of DIgSILENT GmbH.

January 21, 2021
PowerFactory 2021
Revision 2

Contents

1 General Description

This document describes the *PowerFactory* global library DSL macros.

1.1 General naming convention for DSL macros

A general naming convention is adopted for the global library macros in *PowerFactory*. The naming convention of DSL macro objects is as follows:

$$\underbrace{FunctionName}_{\text{function identifier}}\ \underbrace{\{s\backslash p\backslash v;s\backslash p\backslash v\}[s\backslash p\backslash v;s\backslash p\backslash v](s\backslash p\backslash v;s\backslash p\backslash v)}_{\text{Limits section}}\ \underbrace{\_opt1\_opt2\_opt3\ldots}_{\text{Options section}}$$

In the above, "s" stand for input signals, "p" for parameters and "v" for internal variables. The DSL macro object name (object parameter *loc_name*) is split in three main parts, as described below:

- the *FunctionName* (mandatory) - identifies the main functionality implemented in the macro (e.g. "1/sT").
- the Limits section (optional) - contains a compact description of limits existing within the macro (e.g. "[p;p]" refers to an upper and lower output limiter).
- the *Options* section (optional) - provides a listing of various macro options or implementation details which have been programmed into the DSL macro (e.g. "_bypass" refers to a function which contains a parameter bypass implementation).

The syntax for limits applied to output signals uses square brackets, as defined below:

- **[s\p\v;** - limiter, lower side (signal/parameter/internal variable), e.g. "1/s [s;" - Integrator macro, lower output limitation with one signal, no upper output limitation;
- **;s\p\v]** - limiter, upper side (signal/parameter/internal variable), e.g. "1/s ;p]" - Integrator macro, upper output limitation with one parameter, no lower output limitation;
- **[s\p\v;s\p\v]** - upper and lower limiter, asymmetrical(i.e. with separate parameters/signals/variables), e.g. "1/s [p;p]" - Integrator macro, lower and upper output limitation with two different parameters (asymmetrical);
- **[s\p\v]** - upper and lower limiter (symmetrical i.e. with single parameter/signal/variable) output limitation, e.g. "1/sT [p]" - Integrator macro, lower and upper limitation with single parameter (symmetrical)

The syntax for limits applied to state variables (e.g. anti-windup limiters) uses round paranthe-ses, as defined below:

- **(s\p\v;** - state variable, low limit (signal/parameter/internal variable), e.g. "1/s (p;" - Integrator macro, state variable low limit with one parameter, no upper state variable limitation;
- **;s\p\v)** - state variable, upper limit (using signal/parameter/internal variable), e.g. "1/s ;p)" - Integrator macro, state variable upper limit with one parameter,no lower state variable limitation;

- **(s\p\v;s\p\v)** - upper and lower state variable limits, asymmetrical(i.e. with separate signals/parameters/internal variables), e.g. “1/s (p;pv)” - Integrator macro, state variable low limit depending on a parameter, state variable upper limit depending on parameter and internal variable;
- **(s\p\v)** - upper and lower state variable limit, symmetrical i.e. with a single parameter/signal/variable, e.g. “1/sT (p)” - Integrator macro, state variable upper and lower limit with single parameter (symmetrical)

The syntax for gradient limits applied to output signals or state variables uses curly braces, as defined below:

- **{s\p\v;** - gradient upon value decrease (using signal/parameter/internal variable), e.g. “1/s {p;” - Integrator macro, gradient function upon value decrease using one slope parameter, no limitation upon value increase;
- **;s\p\v}** - gradient upon value increase (using signal/parameter/internal variable), e.g. “1/s ;p}” - Integrator macro, gradient function upon value increase using one slope parameter, no limitation upon value decrease;
- **{s\p\v;s\p\v}** - gradient upon increase/decrease, asymmetrical(i.e. with separate signals/parameters/variables), e.g. “1/s {p;p}” - Integrator macro, gradient function upon value increase/decrease using two gradient parameters;
- **{s\p\v}** - gradient upon increase/decrease, symmetrical i.e. with a single slope signal/parameter/internal variable, e.g. “1/sT {p}” - Integrator macro, gradient function upon value increase/decrease using a single gradient parameter;

The DSL macro options are listed below:

- **_bypass** - This macro contains a bypass activated by on one of the (input) parameters.
- **_reset** - This macro includes a state/internal variable reset (based on an input signal).
- **_enable** - This macro includes an enable flag (based either on parameter or signal).
- **_trigger** - This macro includes a trigger port (input signal).
- **_fb** - This macro applies a fallback value to a parameter if condition true.
- **_incforward** - This macro contains already initialisation statements (of states, inputs or outputs). Initialisation occurs from input to output (forward initialisation).
- **_incbackward** - This macro contains already initialisation statements (of states, inputs or outputs). Initialisation occurs from output to input (backward initialisation).
- **_incfreeze** - This macro disables parts of its functionality for the duration of the calculation of initial conditions and a short period afterwards (typically defined by a parameter). This is used to avoid dx/dt<>0 messages.
- **_eps** - This macro contains a user defined epsilon parameter for various purposes.
- **_ip** - This macro uses picdro (instead of select or select_const) for logic evaluation (or comparator) in order to avoid toggling effects by internally applying interpolation and re-evaluation.
- **_sig** - This macro is a variant based on an input signal.
- **_par** - This macro is a variant based on a parameter.

The limits and the options may appear in various combinations within practical DSL macros, for example:

- **1/sT {s}[p] _bypass** - Integrator macro with gradient limiter using one signal (symmetrical); lower and upper limitation using one parameter (symmetrical), function implements a bypass
- **1/sT [(p;p)]** - Integrator macro with lower/upper asymetrical limit on output signal and on state variable, using the same parameter for state variable and the output limit
- **1/sT [p;p] _bypass_reset_trigger** - Integrator macro, lower and upper output limitation with two parameters (asymmetrical); with bypass, reset and trigger functions
- **1/sT [(p)]** - Integrator macro, symmetrical limiter on output signal and state variable using a single parameter
- **1/sT {s}[pv; _bypass** - Integrator macro, gradient limiter up/down with one signal (symmetrical); lower output limitation depending on one parameter and one internal variable; with bypass function

Transfer functions shown throughout this document are in their standard form (monic numerator/denominator polynomials and an optional gain). In some cases, for easier notation, a more simplified form has been used.

2 Characteristics

This section provides a complete listing of the existing DSL macros within the *Characteristics* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

2.1 Analytical Characteristics

2.1.1 a(x-c1)(x-c2)

Quadratic function (factorial form)

Functionality: Quadratic function (factorial form)
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Analytical Characteristics*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: a,c1,c2

2.1.2 axˆ2 + bx + c

Quadratic function (general form)

Functionality: Quadratic function (general form)
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Analytical Characteristics*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: a,b,c

2.1.3 mx + n

Linear function/straight line

Functionality: Computes the linear function/straight line
This macro has a linear behaviour.

Macro location: *Macros\Characteristics\Analytical Characteristics*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: m,n

2.2 Lookup Tables

2.2.1 Inverse Lookup array (linear)

Inverse lookup internal array, linear approximation

Functionality: Inverse lapprox function based on internal array.
Identifies the corresponding value of argument x of the function y=f(x), where y and x are provided in the two-column array "array_K".
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

2.2.2 Inverse Lookup array object (linear)

Inverse lookup external array, linear approximation

Functionality: Inverse lapprox function based on external array.
Identifies the corresponding value of argument x of the function y=f(x), where y and x are provided in the two-column array "oarray_K".
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: oarray_K

2.2.3 Lookup array (linear)

Lookup internal array, linear approximation

Functionality: Lookup table function based on internal array, linear approximation. When input is outside pre-defined range, output values are kept constant (characteristic clipping).
Identifies the corresponding value of the function y=f(x), where y and x are provided in the two-column array "array_K".
Uses the internal common model array definition (refer to lapprox() description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

2.2.4 Lookup array (linear_noclipping)

Lookup internal array, linear approximation and extrapolation outside range

Functionality: Lookup table function based on internal array, linear approximation and extrapolation outside range. When "x" values are outside range of pre-defined characteristic, a linear extrapolation is done based on the last two characteristic points of the upper or lower side, depending on the situation.
Identifies the corresponding value of the function y=f(x), where y and x are provided in the two-column array "array_K".
Uses the internal common model array definition (refer to lapprox() description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

2.2.5 Lookup array (spline)

Lookup internal array, spline approximation

Functionality: Lookup table function based on internal array, spline approximation. When input is outside pre-defined range, output values are kept constant (characteristic clipping).
Identifies the corresponding value of the function y=f(x), where y and x are provided in the two-column array "array_K".
Uses the internal common model array definition (refer to sapprox() description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

2.2.6 Lookup array 1x3 (linear_fixed)

Lookup table 1x3, linear approximation, fixed characteristic based on parameters

Look-up table size 1 by 3, input data points using parameters
The set of input data points arr_xn must be monotonically increasing
if vClip is 1 then whenever the input is outside the defined range, the output is set to the last given point
if vClip is 0 then whenever the input is outside the defined range, the output is calculated based on the slope of the last two points in the characteristic
Macro equations
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: arr_x1,arr_x2,arr_x3,arr_y1,arr_y2,arr_y3,vClip
Internal variables: m,n,m1,m2

2.2.7 Lookup array 1x3 (linear_variable)

Lookup table 1x3, linear approximation, variable characteristic based on signals

Look-up table size 1 by 3, input data points using variable signals
The set of input data points arr_xn must be monotonically increasing
if vClip is 1 then whenever the input is outside the defined range, the output is set to the last given point
if vClip is 0 then whenever the input is outside the defined range, the output is calculated based on the slope of the last two points in the characteristic
Macro equations
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,arr_x1,arr_x2,arr_x3,arr_y1,arr_y2,arr_y3

Parameters: vClip
Internal variables: m,n,m1,m2

2.2.8 Lookup array 1x4 (linear_fixed)

Lookup table 1x4, linear approximation, fixed characteristic based on parameters

Look-up table size 1 by 4, input data points using parameters
The set of input data points arr_xn must be monotonically increasing
if vClip is 1 then whenever the input is outside the defined range, the output is set to the last given point
if vClip is 0 then whenever the input is outside the defined range, the output is calculated based on the slope of the last two points in the characteristic
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: arr_x1,arr_x2,arr_x3,arr_x4,arr_y1,arr_y2,arr_y3,arr_y4,vClip
Internal variables: m,n,m1,m2,m3

2.2.9 Lookup array 1x4 (linear_variable)

Lookup table 1x4, linear approximation, variable characteristic based on signals

Look-up table size 1 by 4, input data points using variable signals
The set of input data points arr_xn must be monotonically increasing
if vClip is 1 then whenever the input is outside the defined range, the output is set to the last given point
if vClip is 0 then whenever the input is outside the defined range, the output is calculated based on the slope of the last two points in the characteristic
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,arr_x1,arr_x2,arr_x3,arr_x4,arr_y1,arr_y2,arr_y3,arr_y4
Parameters: vClip
Internal variables: m,n,m1,m2,m3

2.2.10 Lookup array object (linear)

Lookup external array, linear approximation

Functionality: Lookup table function based on external array, linear approximation. When input is outside pre-defined range, output values are kept constant (characteristic clipping).
Identifies the corresponding value of the function y=f(x), where y and x are provided in the two-column array "oarray_K".

Uses the external object IntMat (refer to lapprox description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: oarray_K

2.2.11 Lookup array object (linear_noclipping)

Lookup external array, linear approximation with extrapolation outside range

Functionality: Lookup table function based on external array, linear approximation and extrapolation outside range. When "x" values are outside range of pre-defined characteristic, a linear extrapolation is done based on the last two characteristic points of the upper or lower side, depending on the situation.
Identifies the corresponding value of the function y=f(x), where y and x are provided in the two-column array "oarray_K".
Uses the external object IntMat (refer to lapprox description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: oarray_K

2.2.12 Lookup array object (spline)

Lookup external array, spline approximation

Functionality: Lookup table function based on external array, spline approximation. When input is outside pre-defined range, output values are kept constant (characteristic clipping).
Identifies the corresponding value of the function y=f(x), where y and x are provided in the two-column external array "oarray_K".
Uses the external object IntMat (refer to sapprox description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: oarray_K

2.2.13 Lookup matrix (linear)

Lookup internal matrix, linear approximation

Functionality: Lookup table functionbased on internal matrix, linear approximation.
Identifies the value of the function z=f(x,y), where x and y are the rows/columns of the internal matrix "matrix_K" and the matrix values are the corresponding z=f(x,y).
Uses the internal common model matrix definition (refer to lapprox2 description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: matrix_K

2.2.14 Lookup matrix (spline)

Lookup internal matrix, spline approximation

Functionality: Lookup table function based on internal matrix, spline approximation.
Identifies the value of the function z=f(x,y), where x and y are the rows/columns of the internal matrix "matrix_K" and the matrix values are the corresponding z=f(x,y).
Uses the internal common model matrix definition (refer to sapprox2 description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: matrix_K

2.2.15 Lookup matrix object (linear)

Lookup external matrix, linear approximation

Functionality: Lookup table function based on external matrix, linear approximation.
Identifies the value of the function z=f(x,y), where x and y are the rows/columns of the external matrix "omatrix_K" and the matrix values are the corresponding z=f(x,y).
Uses the external object IntMat (refer to lapprox2 description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: omatrix_K

2.2.16 Lookup matrix object (spline)

Lookup external matrix, spline approximation

Functionality: Lookup table function based on external matrix, spline approximation.
Identifies the value of the function z=f(x,y), where x and y are the rows/columns of the external matrix "omatrix_K" and the matrix values are the corresponding z=f(x,y).
Uses the external object IntMat (refer to sapprox2 description in the User Manual).
This macro has a non-linear behaviour.

Macro location: *Macros\Characteristics\Lookup Tables*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: omatrix_K

3 Comparators

This section provides a complete listing of the existing DSL macros within the *Comparators* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

3.1 Basic Comparators

3.1.1 yi equals C

Checks if the input is equal with a constant (real numbers)

Functionality: Checks if the input is equal with a constant (real numbers)
This macro has a non-linear behaviour.

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C

3.1.2 yi greater than C

Greater than C (real numbers)

Functionality: Returns 1 if input is greater than parameter C. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi > C \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C

3.1.3 yi greater than or equal C

Greater or equal than C (real numbers)

Functionality: Returns 1 if input is greater or equal than parameter C. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi >= C \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C

3.1.4 yi less than C

Less than C (real numbers)

Functionality: Returns 1 if input is less than parameter C. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi < C \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C

3.1.5 yi less than or equal C

Less or equal than C (real numbers)

Functionality: Returns 1 if input is less or equal than parameter C. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi <= C \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C

3.1.6 yi not equals C

Checks if the input is not equal with a constant (real numbers)

Functionality: Checks if the input is not equal with a constant (real numbers)
This macro has a non-linear behaviour.

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C

3.1.7 yi1 equals yi2

Checks if two real valued inputs are equal (real numbers)

Functionality: Checks if two real numbers are equal
This macro has a non-linear behaviour.

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

3.1.8 yi1 greater than or equal yi2

Greater or equal than (real numbers)

Functionality: Returns 1 if first input is greater or equal than the second. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 >= yi2 \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

3.1.9 yi1 greater than yi2

Greater than (real numbers)

Functionality: Returns 1 if first input is greater than the second. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 > yi2 \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

3.1.10 yi1 less or equal than yi2

Less or equal than (real numbers)

Functionality: Returns 1 if first input is less or equal than the second. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 <= yi2 \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

3.1.11 yi1 less than yi2

Less than (real numbers)

Functionality: Returns 1 if first input is less than the second. Returns 0 otherwise.
Function based on select_const().
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 < yi2 \\ 0 & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

3.1.12 yi1 not equals yi2

Checks if two real valued inputs are not equal (real numbers)

Functionality: Checks if two real numbers are not equal
This macro has a non-linear behaviour.

Macro location: *Macros\Comparators\Basic Comparators*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

3.2 Comparators with Pick-up/Drop-off

3.2.1 abs(in) greater than C _ip

Abs lower than C (real numbers, picdro implementation)

Functionality: Returns 1 if absolute value of input is greater than parameter C. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } |yi| > C \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } |yi| <= C \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo

Input signals: yi
Parameters: C,Tpick,Tdrop

3.2.2 abs(in) less than C _ip

Abs lower than (real numbers, picdro implementation)

Functionality: Returns 1 if absolute value of input is smaller than parameter C. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } |yi| < C \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } |yi| >= C \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,Tpick,Tdrop

3.2.3 yi equals C _ip

Checks if input is equal to parameter value (real numbers, picdro implementation)

Functionality: Checks if input is equal to parameter value (real numbers)
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,Tpick,Tdrop

3.2.4 yi greater than C _ip

Larger than C (real numbers, picdro implementation)

Functionality: Returns 1 if input is greater than parameter C. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi > C \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi <= C \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,Tpick,Tdrop

3.2.5 yi greater than or equal C _ip

Larger or equal than C (real numbers, picdro implementation)

Functionality: Returns 1 if input is greater or equal than parameter C. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi >= C \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi < C \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,Tpick,Tdrop

3.2.6 yi less than C _ip

Lower than C (real numbers, picdro implementation)

Functionality: Returns 1 if input is smaller than parameter C. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi < C \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi >= C \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

Parameters: C,Tpick,Tdrop

3.2.7 yi less than or equal C _ip

Lower or equal than C (real numbers, picdro implementation)

Functionality: Returns 1 if input is smaller or equal than parameter C. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi <= C \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi > C \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,Tpick,Tdrop

3.2.8 yi1 equals yi2 _ip

Checks if two real valued inputs are equal (real numbers, picdro implementation)

Functionality: Checks if two real valued inputs are equal
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

3.2.9 yi1 greater than or equal yi2 _ip

Larger or equal than (real numbers, picdro implementation)

Functionality: Returns 1 if first input is greater or equal than the second. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 >= yi2 \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi1 < yi2 \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

3.2.10 yi1 greater than yi2 _ip

Larger than (real numbers, picdro implementation)

Functionality: Returns 1 if first input is greater than the second. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 > yi2 \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi1 <= yi2 \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

3.2.11 yi1 less than or equal yi2 _ip

Lower or equal than (real numbers, picdro implementation)

Functionality: Returns 1 if first input is smaller or equal than the second. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 <= yi2 \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi1 > yi2 \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

3.2.12 yi1 less than yi2 _ip

Lower than (real numbers, picdro implementation)

Functionality: Returns 1 if first input is smaller than the second. Returns 0 otherwise.
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 < yi2 \text{ for } T_{pick} \text{ seconds} \\ 0 & \text{if } yi1 >= yi2 \text{ for } T_{drop} \text{ seconds} \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

3.3 Comparators with Threshold

3.3.1 yi greater than C _eps

Greater than C (real numbers, with threshold)

Functionality: Returns 1 if input is greater than parameter C. Returns 0 otherwise. Function includes a small threshold upon switching around the constant C.
This block avoids toggling behaviour when the input signal value is close to the constant parameter C.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
If set=rst=1 (at within deadband condition) then output is undefined.
If eps=0, then this block is a simple logic function without switching threshold.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi > (C + eps) \\ 0 & \text{if } yi < (C - eps) \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Threshold*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,eps
Internal variables: set,rst

3.3.2 yi less than C _eps

Less than C (real numbers, with threshold)

Functionality: Returns 1 if input is less than parameter C. Returns 0 otherwise. Function includes a small threshold upon switching around the constant C.
This block avoids toggling behaviour when the input signal value is close to the constant parameter C.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
If set=rst=1 (at within deadband condition) then output is undefined.
If eps=0, then this block is a simple logic function without switching threshold.
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi < (C - eps) \\ 0 & \text{if } yi > (C + eps) \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Threshold*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: C,eps
Internal variables: set,rst

3.3.3 yi1 greater than yi2 _eps

Greater than (real numbers, with threshold)

Functionality: Logic "greater than" function with threshold
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 > (yi2 + eps) \\ 0 & \text{if } yi1 < (yi2 - eps) \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Threshold*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: eps
Internal variables: set,rst

3.3.4 yi1 less than yi2 _eps

Less than (real numbers, with threshold)

Functionality: Logic "less than" function with threshold
This macro has a non-linear behaviour.

