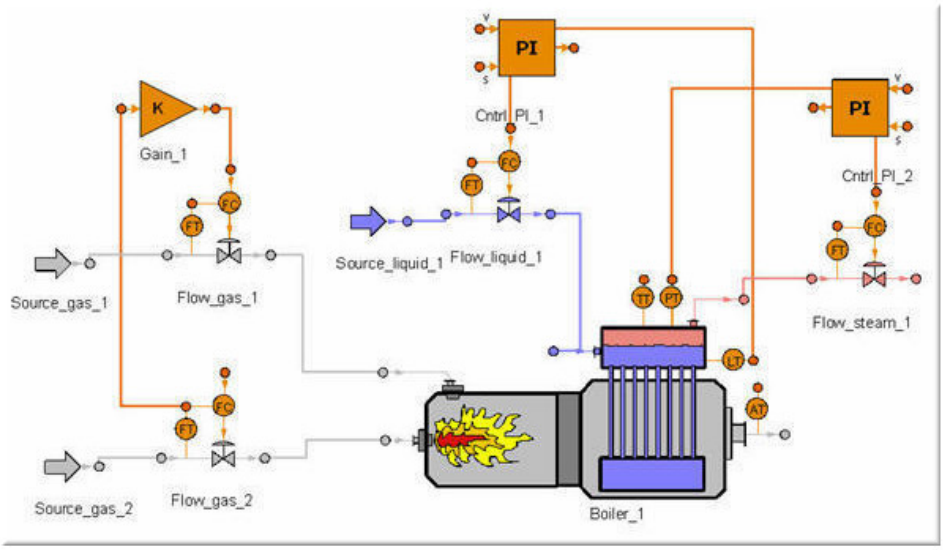
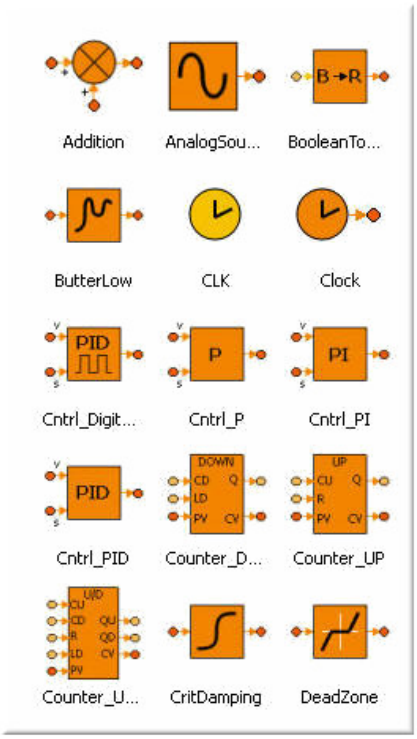
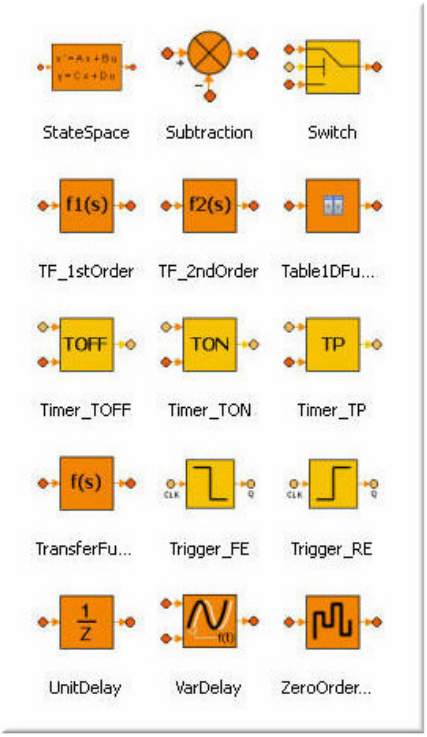
CONTROL Library 4.0.3

Reference Manual







EMPRESARIOS AGRUPADOS

Edition 2015

This page intentionally left blank.

Contents

1. Introduction

1.1 Purpose of This Manual

This document is the User Manual for the CONTROL library. The CONTROL library provides the custom items needed to represent analog and digital control systems. It can be used as a stand-alone library or together with other libraries.

This library allows to easily build models dragging and dropping the symbols from the library palette and connecting them.