Function:

$$yo = \begin{cases} 1 & \text{if } yi1 < (yi2 - eps) \\ 0 & \text{if } yi1 > (yi2 + eps) \\ \text{unchanged} & \text{otherwise} \end{cases}$$

Macro location: *Macros\Comparators\Comparators with Threshold*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: eps
Internal variables: set,rst

4 Constants

This section provides a complete listing of the existing DSL macros within the *Constants* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

4.1 -C

Outputs a constant value equal to -C

Functionality: Outputs a constant value equal to -C.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo
Parameters: C

4.2 0

Outputs a constant zero value

Functionality: Outputs a constant zero value.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.3 1

Outputs 1

Functionality: Outputs 1.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.4 1/SQRT2

Outputs 1/SQRT2

Functionality: Outputs 1/SQRT2.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.5 2PI

Outputs 2*PI

Functionality: Outputs 2*PI.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.6 Bias

Outputs a constant value defined at initialization

Functionality: Outputs a constant value defined by the value of the output at initialisation.
This block requires that the macro's output is initialised externally.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo
Internal variables: bias

4.7 C

Outputs a constant (parameter) value

Functionality: Outputs a constant value, defined by parameter C.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo
Parameters: C

4.8 C1/C2

Outputs a constant (parameter) value equal to C1/C2

Functionality: Outputs a constant value, defined by parameter C1 and C2, equal to C1/C2.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo
Parameters: C1,C2

4.9 E

Outputs e

Functionality: Outputs e, base of the natural logarithm.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.10 PI

Outputs PI

Functionality: Outputs PI.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.11 PI/2

Outputs PI/2

Functionality: Outputs PI/2.
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.12 SQRT(2/3)

Outputs SQRT(2/3)

Functionality: Outputs SQRT(2/3)
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.13 SQRT(3/2)

Outputs SQRT(3/2)

Functionality: Outputs SQRT(3/2)
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.14 SQRT(C1/C2)

Outputs the constant value SQRT(C1/C2)

Functionality: Outputs SQRT(C1/C2)
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo
Parameters: C1,C2

4.15 SQRT2

Outputs SQRT2

Functionality: Outputs SQRT2
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

4.16 SQRT3

Outputs SQRT3

Functionality: Outputs SQRT3
This macro has a linear behaviour.

Macro location: *Macros\Constants*
Macro DSL level: *5*
Output signals: yo

5 DSL Special Functions

This section provides a complete listing of the existing DSL macros within the *DSL Special Functions* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

5.1 aflipflop

Function aflipflop(yi,set,rst)

Functionality: Implements the DSL special function aflipflop(), using input signals yi, set and rst.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,set,rst

5.2 balanced

Function balanced()

Functionality: Returns the network representation type (balanced=1 or unbalanced=0).
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo

5.3 delay

Function delay()

Functionality: Applies a time delay of duration T on the input. The output is the delayed value of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: T

5.4 flipflop

Function flipflop(set,rst)

Functionality: Implements the DSL special function flipflop() based on the input signals set and rst.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: set,rst

5.5 gradlim_const

Function gradlim_const()

Functionality: Implements the DSL special function gradlim_const().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation parameters: gradmin
Upper limitation parameters: gradmax

5.6 invlapprox

Function invlapprox()

Functionality: Implements the DSL special function invlapprox().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

5.7 lapprox

Function lapprox()

Functionality: Implements the DSL special function lapprox().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

5.8 lapprox2

Function lapprox2()

Functionality: Implements the DSL special function lapprox2().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: row,col
Parameters: matrix_K

5.9 lapproxext

Function lapproxext()

Functionality: Implements the DSL special function lapproxext().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*

Output signals: yo
Input signals: yi
Parameters: array_K

5.10 lastvalue

Function lastvalue()

Functionality: Implements the DSL special function lastvalue().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

5.11 lim

Function lim(yi,y_min,y_max)

Functionality: Implements the DSL special function lim(). Limits input yi based on input signals y_min and y_max.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,y_min,y_max

5.12 lim_const

Function lim_const(yi,y_min,y_max)

Functionality: Implements the DSL special function lim_const(). Limits the input yi based on the parameters y_min and y_max.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation parameters: y_min
Upper limitation parameters: y_max

5.13 movingavg

Function Moving Average

Functionality: Implements a buffer based moving average filter. The function includes a delay or a buffer based DSL function in the input-output path.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: Tdel,Tlength

5.14 picdro

Function picdro(condition,Tpick,Tdrop)

Functionality: Implements the DSL special function picdro(). Sets the output yo to HIGH if condition is TRUE for at least Tpick seconds. If the output is HIGH, then it sets the output yo to LOW if condition is FALSE for at least Tdrop seconds. Tpick and Tdrop are input signals.
output can be 1 = HIGH or 0 = LOW
if condition >= 0.5 then evaluate to TRUE
if condition < 0.5 then evaluate to FALSE
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: condition,Tpick,Tdrop

5.15 picdro_const

Function picdro_const(condition,Tpick,Tdrop)

Functionality: Implements the DSL special function picdro(). Sets the output yo to HIGH if condition is TRUE for at least Tpick seconds. If the output is HIGH, then it sets the output yo to LOW if condition is FALSE for at least Tdrop seconds. Tpick and Tdrop are parameters.
output can be 1 = HIGH or 0 = LOW
if condition >= 0.5 then evaluate to TRUE
if condition < 0.5 then evaluate to FALSE
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: condition
Parameters: Tpick,Tdrop

5.16 rms

Function rms()

Functionality: Returns the dynamic simulation type (RMS=1 or EMT=0).
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo

5.17 sapprox

Function sapprox()

Functionality: Implements the DSL special function sapprox().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: array_K

5.18 sapprox2

Function sapprox2()

Functionality: Implements the DSL special function sapprox2().
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: row,col
Parameters: matrix_K

5.19 select

Function select(condition,y_true,y_false)

Functionality: Implements the DSL special function select(). Sets the output yo to y_true if condition is TRUE. Sets the output yo to y_false if condition is FALSE.
output can be y_true OR y_false. Both y_true and y_false are input signals.
if condition >= 0.5 then evaluate to TRUE
if condition < 0.5 then evaluate to FALSE
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: condition,y_true,y_false

5.20 select_const

Function select_const(condition,K_true,K_false)

Functionality: Implements the DSL special function select_const(). Sets the output yo to K_true if condition is TRUE. Sets the output yo to K_false if condition is FALSE.
output can be K_true OR K_false. Both K_true and K_false are parameters.
if condition >= 0.5 then evaluate to TRUE
if condition < 0.5 then evaluate to FALSE
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: condition
Parameters: K_true,K_false

5.21 selfix

Function selfix(condition,y_true,y_false)

Functionality: Implements the DSL special function selfix(). Sets the output yo to y_true if condition is TRUE at initialisation. Sets the output yo to y_false if condition is FALSE at initialisation.
output can be y_true OR y_false. Both y_true and y_false are input signals.
if condition >= 0.5 then evaluate to TRUE
if condition < 0.5 then evaluate to FALSE
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: condition,y_true,y_false

5.22 selfix_const

Function selfix_const(condition,K_true,K_false)

Functionality: Implements the DSL special function selfix_const(). Sets the output yo to K_true if condition is TRUE at initialisation. Sets the output yo to K_false if condition is FALSE at initialisation.
output can be K_true OR K_false. Both K_true and K_false are parameters.
if condition >= 0.5 then evaluate to TRUE

if condition < 0.5 then evaluate to FALSE
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: condition
Parameters: K_true,K_false

5.23 time

Function time()

Functionality: Returns the current simulation time.
This macro has a non-linear behaviour.

Macro location: *Macros\DSL Special Functions*
Macro DSL level: *5*
Output signals: yo

6 Deadbands

This section provides a complete listing of the existing DSL macros within the *Deadbands* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

6.1 Backlash

Backlash function

Functionality: This macro implements a backlash function.
Bypass option: if the deadband db is <=0 then yo is constant throughout the simulation.
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: db
Internal variables: d

6.2 Deadband

Continuos deadband

Functionality: This macro implements a continuous deadband block
Returns:
yi-db if yi > db (outside deadband, positive side)
yi+db if yi <-db (outside deadband, negative side)
0 if -db < yi < db (within deadband)

This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db

6.3 Deadband [p;p] _bypass

Continuous deadband with limits (bypass)

Functionality: This macro implements a continuous deadband with limits and bypass
Bypass option: A bypass function is also included if the deadband db is zero.
yi-db if yi > db (outside deadband, positive side)
yi+db if yi <-db (outside deadband, negative side)
0 if -db < yi < db (within deadband)
Output yo is limited between y_min and y_max
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db
Lower limitation parameters: y_min
Upper limitation parameters: y_max

6.4 Deadband _bypass

Continuos deadband (bypass)

Functionality: This macro implements a continuous deadband blockwith bypass
Returns:
yi-db if |yi| > db (outside deadband, positive side)
yi+db if |yi| <-db (outside deadband, negative side)
0 if -db < yi < db (within deadband)
Bypass option: A bypass function is also included if the deadband db is zero.
if db<=0, then yo=yi
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db

6.5 Deadband discontinuous

Discontinuous deadband

Functionality: This macro implements a discontinuous deadband
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db

6.6 Deadband offset [p;p] _bypass

Continuous deadband with offset and limits (bypass)

Functionality: This macro implements a continuous deadband with offset and limits.
Bypass option: A bypass function is also included if the deadband db<=0.
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db
Lower limitation parameters: y_min
Upper limitation parameters: y_max

6.7 Deadband offset _bypass

Continuous deadband with offset (bypass)

Functionality: This macro implements a continuous deadband with offset.
Bypass option: A bypass function is also included if the deadband db<=0.
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db

6.8 Deadband stepped [p;p] _bypass

Stepped deadband with limits (bypass)

Functionality: This macro implements a stepped deadband with limits.
Bypass option: A bypass function is also included if the deadband db<=0.
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db
Lower limitation parameters: y_min
Upper limitation parameters: y_max

6.9 Deadband stepped _bypass

Stepped deadband (bypass)

Functionality: This macro implements a stepped deadband.
Bypass option: A bypass function is also included if the deadband db<=0.
This macro has a non-linear behaviour.

Macro location: *Macros\Deadbands*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: db

7 Delays

This section provides a complete listing of the existing DSL macros within the *Delays* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

7.1 Keˆ-sT _bypass_incforward

Transport delay with gain and initial condition for output, T>=0 (bypass)

Functionality: This macro implements a transport delay with gain and initial condition for output.
The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be non-negative. Otherwise a message is printed to the output window.
Bypass option: If T<=0 then yo=K*yi (gain block)

Forward initial condition option: Output is initialised based on the input
This macro has a non-linear behaviour.

Function:

$$H(s) = K\,e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K,T

7.2 Keˆ-sT _incforward

Transport delay with gain and initial condition for output, T>0

Functionality: This macro implements a transport delay with gain and initial condition for output.
The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be positive. Otherwise a message is printed to the output window.
Forward initial condition option: Output is initialised based on the input
This macro has a non-linear behaviour.

Function:

$$H(s) = K\,e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K, T

7.3 Pade approximant R12 _incbackward

Pade Second Order Approximation (one zero, two poles; bacward initialisation yi<-yo)

Functionality: Implements the time continuous Pade approximation for delays, 2nd order (R1,2): one zero, two poles.

Parameters:
Td - time delay in seconds
A time delay Td smaller than 0.1 ms is not allowed, for robust operation.
Backward initial condition option: Initialisation occurs from the output to the input.

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 - \frac{Td}{2}s + \frac{Td^2}{12}s^2}{1 + \frac{Td}{2}s + \frac{Td^2}{12}s^2}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: Td
Internal variables: A0,A1,A2,B0,B1,offset

7.4 Pade approximant R12 _incforward

Pade Second Order Approximation (one zero, two poles; forward initialisation yi->yo)

Functionality: Implements the time continuous Pade approximation for delays, 2nd order (R1,2): one zero, two poles.

Parameters:
Td - time delay in seconds
A time delay Td smaller than 0.1 ms is not allowed, for robust operation.
Forward initial condition option: Output is initialised based on the input.

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 - \frac{Td}{2}s + \frac{Td^2}{12}s^2}{1 + \frac{Td}{2}s + \frac{Td^2}{12}s^2}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: Td
Internal variables: A0,A1,A2,B0,B1,offset

7.5 eˆ-s0.01 _incforward

Transport delay of 10 ms with initial condition for output

Functionality: This macro implements a transport delay of 10 ms with initial condition for output. The function includes a delay or a buffer based DSL function in the input-output path.
Forward initial condition option: Output is initialised based on the input.
Note: The delay function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
This macro has a non-linear behaviour.

Function:

$$H(s) = e^{-s0.01}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

7.6 eˆ-sT _bypass

Transport delay, T>=0 (bypass)

Functionality: This macro implements a transport delay. The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be non-negative. Otherwise a message is printed to the output window.
Note: The delay function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
Bypass option: If T<=0 then yo=yi (feedthrough block)
This macro has a non-linear behaviour.

Function:

$$H(s) = e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: T

7.7 eˆ-sT _bypass_incbackward

Transport delay, T>=0 (with bypass, backward initialisation: yi <- yo)

Functionality: This macro implements a transport delay with initial condition for output. The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be non-negative. Otherwise a message is printed to the output window.
Note: The delay function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
Bypass option: If T<=0 then yo=yi (feedthrough block)
Backward initial condition option: Input is initialised based on the output.
This macro has a non-linear behaviour.

Function:

$$H(s) = e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: T

7.8 eˆ-sT _bypass_incforward

Transport delay, T>=0 (with bypass, forward initialisation: yi -> yo)

Functionality: This macro implements a transport delay with initial condition for output. The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be non-negative. Otherwise a message is printed to the output window.
Note: The delay function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
Bypass option: If T<=0 then yo=yi (feedthrough block)
Forward initial condition option: Output is initialised based on the input.
This macro has a non-linear behaviour.

Function:

$$H(s) = e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: T

7.9 eˆ-sT _incbackward

Transport delay, T>0 (backward initialisation: yi <- yo)

Functionality: This macro implements a transport delay with initial condition for output. The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be positive. Otherwise a message is printed to the output window.
Backward initial condition option: Input is initialised based on the output.
This macro has a non-linear behaviour.

Function:

$$H(s) = e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo

Input signals: yi
Parameters: T

7.10 eˆ-sT _incforward

Transport delay, T>0 (forward initialisation: yi -> yo)

Functionality: This macro implements a transport delay with initial condition for output. The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T must be positive. Otherwise a message is printed to the output window.
Note: The delay function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
Forward initial condition option: Output is initialised based on the input.
This macro has a non-linear behaviour.

Function:

$$H(s) = e^{-sT}$$

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: T

7.11 lastvalue

Last value function

Functionality: Outputs the value of the input at the previous simulation time step. The function includes a delay or a buffer based DSL function in the input-output path.
Note: The lastvalue function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
This macro has a non-linear behaviour.

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

7.12 lastvalue _incbackward

Last value function (backward initialisation: yi <- yo)

Functionality: Outputs the value of the input at the previous simulation time step. The function includes a delay or a buffer based DSL function in the input-output path.
Note: The lastvalue function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
Backward initial condition option: Input is initialised based on the output.
This macro has a non-linear behaviour.

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

7.13 lastvalue _incforward

Last value function (forward initialisation: yi -> yo)

Functionality: Outputs the value of the input at the previous simulation time step. The function includes a delay or a buffer based DSL function in the input-output path.
Note: The lastvalue function should be provided with a corresponding initial condition on either its input or output. In the case of forward initialisation, the output should be initialised with the value of the input. In the case of backward initialisation, the input should be initialised based on the value of the output.
This macro has a non-linear behaviour.

Macro location: *Macros\Delays*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

8 Derivatives

This section provides a complete listing of the existing DSL macros within the *Derivatives* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

8.1 s/(1+sT)

Derivative with time constant

Functionality: This block implements a first order lag differentiator.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{s}{1+sT} = \frac{1}{T}\frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Derivatives*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Internal variables: dx

8.2 sK/(1+0.01s)

Derivative with gain and first order lag (T=0.01)

Functionality: This block implements a derivative with gain and first order lag (time constant is 0.01 s)
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sK}{1+s0.01} = \frac{K}{0.01}\frac{s}{\frac{1}{0.01}+s}$$

Macro location: *Macros\Derivatives*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K
Internal variables: dx

8.3 sK/(1+sT)

Derivative with gain and time constant

Functionality: This block implements a first order lag differentiator block with gain K and time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sK}{1+sT} = \frac{K}{T}\frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Derivatives*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

Internal variables: dx

8.4 sK/(1+sT) _fb

Derivative with gain and time constant T (fallback value)

Functionality: This block implements a first order lag differentiator block with gain K and time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: If T<=0, the use T=0.01
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sK}{1+sT} = \frac{K}{T}\frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Derivatives*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T
Internal variables: dx

9 Electric Power

This section provides a complete listing of the existing DSL macros within the *Electric Power* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

9.1 El. Power

Output generator electrical power in MVA/MW base (for synchronous generators)

Functionality: This macro outputs the generator electrical power in generator MVA base (IPB=1). or in generator MW base (IPB=0)

This macro has a linear behaviour.

Macro location: *Macros\Electric Power*
Macro DSL level: *5*
Output signals: pelec
Input signals: pgt,cosn
Parameters: IPB

9.2 PQ Calculator

Calculates P and Q from complex U and I values

Functionality: Computes the active and reactive power of complex current and voltage.
This macro has a linear behaviour.

Macro location: *Macros\Electric Power*
Macro DSL level: *5*
Output signals: P,Q
Input signals: ur,ui,ir,ii

9.3 Power_base

Conversion of base for power (for synchronous generators)

Functionality: This macro tranforms the p.u. generator electrical power in the base defined by
parameter PN. If PN =0, then no conversion is performed.
This macro has a linear behaviour.

Macro location: *Macros\Electric Power*
Macro DSL level: *5*
Output signals: pelec
Input signals: pg,sgnn,cosn
Parameters: PN

10 Filters

This section provides a complete listing of the existing DSL macros within the *Filters* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

10.1 Band Pass Filters

10.1.1 (H0w0/Q)s/(w0ˆ2+sw0/Q+sˆ2)

Second Order Band Pass Filter

Functionality: Second order band-pass filter function
Parameters:
Flow - in Hz, lower cut-off frequency
Fhigh - in Hz, upper cut-off frequency
H0 - circuit gain (at center frequency)
Q - filter selectivity equal to sqrt(Fhigh*Flow)/(Fhigh-Flow) (e.g. narrow frequency bands lead to high selectivity)
w0 - in rad/s, center frequency equal to 2*pi()*sqrt(Fhigh*Flow)

Forward initialisation: output is initialised based on input
This macro has a linear behaviour.