The manual contains four chapters. The chapter two describes the ports of the library, the enumerative type data defined in the library and a brief description of the available components in the library.

The chapter three provides reference information for all abstract components defined in the library. The abstract components describe a behaviour that alone does not represent any physical component and can only be used as a base component for other components.

The chapter four provides reference information for all CONTROL library components.

This page intentionally left blank.

2. CONTROL Library Items

2.1 Enumeration Types

2.1.1 Enumeration Type EndPosBehaviour

ENUM EndPosBehaviour = {end_I, end_PI}

This is an enumeration type to define the anti-windup behavior of a PI or PID controller. The two possibilities are: end_I meaning only integral behavior at the end position of the controller, and end_PI meaning proportional-integral behaviour at the end position.

2.1.2 Enumeration Type InitOption

ENUM InitOption= {InitialStates, InitialOutput}

This is an enumeration type to define how the user wants to initializate the output signal (InitialOutput) or the state variables (InitialStates). This enumeration type is used in the components ZeroPole, dZeroPole and dTransferFunction.

2.1.3 Enumeration Type MathOption

ENUM MathOption = {FunAbs, FunSign, FunSqrt, FunSin, FunCos, FunTan,

FunAsin, FunAcos, FunAtan, FunSinh, FunCosh, FunTanh,

FunExp, FunLog, FunLog10}

This type is employed to select the mathematical function to perform on the input signals. This enumeration type is used in the component MathFunction. And the available options are listed in the following table:

option	Description
FunAbs	To calculate the absolute value of the input signals
FunSign	To calculate the sign of the input signals
FunSqrt	To calculate the square root of the input signals
FunSin	To calculate the sine of the input signals
FunCos	To calculate the cosine of the input signals
FunTan	To calculate the tangent of the input signals
FunAsin	To calculate the arc-sine of the input signals
FunAcos	To calculate the arc-cosine of the input signals
FunAtan	To calculate the arc-tangent of the input signals
FunSinh	To calculate the hyperbolic sine of the input signals
FunCosh	To calculate the hyperbolic cosine of the input signals
FunTanh	To calculate the hyperbolic tangent of the input signals
FunExp	To calculate the exponential of the input signals
FunLog10	To calculate the base 10 logarithm of the input signals
FunLog	To calculate the natural logartihm of the input signals

2.1.4 Enumeration Type RandomOption

ENUM RandomOption = {Uniform, Binomial, Exponential, Gamma, Gaussian, Poisson}

This type is used to define the probability distribution to generate the random numbers. The different options are:

random option	Description
Uniform	Uniform distribution
Binomial	Binomial distribution
Exponential	Exponential distribution
Gamma	Gamma distribution
Gaussian	Gaussian distribution
Poisson	Poisson distribution

2.1.5 Enumeration Type RelationalOption

ENUM RelationalOption = {Equal, NotEqual, GreaterThan, GreaterEqual, LessThan, LessEqual}

This enumeration type allows to define the relational operation to perform on the input signals of the component RelationalOperator. The available relational operations are:

Relational Operator option	Output signal Value
Equal	TRUE if the first input signal is equal to the second input signal
NotEqual	TRUE if the first input signal is not equal to the second input signal
GreaterThan	TRUE if the first input signal is greater than the second input signal
GreaterEqual	TRUE if the first input signal is greater than or equal to the second input signal
LessThan	TRUE if the first input signal is less than the second input signal
LessEqual	TRUE if the first input signal is less than or equal to the second input signal

2.1.6 Enumeration Type SourceOption

ENUM SourceOption = {Source_Constant, Source_Sine, Source_Pulse, Source_Step, Source_SawTooth, Source_Square, Source_Ramp, Source_TableStep, Source_TableInterp}

This enumeration type is used in the component AnalogSource. It allows to select which type of output signal the user wants to generate. The different possibilities are:

Source option	Description
Source_Constant	To generate constant signals
Source_Sine	To generate sinusoidal signals
Source_Pulse	To general pulse signals
Source_Step	To generate step signals
Source_SawTooth	To generate saw tooth signals
Source_Square	To generate square signals
Source_Ramp	To generate ramp signals
Source_Table	To generate signals by interpolation in a table

2.1.7 Enumeration Type dIntegMethod

ENUM dIntegMethod = {ForwardEuler, BackwardEuler, Trapezoidal}

This enumerative type is used in the component dIntegrator. It allows to specify the integration method used to calculate the discrete-time integration of the input signal.