Function:

$$H(s) = \frac{H0 \cdot \omega_0}{Q} \frac{s}{\omega_0^2 + \frac{\omega_0}{Q} s + s^2}$$

Macro location: *Macros\Filters\Band Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: Flow,Fhigh,H0
Internal variables: w0,Q,F0

10.1.2 s/(w0ˆ2+sˆ2)

Resonant Filter with center frequency w0

Functionality: Resonant filter function
Parameters:
f0 - in Hz, filter center frequency
w0 - in rad/s, filter angular frequency, equal to 2*PI*f0
Forward initialisation: output is initialised based on input
This macro has a linear behaviour.

Function:

$$H(s) = \frac{s}{\omega_0^2 + s^2}$$

Macro location: *Macros\Filters\Band Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: f0
Internal variables: w0

10.2 High Pass Filters

10.2.1 -sT/(1+sT)

Negative Derivative with time constant T

Functionality: This macro implements a first order differentiator with time constant T and gain -1.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{-sT}{1+sT} = -1\frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Internal variables: dx

10.2.2 -sT/(1+sT) _bypass

Negative Derivative (bypass)

Functionality: This macro implements a first order differentiator with time constant T and gain -1.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, then output is zero.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{-sT}{1+sT} = -1\frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Internal variables: dx

10.2.3 sKT/(1+sT)

First order lag differentiator with gain and derivative time constant

Functionality: This block implements a first order lag differentiator with gain and derivative time constant
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sKT}{1+sT} = K\frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*

Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T
Internal variables: dx

10.2.4 sKTd/(1+sT)

First order lag differentiator with gain, derivative and lag time constant

Functionality: This block implements a first order lag differentiator with gain, derivative and lag time constant
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sKT_d}{1 + sT} = K\frac{T_d}{T}\frac{s}{\frac{1}{T} + s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,Td,T
Internal variables: dx

10.2.5 sT/(1+sT) _bypass

First order lag differentiator, time constant T (bypass)

Functionality: This block implements a first order lag differentiator block with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, then block is a feedthrough (output is equal to input).
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sT}{1 + sT} = \frac{s}{\frac{1}{T} + s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Internal variables: dx

10.2.6 sT/(1+sT) _enable

First order lag differentiator, time constant T (with enable)

Functionality: This block implements a first order lag differentiator block with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Enable option:
if T>0, then output is enabled (normal transfer function)
if T<=0, then output is 0 and state variable is frozen
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sT}{1+sT} = \frac{s}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Internal variables: dx

10.2.7 sTb/(1+sTa) _fb

First order lag differentiator, time constant Ta (fallback)

Functionality: This block implements a first order lag differentiator, time constant Ta (bypass)
Parameter Ta must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: if Ta<=0, then assign Ta=0.01
This macro has a linear behaviour.

Function:

$$H(s) = \frac{sT_b}{1+sT_a} = \frac{T_b}{T_a}\frac{s}{\frac{1}{T_a}+s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta
Internal variables: dx

10.2.8 sTld/(1+sTlg) _fb

First order lag differentiator, time constant Tlg (fallback value)

Functionality: This block implements a first order lag differentiator with time constant Tlg and derivative time constant Tld

Parameter Tlg must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: If Tlg<=0, then assign Tlg=0.01

This macro has a linear behaviour.

Function:

$$H(s) = \frac{sT_{ld}}{1 + sT_{lg}} = \frac{T_{ld}}{T_{lg}} \frac{s}{\frac{1}{T_{lg}} + s}$$

Macro location: *Macros\Filters\High Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tld,Tlg
Internal variables: dx

10.3 Low Pass Filters

10.3.1 (1-K)/(1+sT)

First order lag with 1-K gain and time constant T

Functionality: This macro implements a first order lag with 1-K gain and time constant T.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1-K}{1 + sT} = \frac{1-K}{T} \frac{1}{\frac{1}{T} + s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.2 (1-K)/(1+sT) _bypass

First order lag with 1-K gain and time constant (bypass)

Functionality: This macro implements a first order lag with 1-K gain and time constant. It also includes a bypass function.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0 then block is a gain, yo=(1-K)*yi
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1-K}{1+sT} = \frac{1-K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.3 1/(1+sT)

First order lag

Functionality: This macro implements a first order lag block with time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T

10.3.4 1/(1+sT) (p)

First order lag, anti-windup limiter, symmetrical

Functionality: This macro implements a first order lag, state variable limit, symmetrical
Time constant T must be positive for correct operation
Upper limitation y_max parameter must be non-negative for correct operation. Otherwise, a message is printed to the output window.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

Continuous states: x
Parameters: T
Upper limitation parameters: y_max

10.3.5 1/(1+sT) (p;p)[p;p]

First order lag, state and derivative rate limit

Functionality: This macro implements a first order lag, state and derivative rate limit
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Lower limitation parameters: y_min,r_min
Upper limitation parameters: y_max,r_max

10.3.6 1/(1+sT) (s)

First order lag, state limit, symmetrical

Functionality: This macro implements a first order lag, state limit based on signal, symmetrical
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: y_max
Continuous states: x
Parameters: T

10.3.7 1/(1+sT) (s;s)

First order lag, state limit

Functionality: This macro implements a first order lag, state limit based on signals (asymmetrical)
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: y_max
Continuous states: x
Parameters: T

10.3.8 1/(1+sT) [(p;p)]

First order lag, state limit based on parameters (asymmetrical)

Functionality: This macro implements a first order lag, state limit based on parameters (asymmetrical)
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Upper limitation parameters: y_max

10.3.9 1/(1+sT) _bypass

First order lag (bypass)

Functionality: This macro implements a first order lag block with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.

Bypass option: if T<=0, block is bypassed (yo=yi)
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T

10.3.10 1/(1+sT) _enable

First order lag with enable signal

Functionality: This block implements a first order lag block with time constant T and enable signal.
Parameter T must be positive. Otherwise, a message is printed to the output window.
Enable option:
If enable < 0.5 then output and state are frozen
If enable >= 0.5 then filter is enabled.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: enable
Continuous states: x
Parameters: T

10.3.11 1/(1+sT) and sx

First order delay (PT1)

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*

Macro DSL level: *5*
Output signals: yo,dx
Input signals: yi
Continuous states: x
Parameters: T

10.3.12 1/(1+sT) {p;p} _fb

First order lag with derivative rate limits (bypass)

Functionality: This macro implements a first order lag with derivative rate limits (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0 then yo = yi
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT} = \frac{1}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Lower limitation parameters: rdown
Upper limitation parameters: rup

10.3.13 1/(1+sT/2)

First order lag variant (T/2 time constant)

Functionality: This macro implements a first order lag variant (T/2 time constant)
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{1+sT/2} = \frac{2}{T}\frac{1}{\frac{2}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T

10.3.14 1/(K+sT)

First order lag with 1/T gain and T/K time constant

Functionality: This macro implements a first order delay with gain 1/K and time constant T/K
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{K + sT} = \frac{1}{T} \frac{1}{\frac{K}{T} + s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.15 1/(K+sT) _bypass

First order lag with 1/T gain and T/K time constant (bypass)

Functionality: This macro implements a first order delay with gain 1/K and time constant T/K
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0 then yo = yi/K
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{K + sT} = \frac{1}{T} \frac{1}{\frac{K}{T} + s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.16 1/K(T1/T2-1)(1/(1+sT2))

First order lag variant, 3 parameters

Functionality: This macro implements a first order lag variant, 3 parameters
Parameter T2 must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{\frac{T_1}{T_2} - 1}{K} \frac{1}{1 + sT_2} = \frac{\frac{T_1}{T_2} - 1}{KT_2} \frac{1}{\frac{1}{T_2} + s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T1,T2

10.3.17 1/K/(1+sT) _bypass

First order lag with gain 1/KT (bypass)

Functionality: This macro implements a first order lag with gain 1/K.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Parameter K must be positive. Otherwise, a message is printed to the output window.
Bypass option: If T<=0, block is a gain 1/K
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{K}\frac{1}{1+sT} = \frac{1}{KT}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.18 K/(1+sT)

First order lag with gain

Functionality: This macro implements a first order lag block with gain K/T and time constant T.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K}{1+sT} = \frac{K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.19 K/(1+sT) (p;s) _bypass

First order lag with gain, state limited (bypass)

Functionality: This macro implements a state limited first order lag block with gain K/T and time constant T.
Upper limit is an input signal, lower limit a parameter.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, then block is a gain (yo=K*yi) with limited output.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{K}{1+sT} = \frac{K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: y_max
Continuous states: x
Parameters: K,T
Lower limitation parameters: y_min

10.3.20 K/(1+sT) (s;s) _bypass

First order lag with gain, state limited (bypass)

Functionality: This macro implements a state limited first order lag block with gain K/T and time constant T.
Upper and lower limit is an input signal.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, then block is a gain (yo=K*yi) with limited output.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{K}{1+sT} = \frac{K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: y_max
Continuous states: x
Parameters: K,T

10.3.21 K/(1+sT) (sp;sp)

First order lag with gain, state limited, limits proportional to limiter input signal

Functionality: This macro implements a state limited first order lag block with gain K/T and time constant T
Limits are proportional to limiter signal.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{K}{1+sT} = \frac{K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: ylim
Continuous states: x
Parameters: K,T
Lower limitation parameters: y_min
Upper limitation parameters: y_max

10.3.22 K/(1+sT) [(p;p)] _bypass

First order lag with gain, state limited (bypass)

Functionality: This macro implements a state limited first order lag block with bypass and gain.
The state variable and the output are limited using parameters "y_min" and "y_max".
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, block is a limited gain yo = K*yi between y_min and y_max limits.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{K}{1+sT} = \frac{K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T
Lower limitation parameters: y_min
Upper limitation parameters: y_max

10.3.23 K/(1+sT) _bypass

First order lag with gain (bypass)

Functionality: This macro implements a first order lag block with gain K/T and time constant T.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, then block is a gain (yo=K*yi) with limited output.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K}{1+sT} = \frac{K}{T}\frac{1}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.3.24 K1 + K2/(1+sT)

First order lag in parallel with gain, 3 parameters

Functionality: This block implements a first order lag in parallel with gain.
This block uses 3 parameters: K1, K2 and T.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = K_1 + \frac{K_2}{1+sT} = K_1\frac{\frac{K_1+K_2}{K_1 T}+s}{\frac{1}{T}+s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K1,K2,T

10.3.25 KT/(1+sT)

First order lag with gain KT

Functionality: This block implements a first order lag with gain K*T.
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{KT}{1 + sT} = K\frac{1}{\frac{1}{T} + s}$$

Macro location: *Macros\Filters\Low Pass Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

10.4 Moving Average Filters

10.4.1 MovingAverage (stateless)

Function Moving Average

Functionality: Implements a buffer based moving average filter. The function includes a delay or a buffer based DSL function in the input-output path.
This macro has a non-linear behaviour.

Macro location: *Macros\Filters\Moving Average Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: Tdel,Tlength

10.4.2 MovingAverage _enable_incforward

Moving Average Filter (time continuous implementation, without buffers, with enable)

Moving average filter with Tavg sliding window and enable signal
Implemented based on a 2nd order Pade approximation, as below
This macro has a linear behaviour.

Macro location: *Macros\Filters\Moving Average Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: enable
Continuous states: x1,x2,x3
Parameters: Tavg
Internal variables: x3_del,tinit

10.4.3 MovingAverage _incforward

Moving Average Filter (time continuous implementation, without buffers)

Moving average filter with Tavg sliding window
Implemented based on a 2nd order Pade approximation, as below
This macro has a linear behaviour.

Macro location: *Macros\Filters\Moving Average Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2,x3
Parameters: Tavg
Internal variables: x3_del,tinit

10.5 Notch Filters

10.5.1 (sˆ2+wnˆ2)/(sˆ2+sBw+wnˆ2)

Notch filter

Functionality: Block that implements a notch filter

Example of initialisation conditions in the main model:
inc(x2)= 0
inc(x1) = yi/sqr(wn)
vardef(wn) = 'rad/s';'Notch filter frequency'
vardef(Bw) = 'rad/s';'Notch filter 3db bandwidth'

This macro has a linear behaviour.

Function:

$$H(s) = \frac{w_n^2 + s^2}{w_n^2 + B_w s + s^2}$$

Macro location: *Macros\Filters\Notch Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: Bw,wn
Internal variables: dx1,dx2

10.5.2 (sˆ2+wnˆ2)/(sˆ2+sBw+wnˆ2) _bypass

Notch filter (bypass)

Functionality: Block that implements a notch filter with bypass for Bw parameter.

Bypass: If Bw <= 0 and wn <= 0 then output = input

Example of initialisation conditions in the main model
inc(x2)= 0
inc(x1) = selfix({abs(Bw)<=0.0}.and.{abs(wn)<=0.0}, yi, yi/sqr(wn))
vardef(wn) = 'rad/s';'Notch filter frequency'
vardef(Bw) = 'rad/s';'Notch filter 3db bandwidth'

This macro has a linear behaviour.

Function:

$$H(s) = \frac{w_n^2 + s^2}{w_n^2 + B_w s + s^2}$$

Macro location: *Macros\Filters\Notch Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: Bw,wn
Internal variables: dx1,dx2

10.6 Other Filters

10.6.1 ((1+sTz)/(1+sTp)ˆM)ˆN

Ramp-tracking filter

Functionality: This block implements a transfer function as described in IEEE Std 421.5 Recommended Practice for Excitation System Models For Power System Stability Studies (Models PSS2A/PSS2B/PSS2C).
Parameter Tp must be positive. Otherwise, a message is printed to the output window.
Parameters M and N must have integer values between 0 and 8 (included). If outside range, a message is printed to the output window. Furthermore, the product N*M must not be greater than 8 (included).
This macro has a linear behaviour.

Function:

$$H(s) = [\frac{1 + sT_z}{(1 + sT_p)^M}]^N$$

Macro location: *Macros\Filters\Other Filters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16
Parameters: Tz,Tp,N,M
Internal variables: yll,nl,yo1,yo2,yo3,yo4,yo5,yo6,yo7,yo8,yo9,yo10,yo11,yo12,yo13,yo14,yo15,yo16,dx1,dx2,dx3,

11 Gains

This section provides a complete listing of the existing DSL macros within the *Gains* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

11.1 Multiply (-1)

Sign invert

Functionality: Multiplies the input by -1.
This macro has a linear behaviour.

Function:

$$H(s) = -1$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.2 Multiply (-K)

Gain -K

Functionality: Multiplies the input by -K.
This macro has a linear behaviour.

Function:

$$H(s) = -K$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K

11.3 Multiply (1-K)

Gain 1-K

Functionality: Multiplies the input by (1-K).
This macro has a linear behaviour.

Function:

$$H(s) = 1 - K$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K

11.4 Multiply (1-K1-K2)

Gain 1-K1-K2

Functionality: Multiplies the input by (1-K1-K2).
This macro has a linear behaviour.

Function:

$$H(s) = 1 - K1 - K2$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2

11.5 Multiply 1

Feedthrough block

Functionality: Feedthrough block, yo=yi.
This macro has a linear behaviour.

Function:

$$H(s) = 1$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.6 Multiply 1/K

Inverse Gain

Functionality: Multiplies the input by 1/K. If K=0 then output is zero
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{K}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K

11.7 Multiply 1/K [p;p]

Inverted Gain with limits

Functionality: Multiplies the input by 1/K and limits the output within y_min and y_max.
If K=0, output is zero (upper/lower limits still applicable on 0).
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{K}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K
Lower limitation parameters: y_min
Upper limitation parameters: y_max

11.8 Multiply 1/K1K2 [p;p]

Inverted Gain, 2 parameters, with limits

Functionality: Multiplies the input by 1/(K1*K2) and limits the output within y_min and y_max.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{K_1 \cdot K_2}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2

Lower limitation parameters: y_min
Upper limitation parameters: y_max

11.9 Multiply 1/SQRT2

Gain 1 over square root 2

Functionality: Divides the input by sqrt(2).
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{\sqrt{2}}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.10 Multiply 1/SQRT3

Gain 1 over square root 3

Functionality: Divides the input by sqrt(3).
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{\sqrt{3}}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.11 Multiply K

Gain K

Functionality: Multiplies the input by K.
This macro has a linear behaviour.

Function:

$$H(s) = K$$

Macro location: *Macros\Gains*

Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K

11.12 Multiply K [p;p]

Gain K with limits

Functionality: Multiplies the input by K and limits the output within y_min and y_max values.
This macro has a non-linear behaviour.

Function:

$$H(s) = K$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K
Lower limitation parameters: y_min
Upper limitation parameters: y_max

11.13 Multiply K1 K2

Gain K1 * K2

Functionality: Multiplies the input by K1*K2.
This macro has a linear behaviour.

Function:

$$H(s) = K_1 \cdot K_2$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2

11.14 Multiply K1 K2 / K3

Gain K1*K2/K3

Functionality: Multiplies the input by K1*K2/K3.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K_1 \cdot K_2}{K_3}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2,K3

11.15 Multiply K1 K2 K3

Gain K1 * K2 * K3

Functionality: Multiplies the input signal with three constant parameters K1, K2 and K3.
This macro has a linear behaviour.

Function:

$$H(s) = K_1 \cdot K_2 \cdot K_3$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2,K3

11.16 Multiply K1/K2

Gain K1/K2

Functionality: Multiplies the input by K1/K2.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K_1}{K_2}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2

11.17 Multiply PI

Gain of PI

Functionality: Multiplies the input by PI.
This macro has a linear behaviour.

Function:

$$H(s) = \pi$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.18 Multiply SQRT(2/3)

Gain square root 2 over square root 3

Functionality: Multiplies the input by sqrt(2/3).
This macro has a linear behaviour.

Function:

$$H(s) = \sqrt{\frac{2}{3}}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.19 Multiply SQRT(3/2)

Gain square root 3 over square root 2

Functionality: Multiplies the input by sqrt(3/2).
This macro has a linear behaviour.

Function:

$$H(s) = \sqrt{\frac{3}{2}}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.20 Multiply SQRT(K1/K2)

Gain square root of K1/K2

Functionality: Multiplies the input by sqrt(K1/K2), where K1 and K2 are constant parameters.
This macro has a linear behaviour.

Function:

$$H(s) = \sqrt{\frac{K1}{K2}}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K1,K2

11.21 Multiply SQRT2

Gain square root 2

Functionality: Multiplies the input by sqrt(2).
This macro has a linear behaviour.

Function:

$$H(s) = \sqrt{2}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

11.22 Multiply SQRT3

Gain square root 3

Functionality: Multiplies the input by sqrt(3).
This macro has a linear behaviour.

Function:

$$H(s) = \sqrt{3}$$

Macro location: *Macros\Gains*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

12 Higher Order Transfer Functions

This section provides a complete listing of the existing DSL macros within the *Higher Order Transfer Functions* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

12.1 (1+B1s+B2ss)/(1+A1s+A2ss) _bypass

(1+B1s+B2ss)/(1+A1s+A2ss) (bypass)

Functionality: This block implements H(s)= (1+B1s+B2ss)/(1+A1s+A2ss).
Parameter A2 must be positive. Otherwise, a message is printed to the output window.
Bypasss option: A bypass is included based on parameters A1, A2, B1 and B2.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + B_1 s + B_2 s^2}{1 + A_1 s + A_2 s^2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: A1,A2,B1,B2
Internal variables: triv,dx1,dx2

12.2 (1+KsTc)/((1+sTa)(1+sTb)(1+sTc)) _bypass

(1+KsTc)/((1+sTa)(1+sTb)(1+sTc)) (bypass)

Functionality: This block implements H(s)=(1+KsTc)/((1+sTa)(1+sTb)(1+sTc))
Parameters Ta, Tb and Tc must be positive. Otherwise, a message is printed to the output window.
Bypass option: A bypass is included based on parameters Ta, Tb and Tc.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + K \cdot T_c s}{(1 + sT_a)(1 + sT_b)(1 + sT_c)}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: xa,xb,xc
Parameters: Ta,Tb,Tc,K
Internal variables: dxc,yxa,yxb

12.3 (1+b1s+b2ss)/(1+a1s+a2ss)

(1+b1s+b2ss)/(1+a1s+a2ss)

Functionality: This block implements H(s)= (1+b1s+b2ss)/(1+a1s+a2ss)
Parameter a2 must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + b_1 s + b_2 s^2}{1 + a_1 s + a_2 s^2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: a1,a2,b1,b2
Internal variables: dx1,dx2

12.4 (1+sT3)/(1+sT1+ssT1T2) _bypass

(1+sT3)/(1+sT1+ssT1T2) (bypass)

Functionality: This block implements H(s)= (1+sT3)/(1+sT1+ssT1T2)
Parameters T1 and T2 must be non-negative. Otherwise, a message is printed to the output window.
Bypass option:
if T1>0 and T2>0 then function is (1+sT3)/(1+sT1+ssT2)
if T1=0 and T2>0 then function is (1+sT3)/(1+ssT2)
if T1>0 and T2=0 then function is 1/(1+sT1)
if T1=0 and T2=0 then function is 1 (yo=yi)

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + sT_3}{1 + sT_1 + s^2 T_1 T_2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: T1,T2,T3
Internal variables: dx1

12.5 (1+sT3+ssT4)/(1+sT1+ssT2)

(1+sT3+ssT4)/(1+sT1+ssT2)

Functionality: This block implements H(s)=(1+sT3+ssT4)/(1+sT1+ssT2)
Parameter T2 must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + sT_3 + s^2T_4}{1 + sT_1 + s^2T_2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: T1,T2,T3,T4
Internal variables: dx1,dx2

12.6 (1+sTb)(sTa)ˆ2/(1+sTa)ˆ4

(1+sTb)(sTa)_2/(1+sTa)_4

Functionality: This block implements H(s)=(1+sTb)(sTa)ˆ2/(1+sTa)ˆ4
Parameters Ta and Tb must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{(1 + sT_b)(sT_a)^2}{(1 + sT_a)^4}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2,x3,x4
Parameters: Tb,Ta
Internal variables: yo1,yo2,yo3,dx1,dx2,dx3

12.7 (1+ssT3)/(1+sT1+ssT2) _bypass

(1+ssT3)/(1+sT1+ssT2) (bypass)

Functionality: This block implements H(s)= (1+ssT3)/(1+sT1+ssT2)
Parameters T1 and T2 must be non-negative. Otherwise, a message is printed to the output window.
Bypass:
if T1>0 and T2>0 then function is (1+ssT3)/(1+sT1+ssT2)

if T1=0 and T2>0 then function is (1+ssT3)/(1+ssT2)
if T1>0 and T2=0 then function is 1/(1+sT1)
if T1=0 and T2=0 then function is 1 (yo=yi)

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + s^2 T_3}{1 + sT_1 + s^2 T_2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: T1,T2,T3
Internal variables: dx2

12.8 (A0+sA1+ssA2)/(B0+sB1+ssB2) _bypass

(A0+sA1+ssA2)/(B0+sB1+ssB2) (bypass)

Functionality: This block implements H(s)= (A0+sA1+ssA2)/(B0+sB1+ssB2).
Parameter B2 must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: A bypass is included on parameter B2.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{A_0 + sA_1 + s^2 A_2}{B_0 + sB_1 + s^2 B_2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: B0,B1,B2,A0,A1,A2
Internal variables: dx1,dx2

12.9 (ss)/(Ass+Bs+1) _bypass

(ss)/(Ass+Bs+1) (bypass)

Functionality: This block implements H(s)=(ss)/(Ass+Bs+1).
Parameter A must be positive. Otherwise, a message is printed to the output window.
Bypass option: A bypass is included on parameters A and B.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{s^2}{As^2 + Bs + 1}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: A,B
Internal variables: triv,dx1,dx2

12.10 (ss+ww)/(ss+sB+ww) _bypass

(ss+ww)/(ss+sB+ww) (bypass)

Functionality: This block implements H(s)=(ss+ww)/(ss+sB+ww).
Bypass option: A bypass is included on parameter B and w.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{\omega^2 + s^2}{s^2 + sB + \omega^2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: B,w
Internal variables: triv,dx1,dx2

12.11 1/(1+s(2 x zeta)/wc+ ss/(wc x wc))

Second order low pass filter with cutoff frequency wc and damping factor zeta

Functionality: This block implements second order low pass filter with characteristic frequency wc and damping factor zeta
vardef(wc)='rad/s';'Cutoff frequency'
vardef(zeta)='n/a';'Damping factor'
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{1 + \frac{2\zeta}{\omega_c}s + \frac{1}{\omega_c^2}s^2} = \frac{\omega_c^2}{\omega_c^2 + 2\zeta\omega_c s + s^2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*

Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: wc,zeta

12.12 1/(1+sT1+ssT2) _bypass

Second order lag (bypass)

Functionality: This block implements H(s)=(1/(1+sT1+ssT2)).
Parameters T1 and T2 must be positive. Otherwise, a message is printed to the output window.
Bypass option: A bypass is included on parameters T1 and T2.
if T1<=0 and T2<=0 -> yo=yi (bypass T2 and T1 parts)
if T1<=0 and T2 >0 -> Block is H(s)=(1/(1+ssT2)) (bypass T1 part)
if T1>0 and T2 <=0 -> Block is H(s)=(1/(1+sT1)) (bypass T2 part)
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{1 + sT_1 + s^2 T_2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: T1,T2

12.13 K(1+sT1)(1+sT2)/(s(1+sT3)) [(p;p)]

K(1+sT1)(1+sT2)/(s(1+sT3)) [(p;p)]

Functionality: This block implements H(s)=K(1+sT1)(1+sT2)/(s(1+sT3)).
State variable and output limits are applied using the same parameters.
This macro has a non-linear behaviour.

Function:

$$H(s) = K\frac{(1 + sT_1)(1 + sT_2)}{s(1 + sT_3)}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: K,T1,T2,T3
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: dx1,yo1,yo2

12.14 K(1+sTd)/((1+sTa)(1+sTb)s) (p;p) _bypass

K(1+sTd)/((1+sTa)(1+sTb)s) (p;p) (bypass)

Functionality: This block implements H(s)=K(1+sTd)/((1+sTa)(1+sTb)s).
State limits by parameters and a bypass function are applied.
Parameters Ta and Tb must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: A bypass is included on parameters Ta and Tb.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{K(1 + sT_d}{s(1 + sT_a)(1 + sT_b)}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: xa,xb,xc
Parameters: K,Td,Ta,Tb
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: xbout,xaout,dxa

12.15 e(-sTd)/((1+sT1)(1+sT1))

Second order lag with ideal delay

Functionality: This block implements a second order lag with ideal delay. The function includes a delay or a buffer based DSL function in the input-output path.
Parameter T1, T2 and Td must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{e^{-sT_d}}{(1 + sT_1)^2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: T1,T2,Td

12.16 sˆ2K/(1+sT)ˆ2

Second order lag differentiator with gain

Functionality: This block implements a second order lag differentiator with gain
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{Ks^2}{(1+sT)^2}$$

Macro location: *Macros\Higher Order Transfer Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: K,T
Internal variables: dx1,dx2

13 Integrators

This section provides a complete listing of the existing DSL macros within the *Integrators* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

13.1 with reset (to initial value)

13.1.1 1/s (p;p) _reset

Integrator, state limited by parameters and with reset (to initial value)

Functionality: This macro implements an integrator, state limited by parameters.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Lower limitation parameters: y_min

Upper limitation parameters: y_max
Internal variables: xinc

13.1.2 1/s (s;s) _enable_reset

Integrator with signal limits (asymmetrical), hold signal and with reset (to initial value)

Functionality: This macro implements an integrator with signal limits (asymmetrical)
Option _enable:
If hold signal >=0.5 then integrator is frozen
If hold signal < 0.5 then integrator is unfrozen
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,hold
Lower limitation input signals: y_min
Upper limitation input signals: rst,y_max
Continuous states: x
Internal variables: xinc

13.1.3 1/s [p;p] _reset

Integrator with parameter limits (asymmetrical) and reset (to initial value)

Functionality: This macro implements an integrator with parameter limits (asymmetrical)
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Lower limitation parameters: y_min
Upper limitation parameters: y_max

Internal variables: xinc

13.1.4 1/s [p] _reset

Integrator with parameter limits (symmetrical) and with reset (to initial value)

Functionality: This macro implements an integrator with parameter limits (symmetrical)
Parameter y_max must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Upper limitation parameters: y_max
Internal variables: xinc

13.1.5 1/s [s;s] _reset

Integrator with signal limits (asymmetrical) and with reset (to initial value)

Functionality: This macro implements an integrator with signal limits (asymmetrical)
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: rst,y_max
Continuous states: x
Internal variables: xinc

13.1.6 1/s [s] _reset

Integrator with signal limits (symmetrical) and with reset (to initial value)

Functionality: This macro implements an integrator with signal limits (symmetrical)
Input signal must always be positive in order to operate correctly.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst,y_max
Continuous states: x
Internal variables: xinc

13.1.7 1/s _enable_reset

Integrator with reset (to initial value)

Functionality: This macro implements a continuous time integrator block.
Option _enable:
If hold signal >=0.5 then integrator is frozen
If hold signal < 0.5 then integrator is unfrozen
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,hold
Upper limitation input signals: rst
Continuous states: x
Internal variables: xinc

13.1.8 1/s _incfreeze _reset

Integrator with delayed start based on parameter Tincfreeze and with reset (to initial value)

Functionality: Integrator with delayed start based on parameter Tincfreeze
Option _incfreeze: blocks integrator at initialization and afterwards for a duration of Tincfreeze seconds.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: Tincfreeze
Internal variables: t0,xinc

13.1.9 1/s _reset

Integrator with reset (to initial value)

Functionality: This macro implements a continuous time integrator block and reset.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Internal variables: xinc

13.1.10 1/sT _reset

Integrator, time constant T and with reset (to initial value)

Functionality: This macro implements an integrator block with time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Internal variables: xinc

13.1.11 1/sT (p) _reset

Integrator, state limits, symmetrical, time constant T and with reset (to initial value)

Functionality: This macro implements a state limited integrator with time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.12 1/sT (p) _fb_reset

Integrator, state limits, symmetrical, time constant T (fallback) and with reset (to initial value)

Functionality: This macro implements a state limited integrator with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.13 1/sT (p; _reset

Integrator, state lower limit, time constant T and with reset (to initial value)

Functionality: This macro implements an integrator, state lower limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Internal variables: xinc

13.1.14 1/sT (p; _fb_reset

Integrator, state lower limit, time constant T (fallback) and with reset (to initial value)

Functionality: This macro implements a state lower limited integrator with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Internal variables: xinc

13.1.15 1/sT (p;p) _fb_reset

Integrator, state limits, time constant T (fallback) and with reset (to initial value)

Functionality: This macro implements a state limited integrator with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: xinc

13.1.16 1/sT (p;s) _fb_reset

Integrator, state limits based on parameter/signal, time constant T (fallback) and with reset

Functionality: This macro implements an integrator, state limits based on parameter/signal, time constant T (bypass)
Lower limits is parameter; upper limit is input signal
Parameter T must be non-negative. Otherwise, a message is printed to the output window.

Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst,y_max
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Internal variables: xinc

13.1.17 1/sT (pp;pp) _reset

Integrator, parameter scaled limits, time constant T and with reset (to initial value)

Functionality: This macro implements an integrator, parameter scaled limits, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T,K
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: xinc

13.1.18 1/sT (s;p) _fb_reset

Integrator, state limits based on signal/parameter, time constant T (fb) and with reset

Functionality: This macro implements an integrator, state limits based on signal/parameter, time constant T (bypass)

Lower limit is a signal; upper limit is a parameter.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.19 1/sT (s;s) _fb_reset

Integrator, state limits by signals, time constant T (fb) and with reset (to initial value)

Functionality: This macro implements an integrator, state limits by signals, time constant T (bypass)
Lower and upper limits are input signals.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: rst,y_max
Continuous states: x
Parameters: T
Internal variables: xinc

13.1.20 1/sT ;p) _fb_reset

Integrator, state lower limit, time constant T (fb) and with reset (to initial value)

Functionality: This macro implements an integrator, state lower limit, time constant T (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.21 1/sT ;p) _reset

Integrator, state lower limit, time constant T and with reset (to initial value)

Functionality: This macro implements an integrator, state lower limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.22 1/sT ;p] _fb_reset

Integrator, state upper limit, time constant T (fb) and with reset (to initial value)

Functionality: This macro implements an integrator, state upper limit, time constant T (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.23 1/sT ;p] _reset

Integrator, state upper limit, time constant T and with reset (to initial value)

Functionality: This macro implements an integrator, state upper limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max
Internal variables: xinc

13.1.24 1/sT [p; _fb_reset

Integrator, state lower limit, time constant T (fb) and with reset (to initial value)

Functionality: This macro implements an integrator, state lower limit, time constant T (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Internal variables: xinc

13.1.25 1/sT [p; _reset

Integrator, state lower limit, time constant T and with reset (to initial value)

Functionality: This macro implements an integrator, state lower limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Internal variables: xinc

13.1.26 1/sT _bypass_incfreeze_reset

Integrator with time constant T and delayed start based on parameter Tincfreeze and with reset

Functionality: Integrator with time constant T and delayed start based on parameter Tincfreeze
Option _bypass: if time constant T<=0 then block is a feedthrough
Option _incfreeze: blocks integrator at initialization and afterwards for a duration of Tincfreeze seconds.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T,Tincfreeze
Internal variables: t0,xinc

13.1.27 1/sT _fb_reset

Integrator, time constant T (fb) and with reset (to initial value)

Functionality: This macro implements an integrator block with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, yo keeps its initial value
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Internal variables: xinc

13.1.28 1/sT _incfreeze_reset

Integrator with time constant T and delayed start based on parameter Tincfreeze and with reset

Functionality: Integrator with time constant T and delayed start based on parameter Tincfreeze
Option _incfreeze: blocks integrator at initialization and afterwards for a duration of Tincfreeze seconds.
Option _reset: The state "x" is reset to its initial value upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to initial value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Upper limitation input signals: rst
Continuous states: x
Parameters: T,Tincfreeze
Internal variables: t0,xinc

13.2 with reset (to input signal value)

13.2.1 1/s (p;p) _reset_sig

Integrator, state limited by parameters, with reset (to input signal value)

Functionality: This macro implements an integrator, state limited by parameters.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Lower limitation parameters: y_min
Upper limitation parameters: y_max

13.2.2 1/s (s;s) _enable_reset_sig

Integrator with signal limits (asymmetrical), hold signal, with reset (to input signal value)

Functionality: This macro implements an integrator with signal limits (asymmetrical)
Option _enable:
If hold signal >=0.5 then integrator is frozen
If hold signal < 0.5 then integrator is unfrozen
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,hold,xrst
Lower limitation input signals: y_min
Upper limitation input signals: rst,y_max
Continuous states: x

13.2.3 1/s [p;p] _reset_sig

Integrator with parameter limits (asymmetrical), with reset (to input signal value)

Functionality: This macro implements an integrator with parameter limits (asymmetrical)
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Lower limitation parameters: y_min
Upper limitation parameters: y_max

13.2.4 1/s [p] _reset_sig

Integrator with parameter limits (symmetrical), with reset (to input signal value)

Functionality: This macro implements an integrator with parameter limits (symmetrical)
Parameter y_max must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Upper limitation parameters: y_max

13.2.5 1/s [s;s] _reset_sig

Integrator with signal limits (asymmetrical), with reset (to input signal value)

Functionality: This macro implements an integrator with signal limits (asymmetrical)
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Lower limitation input signals: y_min
Upper limitation input signals: rst,y_max
Continuous states: x

13.2.6 1/s [s] _reset_sig

Integrator with signal limits (symmetrical), with reset (to input signal value)

Functionality: This macro implements an integrator with signal limits (symmetrical)
Input signal must always be positive in order to operate correctly.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst,y_max
Continuous states: x

13.2.7 1/s _enable_reset_sig

Integrator with reset (to input signal value)

Functionality: This macro implements a continuous time integrator block.
Option _enable:
If hold signal >=0.5 then integrator is frozen
If hold signal < 0.5 then integrator is unfrozen
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,hold,xrst
Upper limitation input signals: rst
Continuous states: x

13.2.8 1/s _incfreeze _reset_sig

Integrator with delayed start based on parameter Tincfreeze, with reset (to input signal value)

Functionality: Integrator with delayed start based on parameter Tincfreeze
Option _incfreeze: blocks integrator at initialization and afterwards for a duration of Tincfreeze seconds.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: Tincfreeze
Internal variables: t0

13.2.9 1/s _reset_sig

Integrator with reset (to input signal value)

Functionality: This macro implements a continuous time integrator block and reset.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{s}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x

13.2.10 1/sT _reset_sig

Integrator, time constant T, with reset (to input signal value)

Functionality: This macro implements an integrator block with time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x

Parameters: T

13.2.11 1/sT (p) _reset_sig

Integrator, state limits, symmetrical, time constant T, with reset (to input signal value)

Functionality: This macro implements a state limited integrator with time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.12 1/sT (p) _fb _reset_sig

Integrator, state limits, symmetrical, time constant T (fallback), with reset (to input signal valu

Functionality: This macro implements a state limited integrator with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.13 1/sT (p; _reset_sig

Integrator, state lower limit, time constant T, with reset (to input signal value)

Functionality: This macro implements an integrator, state lower limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min

13.2.14 1/sT (p; _fb_reset_sig

Integrator, state lower limit, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements a state lower limited integrator with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min

13.2.15 1/sT (p;p) _fb_reset_sig

Integrator, state limits, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements a state limited integrator with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min
Upper limitation parameters: y_max

13.2.16 1/sT (p;s) _fb_reset_sig

Integrator, state limits based on parameter/signal, time constant T (fallback), with reset

Functionality: This macro implements an integrator, state limits based on parameter/signal, time constant T (bypass)
Lower limits is parameter; upper limit is input signal
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst,y_max
Continuous states: x
Parameters: T
Lower limitation parameters: y_min

13.2.17 1/sT (pp;pp) _reset_sig

Integrator, parameter scaled limits, time constant T, with reset (to input signal value)

Functionality: This macro implements an integrator, parameter scaled limits, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T,K
Lower limitation parameters: y_min
Upper limitation parameters: y_max

13.2.18 1/sT (s;p) _fb_reset_sig

Integrator, state limits based on signal/parameter, time constant T (fallback) , with reset

Functionality: This macro implements an integrator, state limits based on signal/parameter, time constant T (bypass)
Lower limit is a signal; upper limit is a parameter.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Lower limitation input signals: y_min
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.19 1/sT (s;s) _fb_reset_sig

Integrator, state limits by signals, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements an integrator, state limits by signals, time constant T (bypass)
Lower and upper limits are input signals.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Lower limitation input signals: y_min
Upper limitation input signals: rst,y_max
Continuous states: x
Parameters: T

13.2.20 1/sT ;p) _fb_reset_sig

Integrator, state lower limit, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements an integrator, state lower limit, time constant T (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.21 1/sT ;p) _reset_sig

Integrator, state lower limit, time constant T, with reset (to input signal value)

Functionality: This macro implements an integrator, state lower limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.22 1/sT ;p] _fb_sig

Integrator, state upper limit, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements an integrator, state upper limit, time constant T (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.23 1/sT ;p] _reset_sig

Integrator, state upper limit, time constant T, with reset (to input signal value)

Functionality: This macro implements an integrator, state upper limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Upper limitation parameters: y_max

13.2.24 1/sT [p; _fb_reset_sig

Integrator, state lower limit, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements an integrator, state lower limit, time constant T (bypass)
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, then output keeps initial value.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min

13.2.25 1/sT [p; _reset_sig

Integrator, state lower limit, time constant T, with reset (to input signal value)

Functionality: This macro implements an integrator, state lower limit, time constant T
Parameter T must be positive. Otherwise, a message is printed to the output window.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T
Lower limitation parameters: y_min

13.2.26 1/sT _bypass_incfreeze_reset_sig

Integrator with time constant T and delayed start based on parameter Tincfreeze, with reset

Functionality: Integrator with time constant T and delayed start based on parameter Tincfreeze
Option _bypass: if time constant T<=0 then block is a feedthrough
Option _incfreeze: blocks integrator at initialization and afterwards for a duration of Tincfreeze seconds.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T,Tincfreeze
Internal variables: t0

13.2.27 1/sT _fb_reset_sig

Integrator, time constant T (fallback), with reset (to input signal value)

Functionality: This macro implements an integrator block with time constant T
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Option _fb: if T<=0, yo keeps its initial value
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T

13.2.28 1/sT _incfreeze_reset_sig

Integrator with time constant T and delayed start based on parameter Tincfreeze, with reset

Functionality: Integrator with time constant T and delayed start based on parameter Tincfreeze
Option _incfreeze: blocks integrator at initialization and afterwards for a duration of Tincfreeze seconds.
Option _reset_sig: The state "x" is reset to value of "xrst" input signal upon "rst" signal crossing 0.5 on rising flank.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1}{sT}$$

Macro location: *Macros\Integrators\with reset (to input signal value)*
Macro DSL level: *6*
Output signals: yo
Input signals: yi,xrst
Upper limitation input signals: rst
Continuous states: x
Parameters: T,Tincfreeze
Internal variables: t0

14 Lead-lag Blocks

This section provides a complete listing of the existing DSL macros within the *Lead-lag Blocks* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

14.1 (1+ATs)/(1+BTs)

Lead-lag block with three parameters

Functionality: Block that implements a continuous time lead-lag transfer function.
Uses three parameters: A, B and T.
Parameters T and B must be positive. Otherwise, a message is printed to the output window.

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + ATs}{1 + BTs} = \frac{A}{B}\frac{\frac{1}{AT} + s}{\frac{1}{BT} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: A,B,T
Internal variables: dx

14.2 (1+sTb)/(1+sTa)

Lead-lag block, time constants Ta and Tb

Functionality: This macro implements a lead-lag block using two parameters Ta and Tb.
Parameter Ta must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + sT_b}{1 + sT_a} = \frac{T_b}{T_a}\frac{\frac{1}{T_b} + s}{\frac{1}{T_a} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta
Internal variables: dx

14.3 (1+sTb)/(1+sTa) [(p;p)] _bypass

Lead-lag block, anti-windup limits (bypass)

Functionality: This macro implements a state limited lead-lag block with bypass.
The state variable and the output are limited using parameters "y_min" and "y_max".
Parameter Ta must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if Ta<=0 or Ta=Tb, block is a limited feed-through yo = yi between y_min and y_max limits.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1+sT_b}{1+sT_a} = \frac{T_b}{T_a}\frac{\frac{1}{T_b}+s}{\frac{1}{T_a}+s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: dx,yox

14.4 (1+sTb)/(1+sTa) [(p;p)] _fb

Lead-lag block, anti-windup limiter, time constant Ta (fallback value)

Functionality: This macro implements a first order lead-lag block with anti-windup limits.
Parameter Ta must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: If lag time constant Ta<=0 then use Ta=0.01.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1+sT_b}{1+sT_a} = \frac{T_b}{T_a}\frac{\frac{1}{T_b}+s}{\frac{1}{T_a}+s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: dx,yox

14.5 (1+sTb)/(1+sTa) [(pp;pp)] _bypass

Lead-lag block, anti-windup limiter, K dependent limits (bypass)

Functionality: This macro implements a lead-lag block with anti windup limits dependent on parameters K, y_min and y_max
Parameter Ta must be non-negative. Otherwise, a message is printed to the output window.
Parameter K must be positive. Otherwise a message is printed to the output window.
Bypass option: if Ta<=0 Ta=Tb, block is a limited feed-through yo = yi between y_min/K and y_max/K limits.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1 + sT_b}{1 + sT_a} = \frac{T_b}{T_a} \frac{\frac{1}{T_b} + s}{\frac{1}{T_a} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta,K
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: dx,yox

14.6 (1+sTb)/(1+sTa) _bypass

Lead-lag block (bypass)

Functionality: This macro implements a lead-lag block with bypass.
Bypass option:
If Ta>0 then block is a lead-lag
If Ta<=0 .or. Ta=Tb then block is bypassed (yo=yi)
Parameter Ta must be non-negative. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + sT_b}{1 + sT_a} = \frac{T_b}{T_a} \frac{\frac{1}{T_b} + s}{\frac{1}{T_a} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta
Internal variables: dx

14.7 (1+sTld)/(1+sTlg) and sx

Lead-lag block with lead time constant Tld and lag time constant Tlg

Lead-lag block (variant)
Tld - lead time constant
Tlg - lag time constant
If Tlg=0, then Tlg=0.01

This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + sT_{ld}}{1 + sT_{lg}} = \frac{T_{ld}}{T_{lg}} \frac{\frac{1}{T_{ld}} + s}{\frac{1}{T_{lg}} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tld,Tlg
Internal variables: dx

14.8 (1-ATs)/(1+sAT/2)

Lead-lag block, with gain and time constant

Functionality: This macro implements a lead-lag block with gain and time constant
Parameter A and T must be non-negative. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 + ATs}{1 + A\frac{T}{2}s} = 2\frac{\frac{1}{AT} - s}{\frac{2}{AT} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: A,T
Internal variables: dx,Tz

14.9 (1-sT)/(1+sT/2) _bypass

Lead-lag block variant, 1 parameter (bypass)

Functionality: This block implements a variant of a lead-lag block based on a single parameter T. it also includes a bypass in case T=0.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: If T<=0 then yo=yi (feedthrough block).
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1 - sT}{1 + sT/2} = 2\frac{\frac{1}{T} - s}{\frac{2}{T} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T

14.10 (K+sTb)/(1+sTa)

Lead-lag block variant, 3 parameters

Functionality: This macro implements a variant of a lead-lag using three parameters K, Ta and Tb.
Parameter Ta must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K + sT_b}{1 + sT_a} = \frac{T_b}{T_a}\frac{\frac{K}{T_b} + s}{\frac{1}{T_a} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,Tb,Ta
Internal variables: dx

14.11 (a+sbT)/(1+sT) _bypass

Lead-lag block variant, 3 parameters (bypass)

Functionality: This macro implements a lead-lag block using three parameters a, b and T.
Parameter T must be non-negative. Otherwise, a message is printed to the output window.
Bypass option: if T<=0, block is a gain yo=a*yi
This macro has a linear behaviour.

Function:

$$H(s) = \frac{a + sbT}{1 + sT} = b\frac{\frac{a}{bT} + s}{\frac{1}{T} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: a,b,T
Internal variables: dx

14.12 K(1+sTb)/(1+sTa)

Lead-lag block with gain, time constant Ta

Functionality: This macro implements a lead-lag block with gain, time constant Ta
Parameter Ta must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K(1+sT_b)}{1+sT_a} = \frac{T_a}{KT_b}\frac{\frac{1}{T_b}+s}{\frac{1}{T_a}+s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,Tb,Ta
Internal variables: dx

14.13 K(1+sTld)/(1+sTlg) _fb

Lead-lag block with gain, time constant Tlg (fallback value)

Functionality: This macro implements a lead-lag block with gain, time constant Tlg (fallback value)
Parameter Tlg must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: If Tlg<=0, then use Tlg=0.01
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K(1+sT_{ld})}{1+sT_{lg}} = \frac{T_{lg}}{KT_{ld}}\frac{\frac{1}{T_{ld}}+s}{\frac{1}{T_{lg}}+s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x

Parameters: K,Tld,Tlg
Internal variables: dx

14.14 K(A1+sT1)/(A2+sT2)

General first order lead-lag block

Functionality: This macro implements a general first order lead-lag block, 5 parameters
Parameter T2 must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K(A_1 + sT_1)}{A_2 + sT_2} = K\frac{T_1}{T_2}\frac{\frac{A_1}{T_1} + s}{\frac{A_2}{T_2} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,A1,T1,A2,T2
Internal variables: dx

14.15 K(A1+sT1)/(A2+sT2) [(p;p)] _fb

General first order lead lag block with state and output limits, time constant T2 (fallback value)

Functionality: This macro implements a general first order lead lag block with state limits, time constant T2 (fallback value)
Parameters T1 and T2 must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: if T2<=0 then use T2=0.01
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{K(A_1 + sT_1)}{A_2 + sT_2} = K\frac{T_1}{T_2}\frac{\frac{A_1}{T_1} + s}{\frac{A_2}{T_2} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,A1,T1,A2,T2
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: dx

14.16 K(A1+sT1)/(A2+sT2) _fb

General first order lead-lag block (fallback value)

Functionality: This macro implements a general first order lead-lag block with 5 parameters.
Parameter T2 must be non-negative. Otherwise, a message is printed to the output window.
Fallback option: If T2<=0, then use T2=0.01
This macro has a linear behaviour.

Function:

$$H(s) = \frac{K(A_1 + sT_1)}{A_2 + sT_2} = K\frac{T_1}{T_2}\frac{\frac{A_1}{T_1} + s}{\frac{A_2}{T_2} + s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,A1,T1,A2,T2
Internal variables: dx

14.17 a23(1+(a11-a13a21/a23)sTw)/(1+a11sTw)

a23(1+(a11-a13a21/a23)sTw)/(1+a11sTw)

Functionality: This block implements the first order transfer function H(s)=a23(1+(a11-a13a21/a23)sTw)/(1+a11sTw)
Parameter Tw must be positive. Otherwise, a message is printed to the output window.
Parameter a11 must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = a_{23}\frac{1 + (a_{11} - a_{13}a_{21}/a_{23})T_w s}{1 + a_{11}T_w s}$$

Macro location: *Macros\Lead-lag Blocks*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: a11,a13,a21,a23,Tw
Internal variables: dx,Ta,Tb

15 Limiters

This section provides a complete listing of the existing DSL macros within the *Limiters* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

15.1 Limit ;p] _eps

Limiter (upper) with small threshold

Functionality: This block implements an upper limiter with a small threshold upon switching between the linear and the limitted regions.
This block avoids toggling behaviour when the input signal is at one of the limit values.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
If set=rst=1 (at within deadband condition) then output is undefined.
If eps=0, then this block is an upper limiter without switching threshold.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: eps
Upper limitation parameters: y_max
Internal variables: setmax,rstmax,yo_max

15.2 Limit [p; _eps

Limiter (lower) with small threshold

Functionality: This block implements a lower limiter with a small threshold upon switching between the linear and the limitted regions.
This block avoids toggling behaviour when the input signal is at one of the limit values.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
If set=rst=1 (at within deadband condition) then output is undefined.
If eps=0, then this block is a lower limiter without switching threshold.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: eps
Lower limitation parameters: y_min
Internal variables: setmin,rstmin,yo_min

15.3 Limit [p;p]

Limiter (lower/upper, asymmetric)

Functionality: This block implements an asymmetric limit (upper/lower)
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

Lower limitation parameters: y_min
Upper limitation parameters: y_max

15.4 Limit [p;p] _eps

Limiter (lower/upper,asymmetric) with small threshold

Functionality: This block implements a symmetric limiter with a small threshold upon switching between the linear and the limitted regions.
This block avoids toggling behaviour when the input signal is at one of the limit values.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
If set=rst=1 (at within deadband condition) then output is undefined.
If eps=0, then this block is an asymmetric limiter without switching threshold.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: eps
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: setmin,rstmin,setmax,rstmax,yo_min,yo_max

15.5 Limit [p]

Limiter (lower/upper,symmetric)

Functionality: This block implements a symmetric limit.
Parameter y_lim must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation parameters: y_lim

15.6 Limit [p] (using min/max)

Limiter (lower/upper,symmetric)

Functionality: This block implements a symmetric limiter.
Parameter y_lim must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*

Output signals: yo
Input signals: yi
Upper limitation parameters: y_lim

15.7 Limit [p] _eps

Limiter (lower/upper,symmetric) with small threshold

Functionality: This block implements a symmetric limiter with a small threshold upon switching between the linear and the limitted regions.
This block avoids toggling behaviour when the input signal is at one of the limit values.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
Parameter y_lim must be positive. Otherwise a message is printed to the output window.
If set=rst=1 (at within deadband condition) then output is undefined.
If eps=0, then this block is a symmetric limiter without switching threshold.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: eps
Upper limitation parameters: y_lim
Internal variables: setmin,rstmin,setmax,rstmax,yo_min,yo_max

15.8 Limit [s;s]

Limiter (lower/upper, asymmetric with signals)

Functionality: This block implements an asymmetric limiter with signals (upper/lower)
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: y_max

15.9 Limit [s;s] _eps

Limiter (lower/upper,asymmetric) with small threshold

Functionality: This block implements a symmetric limiter with a small threshold upon switching between the linear and the limitted regions.
This block avoids toggling behaviour when the input signal is at one of the limit values.
The threshold is defined by parameter eps and it is usually set to a small user defined value.
If set=rst=1 (at within deadband condition) then output is undefined.

If eps=0, then this block is an asymmetric limiter without switching threshold.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation input signals: y_min
Upper limitation input signals: y_max
Parameters: eps
Internal variables: setmin,rstmin,setmax,rstmax,yo_min,yo_max

15.10 Limit [sp;sp]

Limiter (lower/upper, asymmetric with signals)

Functionality: This block implements an asymmetric limiter with signals (upper/lower) and parameters (upper/lower).
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation input signals: yi_min
Upper limitation input signals: yi_max
Lower limitation parameters: y_min
Upper limitation parameters: y_max

15.11 Limit lower [p;

Limiter (lower)

Functionality: This block implements a lower limit.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation parameters: y_min

15.12 Limit upper ;p]

Limiter (upper)

Functionality: This block implements an upper limit.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation parameters: y_max

15.13 Rate limiter ;p}

Gradient limiter (upper)

Functionality: This block implements a gradient limiter (upper)
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation parameters: grd_up

15.14 Rate limiter base ;p}

Gradient limiter with base (upper)

Functionality: This block implements an upper gradient limiter block
Parameters:
base - the base value of the input signal, this leads to a gradient limitation in pu/s
grd_up - upper gradient parameter, must be positive, in p.u., base value = base
Parameter hi must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: base
Upper limitation parameters: grd_up

15.15 Rate limiter base {p;

Gradient limiter with base (lower)

Functionality: This block implements a lower gradient limiter block
Parameters:
base - the base value of the input signal, this leads to a gradient limitation in pu/s
grd_down - lower gradient parameter, in p.u., base value = base

This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: base
Lower limitation parameters: grd_down

15.16 Rate limiter base {p;p}

Gradient limiter with base (lower/upper, asymmetric)

Functionality: This block implements a gradient limiter (asymmetrical) using parameters lo and hi
Parameters:
base - the base value of the input signal, this leads to a gradient limitation in pu/s
lo - lower gradient parameter in p.u., base value = base
hi - upper gradient parameter in p.u., base value = base, must be positive
Parameter hi must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: base
Lower limitation parameters: grd_down
Upper limitation parameters: grd_up

15.17 Rate limiter base {p}

Gradient limiter with base (lower/upper,symmetric)

Functionality: This block implements a gradient limiter block (upper/lower), symmetric, in per-unit
Parameters:
base - the base value of the input signal, this leads to a gradient limitation in pu/s
grd - gradient parameter, in p.u., base value = base, must be positive
Parameter grd must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: base
Upper limitation parameters: grd

15.18 Rate limiter {p;

Gradient limiter (lower)

Functionality: This block implements a gradient limiter (lower)
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation parameters: grd_down

15.19 Rate limiter {p;p}

Gradient limiter (lower/upper, asymmetric)

Functionality: This block implements a gradient limiter (lower/upper, asymmetric)
Parameter grd_up must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Lower limitation parameters: grd_down
Upper limitation parameters: grd_up

15.20 Rate limiter {p}

Gradient limiter (lower/upper,symmetric)

Functionality: This block implements a gradient limiter (lower/upper,symmetric)
Parameter grd must be positive. Otherwise a message is printed to the output window.
This macro has a non-linear behaviour.

Macro location: *Macros\Limiters*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation parameters: grd

16 Logic Functions

This section provides a complete listing of the existing DSL macros within the *Logic Functions* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

16.1 Basic Logic Functions

16.1.1 AND2

Logic function AND with two inputs (boolean)

Functionality: Implements the logic function AND with two inputs.
yi1 and yi2 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yo
false	false	false
true	false	false
false	true	false
true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

16.1.2 AND3

Logic function AND with three inputs (boolean)

Functionality: Implements the logic function AND with three inputs.
yi1, yi2 and yi3 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yi3	yo
false	false	false	false
true	false	false	false
false	true	false	false
true	true	false	false
false	false	true	false
true	false	true	false
false	true	true	false
true	true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3

16.1.3 AND4

Logic function AND with four inputs (boolean)

Functionality: Implements the logic function AND with four inputs.
yi1, yi2, yi3 and yi4 are assumed to be boolean input values
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3,yi4

16.1.4 EOR

Logic function EOR (NOT EQUAL) (boolean)

Functionality: Implements the logic function EOR (NOT EQUAL) with two inputs
yi1 and yi2 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yo
false	false	false
true	false	true
false	true	true
true	true	false

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

16.1.5 EQUAL

Logic function EQUAL (NOT EOR) (boolean)

Functionality: Implements the logic function EQUAL
Description: Returns 1 if yi1 is equal with yi2, 0 otherwise
Behaviour identical with negated exclusive OR

yi1 and yi2 are assumed boolean inputs
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yo
false	false	true
true	false	false
false	true	false
true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

16.1.6 NOR

Logic function NOR (boolean)

Functionality: Implements the logic function NOR.
yi1 and yi2 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yo
false	false	true
true	false	false
false	true	false
true	true	false

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

16.1.7 NOT

Logic function NOT (boolean)

Functionality: Implements the logic function NOT.
yi is assumed to be a boolean input value
false =(-oo,0.5)
true =[0.5, oo)

Truth table
----
yi	yo
false	true
true	false

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

16.1.8 OR2

Logic function OR with two inputs (boolean)

Functionality: Implements the logic function OR with two inputs.
yi1 and yi2 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yo
false	false	false
true	false	true
false	true	true
true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

16.1.9 OR3

Logic function OR with three inputs (boolean)

Functionality: Implements the logic function OR with three inputs.
yi1, yi2 and yi3 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yi3	yo
false	false	false	false

true	false	false	true
false	true	false	true
true	true	false	true
false	false	true	true
true	false	true	true
false	true	true	true
true	true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3

16.1.10 OR4

Logic function OR with four inputs (boolean)

Functionality: Implements the logic function OR with four inputs.
yi1, yi2, yi3 and yi4 are assumed to be boolean input values
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Basic Logic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3,yi4

16.2 Logic Functions with Pick-up/Drop-off

16.2.1 2_out_of_3 _ip

Output equals to 1 if two inputs are true

Functionality: Output equals to 1 if two inputs are true
Logic function that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3
Parameters: Tpick,Tdrop

16.2.2 AND2 _ip

Logic AND function applied on two inputs (boolean, picdro implementation)

Functionality: Logic AND function applied on two inputs
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
yi1 and yi2 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table

```
-------------------------
| yi1 | yi2 | yo |
|----------------------|
| false | false | false |
| true | false | false |
| false | true | false |
| true | true | true |
-------------------------
```

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

16.2.3 AND3 _ip

Logic function AND applied to three inputs (boolean, picdro implementation)

Functionality: Implements the logic function AND applied to three inputs
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
yi1, yi2 and yi3 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table

```
---------------------------------
| yi1 | yi2 | yi3 | yo |
|----------------------|-------|
| false | false | false | false |
| true | false | false | false |
| false | true | false | false |
| true | true | false | false |
| false | false | true | false |
| true | false | true | false |
| false | true | true | false |
| true | true | true | true |
---------------------------------
```

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3
Parameters: Tpick,Tdrop

16.2.4 Invert Logic _ip

Invert a logical signal (boolean, picdro implementation)

Functionality: Invert a logical signal
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: Tpick,Tdrop

16.2.5 NOT _ip

Logic NOT function (boolean, picdro implementation)

Functionality: Implements the logic NOT function
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
yi is assumed to be a boolean input value
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi	yo
false	true
true	false

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1
Parameters: Tpick,Tdrop

16.2.6 OR2 _ip

Logic OR function applied on two inputs (boolean, picdro implementation)

Functionality: Implements the logic OR function applied on two inputs (delay)
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
yi1 and yi2 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yo

false	false	false
true	false	true
false	true	true
true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: Tpick,Tdrop

16.2.7 OR3 _ip

Logic OR function applied on three inputs (boolean, picdro implementation)

Functionality: Implements the logic OR function applied on three inputs
Logic function based on picdro() that avoids toggling effects but introduces step re-evaluation.
yi1, yi2 and yi3 are assumed to be boolean input values
false =(-oo,0.5)
true =[0.5, oo)
Truth table
----
yi1	yi2	yi3	yo
false	false	false	false
true	false	false	true
false	true	false	true
true	true	false	true
false	false	true	true
true	false	true	true
false	true	true	true
true	true	true	true

This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Logic Functions with Pick-up/Drop-off*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3
Parameters: Tpick,Tdrop

16.3 Special Logic Functions

16.3.1 Bistable

Prioritized S-R flip-flop (bistable multivibrator)

Functionality: Prioritized S-R flip-flop (bistable multivibrator)
The Bistable macro operates as below:

When S input is true and R input is false, the flip-flop goes to the Set state. This is the first stable state, where Q is true.
When R is true and S is false, the flip-flop goes to the reset state. This is the second stable state, where Q is false.
When both S and R are true, the flip-flop goes to the prioritized state defined by the Select priority parameter.
When both S and R are false, the flip-flop stays in its previous state.
vardef(PRIO)='0/1';'Prioritised state: 1=SET, 0=RESET'
PRIO_SET=1: If S=R=1 then Q=1
PRIO_SET=0: If S=R=1 then Q=0
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Special Logic Functions*
Macro DSL level: *6*
Output signals: Q,not_Q
Input signals: S,R
Parameters: PRIO_SET

16.3.2 Edge detector

Outputs an impulse when input changes

Functionality: The Edge Detector block outputs an impulse (the value of 1 for one time step) when the logical input changes
If EDGE< 0.5 (e.g. EDGE=0 (default value)) then the detection occurs on the rising edge (from FALSE (yi<0.5) to TRUE (yi>=0.5)).
If 0.5<=EDGE< 1.5 (e.g. EDGE=1) then the detection occurs on the falling edge (from TRUE (yi>=0.5) to FALSE (yi<0.5)).
If EDGE>=1.5 (e.g. EDGE=2) then the detection occurs on either edges.
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Special Logic Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Parameters: EDGE
Internal variables: rise,fall

16.3.3 Monostable

Monostable flip-flop (one-shot multivibrator)

Functionality: Monostable/monoflop (one-shot) - it generates a pulse of pre-defined duration when triggered (output yo is set to 1). When the timer has expired, it returns to its stable state and produces no more output until triggered again.
The output is set to TRUE (yo = 1 i.e. switches on) upon a rising edge of the input (i.e. yi>0.5) if the monostable output at the previous time step was FALSE (yo=0). Upon triggering, the monostable's output is held TRUE (yo = 1) for a time duration of T seconds. The Monostable block ignores any edge occurring when the pulse is TRUE. Upon time period T expiry since triggering, the output is set to FALSE (yo = 0, i.e. switching off).
This macro has a non-linear behaviour.

Macro location: *Macros\Logic Functions\Special Logic Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: T
Internal variables: set,rst,t0

17 Math Functions

This section provides a complete listing of the existing DSL macros within the *Math Functions* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

17.1 Basic Functions

17.1.1 ABS

Absolute value of real input

Functionality: Returns the absolute value of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.2 CEIL

Round input to upper integer value

Functionality: Returns the ceil() value of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.3 EXP

Exponential function

Functionality: Computes the exponential function of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.4 FLOOR

Round input to lower integer value

Functionality: Computes the floor function of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.5 FRAC

Extraction of fractional part of input

Functionality: Retrieves the fractional part of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.6 LN

Natural logarithm of input

Functionality: Computes the natural logarithm of the input.
The domain of the function is yi>0.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.7 LOG

Logarithm to base of 10

Functionality: Computes the base 10 logarithm of the input.
The domain of the function is yi>0.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.8 MAX2

Maximum function (of two signals)

Functionality: Computes the maximum between two input signals.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

17.1.9 MAX3

Maximum function (of three signals)

Functionality: Computes the maximum between three input signals.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3

17.1.10 MAX4

Maximum function (of four signals)

Functionality: Computes the maximum between four input signals.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3,yi4

17.1.11 MIN2

Minimum function (of two signals)

Functionality: Computes the minimum between two input signals.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2

17.1.12 MIN3

Minimum function (of three signals)

Functionality: Computes the minimum between three input signals.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3

17.1.13 MIN4

Minimum function (of four signals)

Functionality: Computes the minimum between four input signals.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3,yi4

17.1.14 MODULO

Remainder after numerical division of input by parameter n

Functionality: Computes the remainder after numerical division of yi by n
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: n

17.1.15 RECIPROCAL

Reciprocal function: returns the inverse of input u

Functionality: Returns the inverse of input u
This macro has a linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: u

17.1.16 RECIPROCAL _fb

Reciprocal function: returns the inverse of input u (fallback)

Functionality: Returns the inverse of input u
Option _fb: If abs(u)<0.000001 then output returns 1/0.000001
This macro has a linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: u

17.1.17 ROUND

Rounds input to nearest integer value

Functionality: Round input to nearest integer value.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.18 SIGN

Determines sign of input

Functionality: Determines the sign of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.19 SQRT

Square root of input

Functionality: Computes the square root of the input.
Note: input may not be negative. If input negative then the absolute value is taken in order to avoid error.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.20 xˆ2

Square function

Functionality: Computes the square of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.1.21 xˆp

Input to the power of p

Functionality: Computes the input to the power of p
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Basic Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: p

17.2 Complex Operations

17.2.1 ADD COMPLEX

Addition of two complex quantities

Functionality: Addition of two complex quantities defined by (re1,im1) and (re2,im2).
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: re,im
Input signals: re1,im1,re2,im2

17.2.2 CONJ COMPLEX

Compute the complex conjugate

Functionality: Retrieve the complex conjugate of a complex number (re1,im1).
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: re,im
Input signals: re1,im1

17.2.3 DIV COMPLEX

Division of two complex quantities

Functionality: Division of two complex quantities defined by (re1,im1) and (re2,im2).
Returns: real an imaginary components of the complex division (re1,im1)/(re2,im2).
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: re,im
Input signals: re1,im1,re2,im2
Internal variables: var1

17.2.4 MAG COMPLEX

Magnitude of complex quantity

Functionality: Retrieves the magnitude of a complex quantity with real and imaginary components 're' and 'im'.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: mag
Input signals: re,im

17.2.5 MUL COMPLEX

Multiplication of two complex quantities

Functionality: Multiplication of two complex quantities defined by (re1,im1) and (re2,im2).
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: re,im
Input signals: re1,im1,re2,im2

17.2.6 SUB COMPLEX

Subtraction of two complex quantities

Functionality: Subtraction of two complex quantities defined by (re1,im1) and (re2,im2).
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: re,im
Input signals: re1,im1,re2,im2

17.2.7 TO POLAR COMPLEX

Compute the polar form of a complex number from rectangular form

Functionality: Compute the polar form of a complex number from rectangular form.
Output phi is in radians.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: mag,phi
Input signals: re,im

17.2.8 TO RECTANGULAR COMPLEX

Compute the rectangular form of a complex number from polar form

Functionality: Compute the rectangular form of a complex number from polar form.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Complex Operations*
Macro DSL level: *5*
Output signals: re,im
Input signals: mag,phi

17.3 Trigonometric Functions

17.3.1 ACOS

Arccosine of input

Functionality: Returns the arccosine of the input.
The output yo is in radians.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.2 ASIN

Arcsine of input

Functionality: Returns the arcsine of the input.
The input yi is in radians.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.3 ATAN

Arctangent of input

Functionality: Returns the arctangent of the input.
Output yo is expressed in radians
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.4 ATAN2

Four-quadrant arctangent

Functionality: Returns the principal value of the angle, given in radians, between the Euclidean plane positive x-axis and the ray to the point (re,im) of the complex value defined by inputs "re" and "im".

Output is in radians, ranging between (-pi,pi].
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: re,im

17.3.5 ATAN2D

Four-quadrant arctangent (degrees)

Functionality: Returns the principal value of the angle, given in degrees, between the Euclidean plane positive x-axis and the ray to the point (re,im) of the complex value defined by inputs "re" and "im".
Output is in degrees, ranging between (-180,180].
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: re,im

17.3.6 ATAND

Arctangent of input (output angle in degrees)

Functionality: Returns the arctangent of the input (input is the tangent of angle).
Output yo is expressed in degrees
This macro has a linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: yi

17.3.7 COS

Cosine of input

Functionality: Returns the cosine of the input.
Input yi is in radians.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.8 COSD

Cosine of input (degrees)

Functionality: Computes the cosine of the input (expressed in degrees).
This macro has a linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: yi

17.3.9 COSH

Hyperbolic cosine of input

Functionality: Returns the cosine hyperbolic of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.10 COSHD

Hyperbolic cosine of input (degrees)

Functionality: Returns the cosine hyperbolic of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.11 SIN

Sine of input

Functionality: Computes the sine of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.12 SIND

Sine of input (degrees)

Functionality: Computes the sine of the input (expressed in degrees).
This macro has a linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: yi

17.3.13 SINH

Hyperbolic sine of input

Functionality: Computes the sine hyperbolic of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.14 SINHD

Hyperbolic sine of input (degrees)

Functionality: Computes the sine hyperbolic of the input (expressed in degrees).
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.15 TAN

Tangent of input

Functionality: Computes the tangent of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.16 TAND

Tangent of angle (input angle in degrees)

Functionality: Returns the tangent of an angle yi.
Input yi is expressed in degrees.
This macro has a linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *6*
Output signals: yo
Input signals: yi

17.3.17 TANH

Hyperbolic tangent of input

Functionality: Computes the tangent hyperbolic of the input.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

17.3.18 TANHD

Hyperbolic tangent of input (degrees)

Functionality: Computes the tangent hyperbolic of the input.
Input yi is expressed in degrees.
This macro has a non-linear behaviour.

Macro location: *Macros\Math Functions\Trigonometric Functions*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

18 Mechanical

This section provides a complete listing of the existing DSL macros within the *Mechanical* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

18.1 1/(2Hs)

Inertia H

This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: H

18.2 Accelerating Power (simple)

Computation of accelerating power (simple)

Functionality: This macro computes the accelerating power
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: yo
Input signals: speed,xmt,xme

18.3 Accelerating Power IPB

Computation of accelerating power (variant)

Functionality: This macro calculates the accellerating power of a machine in either generator MVA base
(IPB=1) or in generator MW base (IPB=0)

This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: Pa
Input signals: cosn,speed,xmt,xme
Parameters: IPB

18.4 Gear Box

Gear Box

This macro has a linear behaviour.

Macro location: *Macros\Mechanical*

Macro DSL level: *5*
Output signals: omega
Input signals: speed
Parameters: Nratio,RPM_syn

18.5 Mass_J

Mass_J

This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: omega
Input signals: M1,M2
Continuous states: xomega
Parameters: J

18.6 P/omg -> Torque

Calculation of torque based on power and speed

Functionality: This macro computes torque based on speed and power.
Speed input ’omega’ is expressed in rad/s.
’Power’ input is expressed in kW.
’Torque’ output is expressed in Nm.
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: Torque
Input signals: Power,omega

18.7 Pt/Pturb

Convert p.u. turbine power in generator MW base (for synchronous generators)

Functionality: This block converts the turbine power from the base defined by parameter PN to the generator MW base. If PN=0, no conversion is perfromed
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: pt
Input signals: pturb,sgnn,cosn
Parameters: PN

18.8 Shaft J-k and Pin

Single mass p.u. shaft model with one coupling and input power

Functionality: This block implements a single mass p.u. shaft model with one coupling and input power
Nominal torque Tnom = Snom/(2*pi*fnom)

Initial conditions:
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *6*
Output signals: omega_j,torque_jk
Input signals: omega_k,Pin
Continuous states: xdtheta_jk,xomega_j
Parameters: K_jk,D_jk,D_jj,H_j,fnom
Internal variables: T_j,Td_jj,Td_jk,Tin,dtheta_jk,dw_j,dw_jk,o1,o2,o3,yi

18.9 Shaft i-J

Single mass p.u. shaft model with one coupling, no input power

Functionality: This block implements a single mass p.u. shaft model with one coupling, no input power
Nominal torque Tnom = Snom/(2*pi*fnom)

Initial conditions:
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *6*
Output signals: torque_ji
Input signals: omega_i
Continuous states: xdtheta_ji,xomega_j
Parameters: K_ji,D_ji,D_jj,H_j,fnom
Internal variables: T,Td_ji,Td_jj,dtheta_ji_deg,dw_ji,dwj,fnom_,o3,omega_j,twopi,yi

18.10 Shaft i-J-k

Single mass p.u. shaft model with two couplings, no input power

Functionality: This block implements a single mass p.u. shaft model with two couplings, no input power
Nominal torque Tnom = Snom/(2*pi*fnom)

Initial conditions:
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *6*

Output signals: omega_j,torque_jk
Input signals: omega_k,omega_i,torque_ij
Continuous states: xdtheta_jk,xomega_j
Parameters: K_jk,D_jk,D_ij,D_jj,H_j,fnom
Internal variables: T,T13,Td12,Td2,add,dtheta_deg,dw2,dw_jk,dw_ki,o1,o2,o3,yi

18.11 Shaft i-J-k and Pin

Single mass p.u. shaft model with two couplings and input power

Functionality: This block implements a single mass p.u. shaft model with two couplings and input power
Nominal torque Tnom = Snom/(2*pi*fnom)

Initial conditions:
This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: omega_j,torque_jk
Input signals: omega_k,omega_i,torque_ij,Pin
Continuous states: xdtheta_jk,xomega_j
Parameters: K_jk,D_jk,D_ij,D_jj,H_j,fnom
Internal variables: T,T1,T13,Td12,Td2,add,dtheta_deg,dw2,dw_jk,dw_ki,o1,o2,o3,yi

18.12 Spring

Spring

This macro has a linear behaviour.

Macro location: *Macros\Mechanical*
Macro DSL level: *5*
Output signals: M
Input signals: omega1,omega2
Continuous states: xphi
Parameters: K,D
Internal variables: dxphi

19 PI(D) Controllers

This section provides a complete listing of the existing DSL macros within the *PI(D) Controllers* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

19.1 (1+sT)/KsT [p;p]

PI controller, limited output

Functionality: This macro implements a PI controller with limited output
Proportional gain is 1/K.
Integral gain is 1/(K*T).
The output is limited using two parameters "y_min" and "y_max". Anti-windup limits are included.
Parameters T and K must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1+sT}{KsT} = \frac{1}{K}\frac{\frac{1}{T}+s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T
Lower limitation parameters: y_min
Upper limitation parameters: y_max

19.2 (1+sT)/KsT {p}[(p;p)]

PI controller, rate limit (symmetrical) and anti-windup limiter

Functionality: PI controller with a symmetrical rate limiter and an upper/lower output limitation with two parameters (y_min and y_max)
The rate limiter parameter "ylim" defines the derivative limit.
The upper/lower output limitation is defined using two parameters "y_min" and "y_max".
Parameters T and K must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1+sT}{KsT} = \frac{1}{K}\frac{\frac{1}{T}+s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T
Lower limitation parameters: y_min
Upper limitation parameters: ylim,y_max
Internal variables: dx

19.3 (1+sTb)/sTa

PI controller (variant)

Functionality: This macro implements a variant of a PI controller, using two parameters Ta and Tb.
Gain: Tb/Ta
Parameter Ta must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = \frac{1+sT_b}{sT_a} = \frac{T_b}{T_a}\frac{\frac{1}{T_b}+s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Tb,Ta
Internal variables: dx

19.4 (1+sTp)/sTi ;p)]

PI controller with upper output limit and upper anti-windup

Functionality: This macro implements a variant of a PI controller, using two parameters Tp and Ti.
An upper limitation is included
Gain: Tp/Ti
Parameter Ti must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = \frac{1+sTp}{sT_i} = \frac{T_p}{T_i}\frac{\frac{1}{T_p}+s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Ti,Tp
Upper limitation parameters: y_max
Internal variables: dx

19.5 1+K/sT

PI controller with unity proportional gain, integral gain and time constant

Functionality: This macro implements a variant of a PI controller with unity proportional gain, integral gain and time constant
Parameter T must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = 1 + K\frac{1}{sT} = \frac{\frac{K}{T} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: K,T

19.6 Kp(1/Ti+s)/s

PI controller, series variant, gain Kp, time constant Ti

Functionality: This block implements a PI controller, series variant.
Proportional gain: Kp
Integral time constant: Ti
This macro has a linear behaviour.

Function:

$$H(s) = K_p\frac{1/T_i + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Kp,Ti

19.7 Kp(1/Ti+s)/s (s)

PI controller, series variant, gain Kp, time constant Ti, with anti-windup

Functionality: This block implements a PI controller, series variant with anti-windup.
The anti-windup implementation is based on the backfeed of the saturated output (external signal) and the unsaturated output (internal).
The difference between these two is used as a second term fed to the integrator input.
Proportional gain: Kp

Integral time constant: Ti
Time constant Tt determines how quickly the integrator is reset.
This macro has a non-linear behaviour.

Function:

$$H(s) = K_p \frac{1/T_i + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: yo_lim
Continuous states: x
Parameters: Kp,Ti,Tt

19.8 Kp+1/sTi

PI Controller, parallel variant, proportional gain Kp, integral time constant Ti

Functionality: This macro implements a variant of a PI Controller (parallel variant).
Proportional gain: Kp
Integral time constant: Ti
Parameter Ti must be positive. Otherwise, a message is printed to the output window.
This macro has a linear behaviour.

Function:

$$H(s) = K_p + \frac{1}{sT_i} = K_p \frac{\frac{1}{K_p T_i} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Kp,Ti

19.9 Kp+1/sTi [(p;p)]

PI controller, parallel variant, with anti-windup limiter and integrator time constant

Functionality: This macro implements a variant of a PI Controller (parallel variant).
Proportional gain: Kp
Integral time constant: Ti
Parameter Ti must be positive. Otherwise, a message is printed to the output window.
This macro has a non-linear behaviour.

Function:

$$H(s) = K_p + \frac{1}{sT_i} = K_p \frac{\frac{1}{K_p T_i} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Kp,Ti
Lower limitation parameters: y_min
Upper limitation parameters: y_max

19.10 Kp+Ki/s

PI controller, parallel variant, proportional and integral gains

Functionality: This block implements a PI controller, parallel variant.
Proportional gain: Kp
Integral gain: Ki
This macro has a linear behaviour.

Function:

$$H(s) = K_p + \frac{K_i}{s} = K_p \frac{\frac{K_i}{K_p} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Kp,Ki

19.11 Kp+Ki/s (s)

PI controller, parallel variant, proportional and integral gains, ext. signal tracking anti-windup

Functionality: This block implements a PI controller, parallel variant with external signal tracking anti-windup.
The anti-windup implementation is based on the backfeed of the saturated output (external signal) and the unsaturated output (internal).
Note: Signal yo_lim is the saturated output (not the limit itself) obtained by externally limiting the signal yo.
The difference between these two is used as a second term fed to the integrator input.
Proportional gain: Kp
Integral gain: Ki
Time constant Tt determines how quickly the integrator state is frozen.
This macro has a non-linear behaviour.

Function:

$$H(s) = K_p + \frac{K_i}{s} = K_p \frac{\frac{K_i}{K_p} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: yo_lim
Continuous states: x
Parameters: Kp,Ki,Tt

19.12 Kp+Ki/s [(p;p)]

PI controller, parallel variant, with Kp, Ki gains and anti-windup limits

Functionality: This block implements a Proportional-Integral (PI) block (parallel variant) with anti-windup limits.
Proportional gain: Kp
Integral gain: Ki
This macro has a non-linear behaviour.

Function:

$$H(s) = K_p + \frac{K_i}{s} = K_p \frac{\frac{K_i}{K_p} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Kp,Ki
Lower limitation parameters: y_min
Upper limitation parameters: y_max

19.13 Kp+Ki/s [p;p](pv;pv)

PI Controller, parallel variant, with anti-windup acc. to IEEE 421.5 (parallel form)

Functionality: This macro implements a PI Controller with anti-windup acc. to IEEE 421.5 (parallel form), in which the integrator state variable is frozen if the output reaches the limits.
Note: Kp = 0 is not supported, due to risk of integrator latching to limit.
vardef(Kp) = 'p.u.';'Proportional gain'
vardef(Ki) = 'p.u.';'Integral gain'

This macro has a non-linear behaviour.

Function:

$$H(s) = K_p + \frac{K_i}{s} = K_p \frac{\frac{K_i}{K_p} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x
Parameters: Kp,Ki
Lower limitation parameters: ymin
Upper limitation parameters: ymax

19.14 Kp+Ki/s [s;s](sv;sv)

PI Controller, parallel variant, with anti-windup acc. to IEEE 421.5 (parallel form, variable limit)

Functionality: This macro implements a PI Controller with anti-windup acc. to IEEE 421.5 (parallel form), in which the integrator state variable is frozen if the output reaches the limits.
Limits ymin and ymax are variable and provided as input signals.
Note: Kp = 0 is not supported, due to risk of integrator latching to limit.
vardef(Kp) = 'p.u.';'Proportional gain'
vardef(Ki) = 'p.u.';'Integral gain'

This macro has a non-linear behaviour.

Function:

$$H(s) = K_p + \frac{K_i}{s} = K_p \frac{\frac{K_i}{K_p} + s}{s}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *6*
Output signals: yo
Input signals: yi
Lower limitation input signals: ymin
Upper limitation input signals: ymax
Continuous states: x
Parameters: Kp,Ki

19.15 Kp+Ki/s+sKd/(1+sTd) [(p;p)]

PID controller,parallel variant, with anti-windup and limits

Functionality: This block implements a PID controller with anti-windup limit and output limits based on the same parameters.
Proportional gain: Kp
Integral gain: Ki
Derivative gain: Kd

Derivative term time constant: Td
This macro has a non-linear behaviour.

Function:

$$H(s) = K_p + \frac{sK_d}{1+sT_d} + \frac{K_i}{s} = (K_p + \frac{K_d}{T_d})\frac{\frac{K_i}{K_pT_d+K_d} + \frac{K_p+K_iT_d}{K_pT_d+K_d}s + s^2}{\frac{1}{T_d}s + s^2}$$

Macro location: *Macros\PI(D) Controllers*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Continuous states: x1,x2
Parameters: Kp,Ki,Kd,Td
Lower limitation parameters: y_min
Upper limitation parameters: y_max
Internal variables: yk,yis,yd,dx2

20 Signals

This section provides a complete listing of the existing DSL macros within the *Signals* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

20.1 Clock (t>0) _par

Clock signal (square wave) of fixed frequency at t>0

Functionality: Outputs a square wave signal with frequency cFreq [Hz].
The "on" pulse has the half width of the period of the signal (Ton/Tp = 0.5).
The clock frequency can be changed via parameter "cFreq".
The waveform is precise if the maximum simulation time step size is below 1/(2*cFreq).
The clock signal is generated only at time > 0.
vardef(cFreq) = 'Hz';'Clock frequency'
vardef(clock) = ;'Clock signal'
vardef(output) = ;'Clock output signal'

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *6*
Output signals: output
Continuous states: x
Parameters: cFreq
Internal variables: clock

20.2 Clock (t>0) _sig

Clock signal (square wave) of variable frequency at t>0

Functionality: Outputs a square wave signal with frequency "extfrq" [Hz].
The "on" pulse has the half width of the period of the signal (Ton/Tp = 0.5).
The clock frequency can be changed throughout the simulation via input signal "extfrq".
The waveform is precise if the maximum simulation time step size is below 1/(2*extfrq).
The clock signal is generated only at time > 0.
vardef(extfrq) = 'Hz';'Clock frequency'
vardef(clock) = ;'Clock signal'
vardef(output) = ;'Clock output signal'

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *6*
Output signals: output
Input signals: extfrq
Continuous states: x
Internal variables: clock

20.3 Clock (t>t0) _par

Clock signal (square wave) of fixed frequency at t>t0

Functionality: Outputs a square wave signal with frequency cFreq [Hz].
The "on" pulse has the half width of the period of the signal (Ton/Tp = 0.5).
The clock frequency can be changed via parameter "cFreq".
The waveform is precise if the maximum simulation time step size is below 1/(2*cFreq).
The macro supresses a derivative of state variable being larger than zero at initial condition, this results in a delay of 1.0 us of the clock signal at simulation start.
vardef(cFreq) = 'Hz';'Clock frequency'
vardef(clock) = ;'Clock signal'
vardef(output) = ;'Clock output signal'

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *6*
Output signals: output
Continuous states: x
Parameters: cFreq
Internal variables: t0,clock

20.4 Clock (t>t0) _sig

Clock signal (square wave) of variable frequency at t>t0

Functionality: Outputs a square wave signal with frequency "extfrq" [Hz].
The "on" pulse has the half width of the period of the signal (Ton/Tp = 0.5).

The clock frequency can be changed throughout the simulation via input signal "extfrq".
The waveform is precise if the maximum simulation time step size is below 1/(2*extfrq).
The macro supresses a derivative of state variable being larger than zero at initial condition, this results in a delay of 1.0 us of the clock signal at simulation start.
vardef(extfrq) = 'Hz';'Clock frequency'
vardef(clock) = ;'Clock signal'
vardef(output) = ;'Clock output signal'

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *6*
Output signals: output
Input signals: extfrq
Continuous states: x
Internal variables: t0,clock

20.5 Clock _par

Clock signal (square wave) of fixed frequency

Functionality: Outputs a square wave signal with frequency cFreq [Hz].
The "on" pulse has the half width of the period of the signal (Ton/Tp = 0.5).
The clock frequency can be changed via parameter "cFreq".
The waveform is precise if the maximum simulation time step size is below 1/(2*cFreq).
The macro contains a state variable with a derivative being larger than zero at initial condition.
vardef(cFreq) = 'Hz';'Clock frequency'
vardef(clock) = ;'Clock signal'
vardef(output) = ;'Clock output signal'

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *6*
Output signals: output
Continuous states: x
Parameters: cFreq
Internal variables: clock

20.6 Clock _sig

Clock signal (square wave) of variable frequency

Functionality: Outputs a square wave signal with frequency "extfrq" [Hz].
The "on" pulse has the half width of the period of the signal (Ton/Tp = 0.5).
The clock frequency can be changed throughout the simulation via input signal "extfrq".
The waveform is precise if the maximum simulation time step size is below 1/(2*extfrq).
The macro contains a state variable with a derivative being larger than zero at initial condition.
vardef(extfrq) = 'Hz';'Clock frequency'
vardef(clock) = ;'Clock signal'
vardef(output) = ;'Clock output signal'

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *6*
Output signals: output
Input signals: extfrq
Continuous states: x
Internal variables: clock

20.7 Pulse

Creates a pulse for T1 sec, if input is larger than K

Functionality: Creates a pulse for T1 sec, if input is larger than K
This macro has a linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: K,T1
Internal variables: y1,y2

20.8 Sawtooth Wave Generator

Sawtooth Wave Generator

Functionality: Outputs a sawtooth signal with f=freq [Hz]
The block does not generate interruption events.
The waveform is precise only if the simulation time step size (throughout the simulation) is less than 1/f.

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: f
Internal variables: t0

20.9 Sawtooth Wave Generator _ip

Sawtooth Wave Generator, precise with interruption events

Functionality: Outputs a sawtooth signal with f=freq [Hz]
The block enforces interruption events at sawtooth wave peaks.
The waveform is precise if the minimum simulation time step size is less than 1/f (maximum

time step can be greater). Simulations using this block ("Sawtooth Wave Generator _ip") are typically slower than when using the block "Sawtooth Wave Generator".

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: f
Internal variables: t0,trigger

20.10 Sine Wave Generator

Outputs a sinusoidal signal with f=freq [Hz]

Functionality: Outputs a sinusoidal signal with f=freq [Hz]
This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: ampl,freq
Internal variables: t

20.11 Sine Wave Generator (t>0)

Outputs a sinusoidal signal with f=freq [Hz] at t>0

Functionality: Outputs a sinusoidal signal with f=freq [Hz] at t>0
This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: ampl,freq
Internal variables: t

20.12 Square Wave Generator

Square Wave Generator

Functionality: Outputs a square wave signal with f=freq [Hz]
The block does not generate interruption events.
The waveform is precise only if the simulation time step size (throughout the simulation) is less than 1/f.

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: f
Internal variables: t0

20.13 Square Wave Generator_ip

Square Wave Generator, precise with interruption events

Functionality: Outputs a square wave signal with f=freq [Hz]
The block enforces interruption events at square wave flanks (rising/falling).
The waveform is precise if the MAXIMUM simulation time step size is below 1/(2f). Simulations using this block ("Square Wave Generator _ip") are typically slower than when using the block "Square Wave Generator".

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: f
Internal variables: t0

20.14 Time

Outputs the simulation time

Functionality: Outputs the simulation time
This macro has a linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: t

20.15 Triangle Wave Generator

Triangle Wave Generator

Functionality: Outputs a triangle signal with f=freq [Hz].
The block does not generate interruption events.
The waveform is precise only if the simulation time step size (throughout the simulation) is less than 1/f.

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*

Output signals: yo
Parameters: f
Internal variables: toffset

20.16 Triangle Wave Generator_ip

Triangle Wave Generator, precise with interruption events

Functionality: Outputs a triangle signal with f=freq [Hz]
Enforces interruption events at top and bottom triangle wave peaks.
The waveform is precise if the minimum simulation time step size is below 1/f. Simulations using this block ("Triangle Wave Generator _ip") are typically slower than when using the block "Triangle Wave Generator".

This macro has a non-linear behaviour.

Macro location: *Macros\Signals*
Macro DSL level: *5*
Output signals: yo
Parameters: f
Internal variables: toffset,trigger

21 Switches / Selectors

This section provides a complete listing of the existing DSL macros within the *Switches / Selectors* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

21.1 Enable 1 sig

Enable an input to pass through depending on signal flag

Functionality: Enable an input to pass through depending on signal flag
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi (throughout the simulation)
Enable < 0.5 (at inc) then yo = yi_default (throughout the simulation)

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: Enable
Parameters: yi_default

21.2 Enable 1 sig _hold

Enable an input to pass through depending on signal flag or hold

Functionality: Enable an input to pass through depending on signal flag
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Upper limitation input signals: Enable
Internal variables: yi_register

21.3 Enable 2 sig

Enable inputs to pass through depending on signal flag

Functionality: Enable inputs to pass through depending on signal flag
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then yo = yi_default

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2
Input signals: yi1,yi2
Upper limitation input signals: Enable
Parameters: yi1_default,yi2_default

21.4 Enable 2 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*

Output signals: yo1,yo2
Input signals: yi1,yi2
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register

21.5 Enable 3 sig

Enable inputs to pass through depending on signal flag

Functionality: Enable inputs to pass through depending on signal flag
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then yo = yi_default

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3
Input signals: yi1,yi2,yi3
Upper limitation input signals: Enable
Parameters: yi1_default,yi2_default,yi3_default

21.6 Enable 3 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3
Input signals: yi1,yi2,yi3
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register,yi3_register

21.7 Enable 4 sig

Enable inputs to pass through depending on signal flag

Functionality: Enable inputs to pass through depending on signal flag
Output:

Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then yo = yi_default

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4
Input signals: yi1,yi2,yi3,yi4
Upper limitation input signals: Enable
Parameters: yi1_default,yi2_default,yi3_default,yi4_default

21.8 Enable 4 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4
Input signals: yi1,yi2,yi3,yi4
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register,yi3_register,yi4_register

21.9 Enable 5 sig

Enable inputs to pass through depending on signal flag

Functionality: Enable inputs to pass through depending on signal flag
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then yo = yi_default

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4,yo5
Input signals: yi1,yi2,yi3,yi4,yi5
Upper limitation input signals: Enable
Parameters: yi1_default,yi2_default,yi3_default,yi4_default,yi5_default

21.10 Enable 5 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4,yo5
Input signals: yi1,yi2,yi3,yi4,yi5
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register,yi3_register,yi4_register,yi5_register

21.11 Enable 6 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4,yo5,yo6
Input signals: yi1,yi2,yi3,yi4,yi5,yi6
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register,yi3_register,yi4_register,yi5_register,yi6_register

21.12 Enable 7 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*

Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4,yo5,yo6,yo7
Input signals: yi1,yi2,yi3,yi4,yi5,yi6,yi7
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register,yi3_register,yi4_register,yi5_register,yi6_register,yi7_register

21.13 Enable 8 sig _hold

Enable inputs to pass through depending on signal flag or hold

Functionality: Enable inputs to pass through depending on signal flag or hold otherwise
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi
Enable < 0.5 (at inc) then hold output

This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2,yo3,yo4,yo5,yo6,yo7,yo8
Input signals: yi1,yi2,yi3,yi4,yi5,yi6,yi7,yi8
Upper limitation input signals: Enable
Internal variables: yi1_register,yi2_register,yi3_register,yi4_register,yi5_register,yi6_register,yi7_register,yi8_regist

21.14 Enable signal

Enable an input to pass through depending on parameter flag

Functionality: Enable an input to pass through depending on parameter flag
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 (at inc) then yo = yi (throughout the simulation)
Enable < 0.5 (at inc) then yo = 0 (throughout the simulation)

vardef(Enable)='0/1';'Enable flag (0=disable;1=enable)'
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: Enable

21.15 Enable signal (fixed)

Enable an input to pass through depending on parameter flag (fixed)

Enable an input to pass through depending on parameter flag
Note: Switch is fixed throughout the simulation
Output:
Enable >= 0.5 (at inc) then yo = yi (throughout the simulation)
Enable < 0.5 (at inc) then yo = 0 (throughout the simulation)

vardef(Enable)='0/1';'Enable flag (0=disable;1=enable)'
This macro has a linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: Enable

21.16 Switch par 1->1 by par

Outputs either the input or a parameter depending on parameter flag

Outputs either the input or a parameter depending on parameter flag
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 then yo = yi
Enable < 0.5 then yo = p

vardef(Enable)='0/1';'Enable flag (0=disable;1=enable)'
vardef(p) = ;'Output value if flag disabled'
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: Enable,p

21.17 Switch par 1->1 by par (fixed)

Outputs either the input or a parameter depending on parameter flag (fixed)

Outputs either the input or a parameter depending on parameter flag
Note: Switch is fixed throughout the simulation
Output:
Enable >= 0.5 (at inc) then yo = yi (throughout the simulation)
Enable < 0.5 (at inc) then yo = p (throughout the simulation)

vardef(Enable)='0/1';'Enable flag (0=disable;1=enable)'
vardef(p) = ;'Output value if flag disabled'
This macro has a linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*

Output signals: yo
Input signals: yi
Parameters: Enable,p

21.18 Switch par 1->2 by par

Parameter switch 1->2 by parameter

Functionality: Parameter switch 1->2 by parameter
Note: Switch is fixed throughout the simulation
Operation:
If sw < 0.5 then parameter K is routed to output yo1 (and yo2 is 0)
If sw >=0.5 then parameter K is routed to output yo2 (and yo1 is 0)
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2
Parameters: sw,K

21.19 Switch par 1->2 by sig

Parameter switch 1->2 by signal

Functionality: Parameter switch 1->2 by signal sw
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw < 0.5 then parameter K is routed to output yo1 (and yo2 is 0)
If sw >=0.5 then parameter K is routed to output yo2 (and yo1 is 0)
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2
Input signals: sw
Parameters: K

21.20 Switch par 2->1 by par

Parameter switch 2->1 by parameter

Functionality: Parameter switch 2->1 by parameter
Note: Switch is fixed throughout the simulation
Operation:
If sw <=0 then K1 is routed to output
If sw > 0 then K2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Parameters: sw,K1,K2

21.21 Switch par 2->1 by sig

Signal switch 2->1 by signal

Functionality: Signal switch 2->1 by signal
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw <=0 then parameter K1 is routed to output
If sw > 0 then parameter K2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: sw
Parameters: K1,K2

21.22 Switch sig 1->1 by sig

Outputs either the input or a parameter depending on signal flag

Outputs either the input or a parameter depending on an input signal flag "Enable"
Note: Output can switch during the simulation via parameter events on Enable
Output:
Enable >= 0.5 then yo = yi
Enable < 0.5 then yo = p

vardef(Enable)='0/1';'Enable flag (0=disable;1=enable)'
vardef(p) = ;'Output value if flag disabled'
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,Enable
Parameters: p

21.23 Switch sig 1->1 by sig (fixed)

Outputs either the input or a parameter depending on signal flag (fixed)

Outputs either the input or a parameter depending on an input signal flag "Enable"
Note: Switch is fixed throughout the simulation

Output:
Enable >= 0.5 (at inc) then yo = yi (throughout the simulation)
Enable < 0.5 (at inc) then yo = p (throughout the simulation)

vardef(Enable)='0/1';'Enable flag (0=disable;1=enable)'
vardef(p) = ;'Output value if flag disabled'
This macro has a linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,Enable
Parameters: p

21.24 Switch sig 1->2 by par

Signal switch 1->2 by parameter

Functionality: Signal switch 1->2 by parameter
Note: Switch is fixed throughout the simulation
Operation:
If sw < 0.5 then input is routed to output yo1 (and yo2 is 0)
If sw >=0.5 then input is routed to output yo2 (and yo1 is 0)
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2
Input signals: yi
Parameters: sw

21.25 Switch sig 1->2 by par (bool)

Signal switch 1->2 by parameter(variant)

Functionality: Signal switch 1->2 by parameter (variant)
Note: Switch is fixed throughout the simulation
Operation:
If sw< 0.5 then outputs are zero
If 0.5<=sw<=1.5 then input is routed to yo1 (and yo2=0)
If sw> 1.5 then input is routed to yo2 (and yo1=0)
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2
Input signals: yi
Parameters: sw

21.26 Switch sig 1->2 by sig

Signal switch 1->2 by signal

Functionality: Signal switch 1->2 by signal sw
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw < 0.5 then input is routed to output yo1 (and yo2 is 0)
If sw >=0.5 then input is routed to output yo2 (and yo1 is 0)
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo1,yo2
Input signals: yi,sw

21.27 Switch sig 2->1 (NOT EQ K) by s/p

Signal switch 2->1 by signal

Functionality: Signal switch 2->1 (NOT EQ K) by signal/parameter
Note: Output can switch during the simulation (via changing signal sw or parameter events on K)
Operation:
If sw <> K then yi1 is routed to output.
Otherwise yi2 is routed to output.
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,sw
Parameters: K

21.28 Switch sig 2->1 by par

Signal switch 2->1 by parameter

Functionality: Signal switch 2->1 by parameter
Note: Switch is fixed throughout the simulation
Operation:
If sw <=0 then yi1 is routed to output
If sw > 0 then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: sw

21.29 Switch sig 2->1 by par (bool)

Signal switch 2->1 by parameter

Functionality: Signal switch 2->1 by parameter
Note: Switch is fixed throughout the simulation
Operation:
If sw< 0.5 then output is zero
If 0.5<=sw<=1.5 then yi1 is routed to output
If sw> 1.5 then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2
Parameters: sw

21.30 Switch sig 2->1 by sig

Signal switch 2->1 by signal

Functionality: Signal switch 2->1 by signal
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw <=0 then yi1 is routed to output
If sw > 0 then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,sw

21.31 Switch sig 2->1 by sig (bool)

Signal switch 2->1 by signal (threshold at 0.5)

Passes through a single input depending on the state of input signal "sw":
Note: Output can switch during the simulation (via changing signal sw)
pass first input if sw< 0.5 e.g. sw = 0
pass second input if sw>=0.5 e.g. sw = 1
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,sw

21.32 Switch sig 2->1 by sig (fixed)

Signal switch 2->1 fixed by signal (threshold at 0.5)

Functionality: Passes through a single input depending on the state of input signal "sw":
Note: Switch is fixed throughout the simulation
pass first input if sw< 0.5 e.g. sw = 0
pass second input if sw>=0.5 e.g. sw = 1
This macro has a linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,sw

21.33 Switch sig 3->1 by sig

Signal switch 3->1 by signal

Passes through a single input depending on the state of input signal "sw":
Note: Output can switch during the simulation (via changing signal sw)
pass first input if sw<0.5 e.g. sw = 0
pass second input if 0.5<=sw<1.5 e.g. sw = 1
pass third input if 1.5<=sw e.g. sw = 2
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3,sw

21.34 Switch sig 4->1 by sig

Signal switch 4->1 by signal

Passes through a single input depending on the state of input signal "sw":
Note: Output can switch during the simulation (via changing signal sw)
pass first input if sw<0.5 e.g. sw = 0
pass second input if 0.5<=sw<1.5 e.g. sw = 1
pass third input if 1.5<=sw<2.5 e.g. sw = 2
pass fourth input if 2.5<=sw e.g. sw = 3
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,yi2,yi3,yi4,sw

21.35 Switch sw equal C 2s->1s

Switch sw =C 2s->1s

Functionality: Switch sw=C 2s->1s
Pass first input if condition is true, otherwise pass third input.
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw =C then yi1 is routed to output
If sw ~=C then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,sw,yi2
Parameters: C

21.36 Switch sw greater than C 2s->1s

Switch sw>C 2s->1s

Functionality: Switch sw>C 2s->1s
Pass first input if condition is true, otherwise pass third input.
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw >C then yi1 is routed to output
If sw <=C then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,sw,yi2
Parameters: C

21.37 Switch sw greater than or equal C 2s->1s

Switch sw>=C 2s->1s

Functionality: Switch sw>=C 2s->1s
Pass first input if condition is true, otherwise pass third input.
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw >=C then yi1 is routed to output
If sw <C then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo

Input signals: yi1,sw,yi2
Parameters: C

21.38 Switch sw not equal C 2s->1s

Switch sw =C 2s->1s

Functionality: Switch sw~=C 2s->1s
Pass first input if condition is true, otherwise pass third input.
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw ~=C then yi1 is routed to output
If sw =C then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,sw,yi2
Parameters: C

21.39 Switch sw smaller than C 2s->1s

Switch sw<C 2s->1s

Functionality: Switch sw<C 2s->1s
Pass first input if condition is true, otherwise pass third input.
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw <C then yi1 is routed to output
If sw >=C then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,sw,yi2
Parameters: C

21.40 Switch sw smaller than or equal C 2s->1s

Switch sw<=C 2s->1s

Functionality: Switch sw<=C 2s->1s
Pass first input if condition is true, otherwise pass third input.
Note: Output can switch during the simulation (via changing signal sw)
Operation:
If sw <=C then yi1 is routed to output

If sw > C then yi2 is routed to output
This macro has a non-linear behaviour.

Macro location: *Macros\Switches / Selectors*
Macro DSL level: *5*
Output signals: yo
Input signals: yi1,sw,yi2
Parameters: C

22 Timers

This section provides a complete listing of the existing DSL macros within the *Timers* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

22.1 Timer _reset

Timer with reset, resets at 0 upon triggering (rising edge of rst)

Functionality: Timer with reset. Starts always at zero. If trigerred via signal rst then it resets to zero.
Reset signal rst expected within range [0,1].
Triggering occurs when 'rst' signal crosses 0.5 on the rising edge.
This macro has a non-linear behaviour.

Macro location: *Macros\Timers*
Macro DSL level: *6*
Output signals: yo
Input signals: rst
Internal variables: dt

22.2 Timer (reset/hold reset/t0) _reset_incfw

Timer with reset, hold reset, initial timer value and delayed start

Functionality: Timer with reset, hold reset, initial timer value and delayed start.
Reset option: This function includes a state/internal variable reset (based on an input signal)
Forward initial condition option: Output is initialised based on the input.
Reset signal rst expected within range [0,1]
Inputs: rst - reset signal between 0 and 1 (including); when rst input greater/smaller than 0.5 then reset timer
t0 - value of time upon reset
Outputs:
yo - outputs the timer value; initialises to t0, resets to t0
Parameters:
flank - '0/1';'0:rising / 1:falling'
flank = 0 (rising flank) then when rst>0.5 apply timer reset
flank = 1 (falling flank) then when rst<0.5 apply timer reset

t_start_delay - Delay in seconds until timer starts: freezes the output for t_start_delay seconds upon simulation start and has no influence afterwards.
This macro has a non-linear behaviour.

Macro location: *Macros\Timers*
Macro DSL level: *5*
Output signals: yo
Input signals: rst,t0
Continuous states: x
Parameters: flank,t_start_delay
Internal variables: tinc,tldf,dir,tx

23 Transformations

This section provides a complete listing of the existing DSL macros within the *Transformations* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

23.1 Clarke transform

Clarke transform

Functionality: implements the Clarke transform for EMT type signals
Transforms instantaneous A/B/C components to alpha/beta/gamma components
This macro has a linear behaviour.

Function:

$$\begin{bmatrix} alpha \\ beta \\ gamma \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: alpha,beta,gamma
Input signals: a,b,c

23.2 Inverse Clarke transform

Inverse Clarke transform

Functionality: implements the inverse Clarke transform for EMT type signals
Transforms alpha/beta/gamma components to A/B/C instantaneous components
This macro has a linear behaviour.

Function:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \begin{bmatrix} alpha \\ beta \\ gamma \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: a,b,c
Input signals: alpha,beta,gamma

23.3 Inverse Park transform (dq)

Inverse Park transform (dq)

Functionality: This macro implements the inverse Park transform (d/q)
This macro has a linear behaviour.

Function:

$$\begin{bmatrix} alpha \\ beta \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) \\ sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: alpha,beta
Input signals: d,q,cosphi,sinphi

23.4 Inverse Park transform (dq0)

Inverse Park transform (dq0)

Functionality: This macro implements the inverse Park transform (d/q/0)
This macro has a linear behaviour.

Function:

$$\begin{bmatrix} alpha \\ beta \\ gamma \end{bmatrix} = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d \\ q \\ zero \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: alpha,beta,gamma
Input signals: d,q,zero,cosphi,sinphi

23.5 Park transform (dq)

Park transform (dq)

Functionality: This macro implements the forward Park transform (alpha,beta)->(d,q)
This macro has a linear behaviour.

Function:

$$\begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} alpha \\ beta \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: d,q
Input signals: alpha,beta,cosphi,sinphi

23.6 Park transform (dq0)

Park transform (dq0)

Functionality: This macro implements the forward Park transform (alpha,beta,gamma)->(d,q,0)
This macro has a linear behaviour.

Function:

$$\begin{bmatrix} d \\ q \\ zero \end{bmatrix} = \begin{bmatrix} cos(\phi) & sin(\phi) & 0 \\ -sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} alpha \\ beta \\ gamma \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: d,q,zero
Input signals: alpha,beta,gamma,cosphi,sinphi

23.7 RMS value

Calculation of RMS value of a signal in EMT simulation

Functionality: Calculation of RMS value of a signal in EMT simulation
Caution: The RMS value needs one period to settle.
The result is not precise in the first period.
In RMS simulation this block is a simple feed-through (yo = yi).

This macro has a linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: fn

23.8 RMS value p.u.

Calculation of RMS value of a signal in p.u. in EMT simulation

Functionality: Calculation of RMS value of a signal in p.u. in EMT simulation
Input signal yi: p.u. base is nominal peak value (1 p.u. = nominal peak value = sqrt(2) * nominal RMS value).
Outut signal yo: p.u. base is nominal RMS value (1 p.u. = nominal RMS value).
Caution: The RMS value needs one period to settle.
The result is not precise in the first period.
In RMS simulation this block is a simple feed-through (yo = yi).

This macro has a linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: fn

23.9 U seq/ab0 -> U abc

Calculation of line-ground and line-line voltage RMS magnitudes in RMS and EMT simulation

Functionality:
In case of RMS simulation: vector transformation from sequence components to phase voltages (line and lini-line)
[ua] [1 1 1] [u1]
[ub] = [aˆ2 a 1] * [u2]
[uc] [a aˆ2 1] [u0]
with:
a = -1/2 + j*sqrt(3)/2
aˆ2 = -1/2 - j*sqrt(3)/2

In case of EMT simulation: inverse Clarke transform for instantaneous values with subsequent per phase RMS value calculation.
[ua] [1 0 1] [ualpha] with ualpha = u1r
[ub] = [-1/2 +sqrt(3)/2 1] * [ubeta] with ubeta = u1i
[uc] [-1/2 -sqrt(3)/2 1] [u0]

This macro has a linear behaviour.

Function for RMS simulation:

$$\begin{bmatrix} \underline{u}_a \\ \underline{u}_b \\ \underline{u}_c \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \underline{a}^2 & \underline{a} & 1 \\ \underline{a} & \underline{a}^2 & 1 \end{bmatrix} \begin{bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \underline{u}_0 \end{bmatrix}$$

with: $\underline{a} = -\frac{1}{2} + j\frac{\sqrt{3}}{2}$
and:

$$\begin{bmatrix} ua \\ ub \\ uc \end{bmatrix} = \begin{bmatrix} \|\underline{u}_a\| \\ \|\underline{u}_b\| \\ \|\underline{u}_c\| \end{bmatrix}$$

Function for EMT simulation:

$$\begin{bmatrix} ua \\ ub \\ uc \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} \begin{bmatrix} u1r \\ u1i \\ u0 \end{bmatrix}$$

Macro location: *Macros\Transformations*
Macro DSL level: *5*
Output signals: ua,ub,uc,uab,ubc,uca
Input signals: u1,u1r,u1i,u2r,u2i,u0r,u0i,u0
Parameters: fn
Internal variables: uar,uai,ubr,ubi,ucr,uci,t,t0

23.10 abc->dq0 (power invariant – align a->d)

DQ0 transform - power invariant (abc->dq0, a->d alignment, emt)

Functionality: This macro implements the power invariant forward Park transformatopm (a,b,c)->(d,q,0)
It aligns phase a to d-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: d,q,zero
Input signals: a,b,c,theta
Internal variables: sqrt2_3,one_sqrt2,twopi_3

23.11 abc->dq0 (power invariant – align a->q)

DQ0 transform - power invariant (abc->dq0, a->q alignment, emt)

Functionality: This macro implements the power invariant forward Park transformatopm (a,b,c)->(d,q,0)
It aligns phase a to the q-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: d,q,zero
Input signals: a,b,c,theta
Internal variables: sqrt2_3,one_sqrt2,twopi_3

23.12 abc->dq0 (power variant – align a->d)

DQ0 transform - power variant (abc->dq0, a->d alignment, emt)

Functionality: This macro implements the power variant forward Park transformatopm (a,b,c)->(d,q,0)
It aligns phase a to the d-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: d,q,zero
Input signals: a,b,c,theta
Internal variables: ratio2_3,twopi_3

23.13 abc->dq0 (power variant – align a->q)

DQ0 transform - power variant (abc->dq0, a->q alignment, emt)

Functionality: This macro implements the power variant forward Park transformatopm (a,b,c)->(d,q,0)
It aligns phase a to the q-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: d,q,zero
Input signals: a,b,c,theta
Internal variables: ratio2_3,twopi_3

23.14 dq0->abc (power invariant – align a->d)

Inverse DQ0 transform - power invariant (dq0->abc, a->d alignment, emt)

Functionality: This macro implements the power invariant inverse Park transformatopm (d,q,0)->(a,b,c)
It aligns phase a to d-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: a,b,c
Input signals: d,q,zero,theta
Internal variables: sqrt2_3,one_sqrt2,twopi_3

23.15 dq0->abc (power invariant – align a->q)

Inverse DQ0 transform - power invariant (dq0->abc, a->q alignment, emt)

Functionality: This macro implements the power invariant inverse Park transformatopm (d,q,0)->(a,b,c)
It aligns phase a to q-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: a,b,c
Input signals: d,q,zero,theta
Internal variables: sqrt2_3,one_sqrt2,twopi_3

23.16 dq0->abc (power variant – align a->d)

Inverse DQ0 transform - power invariant (dq0->abc, a->d alignment, emt)

Functionality: This macro implements the power variant inverse Park transformatopm (d,q,0)->(a,b,c)
It aligns phase a to d-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: a,b,c
Input signals: d,q,zero,theta
Internal variables: twopi_3

23.17 dq0->abc (power variant – align a->q)

Inverse DQ0 transform - power variant (dq0->abc, a->q alignment, emt)

Functionality: This macro implements the power variant inverse Park transformatopm (d,q,0)->(a,b,c)
It aligns phase a to q-axis.
Supports EMT simulation only.
This macro has a non-linear behaviour.

Macro location: *Macros\Transformations*
Macro DSL level: *6*
Output signals: a,b,c
Input signals: d,q,zero,theta
Internal variables: twopi_3

24 Unit Conversion

This section provides a complete listing of the existing DSL macros within the *Unit Conversion* folder. Their functionality is explained along with a list of the input and output signals, state variables and parameters.

24.1 Hz -> p.u.

Convert speed/frequency from Hz to p.u.

Functionality: This macro converts an input frequency/speed to p.u., with the base frequency equal to parameter freqbase.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: fpu
Input signals: Freq
Parameters: freqbase

24.2 Nm -> p.u.

Convert torque in Nm to p.u.

Functionality: This macro converts a torque quantity expressed in Nm into pu, with base parameters freqbase, Zp and Pel_base.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: m
Input signals: M
Parameters: freqbase,Zp,Pel_base

24.3 abs -> p.u. (par)

Divide by base (parameter)

Functionality: This block converts the input quantity to per unit, with the base value defined by the parameter 'base'.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: base

24.4 abs -> p.u. (sig)

Divide by base (input signal)

Functionality: This block converts the input quantity to per unit, with the base value defined by the input signal 'base'.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,base

24.5 deg -> rad

Convert angle from degrees to radians

Functionality: This macro converts the input yi, expressed in degrees, into radians.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

24.6 p.u. -> Hz

Convert speed/frequency from p.u. to Hz

Functionality: This macro converts the input signal fpu, in p.u., to Hz using the nominal frequency "freqbase".
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: Freq
Input signals: fpu
Parameters: freqbase

24.7 p.u. -> abs (par)

Multiply by base (parameter)

Functionality: This macro multiplies the input by the parameter "base"
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: yo
Input signals: yi
Parameters: base

24.8 p.u. -> abs (sig)

Multiply by base (signal)

Functionality: This macro multiplies the input "yi" by the input signal "base"
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: yo
Input signals: yi,base

24.9 p.u. -> rpm

Convert speed/frequency from p.u. to rpm

Functionality: This macro converts the input signal fpu, in p.u., to rpm using the nominal frequency "freqbase" and "Zp".
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: n
Input signals: speed
Parameters: Zp,freqbase

24.10 rad -> deg

Convert angle from radians to degrees

Functionality: This macro converts the input, expressed in radians, into degrees.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: yo
Input signals: yi

24.11 rad/s -> rpm

Convert speed/frequency from rad/s to rpm

Functionality: This macro converts the input, expressed in rad/s, into rpm.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: n
Input signals: omega

24.12 rpm -> p.u.

Convert speed/frequency from rpm to p.u.

Functionality: This macro converts the input signal "n", expressed in rpm, into p.u. using the parameters "freqbase" and "Zp".
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: speed
Input signals: n
Parameters: Zp,freqbase

24.13 rpm -> rad/s

Convert speed/frequency from rpm to rad/s

Functionality: This macro converts the input signal "n", expressed in rpm, into rad/s.
This macro has a linear behaviour.

Macro location: *Macros\Unit Conversion*
Macro DSL level: *5*
Output signals: omega
Input signals: n