2.1.8 Enumeration Type state_type

ENUM state_type = {OFF, ON}

This type is employed to represent the state of the Relay components.

2.1.9 Enumeration Type tableMethod

ENUM tableMethod = {LinearInterpWithEvents, LinearInterpWithoutEvents, SplineInterp, StepConnect}

This type represents the available methods to interpolate or connect the table points. It is used in the AnalogSource and the SourceDataFile components.

2.2 Port Types

2.2.1 analog_signal

Description

This port type represents an 1-dimension array of analog signals

Limitations

SINGLE IN

"SINGLE IN" in the declaration of the signal port types means that multiple connections to signal input ports are forbidden. However, it is possible to make multiple connection from an outlet signal port, i.e it is possible to broadcast an outlet signal.

Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
n	CONST INTEGER	1	Dimension of the signal array	-

Variables

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
signal[n]	EQUAL OUT REAL			Real variable array	-

The EQUAL OUT type for the variable especifies that it is possible to connect an output port to multiple input ports and the values of the signals of the connected ports are equal.

2.2.2 bool_signal

Description

This port type represents an 1-dimension array of Boolean signals

Limitations

SINGLE IN

"SINGLE IN" in the declaration of the signal port types means that multiple connections to signal input ports are forbidden. However, it is possible to make multiple connection from an outlet signal port, i.e it is possible to broadcast an outlet signal.

Parameters

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
n	CONST INTEGER	1	Dimension of the signal array	-

Variables

NAME	TYPE	INITIAL	RANGE	DESCRIPTION	UNITS
signal[n]	EQUAL OUT BOOLEAN			Boolean variable array	-

The EQUAL OUT type for the variable especifies that it is possible to connect an output port to multiple input ports and the values of the signals of the connected ports are equal.

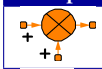
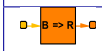
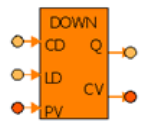
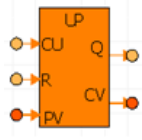
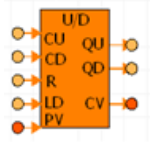
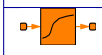
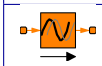
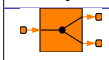
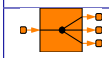
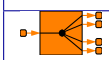
2.3 Component Types

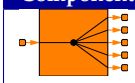
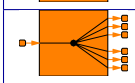
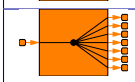
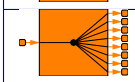
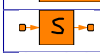
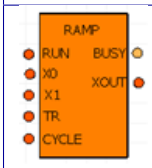
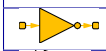
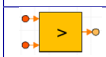
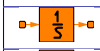
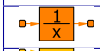
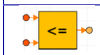
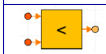
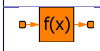
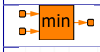
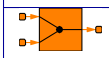
The following tables list the names and a brief description of the components available in the CONTROL library.

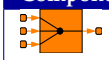
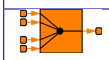
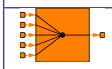
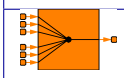
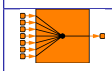
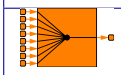
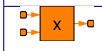
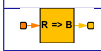
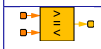
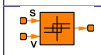
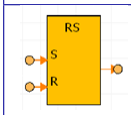
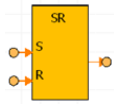
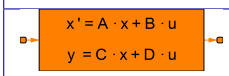
Abstract Components

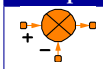
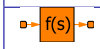
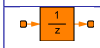
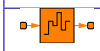
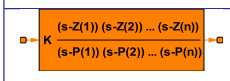
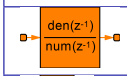
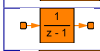
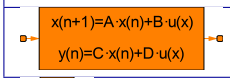
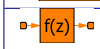
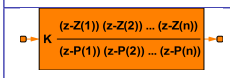
Component Name	Description
ASensor	Abstract sensor to derive sensors using multiple inheritance
Controller	Abstract definition of Controllers
Demux	Abstract definition of demultiplexors
MI2MOs	Control component with two multiple inputs analog ports and one multiple output analog port of the same dimension
MIMO	Control component with one multiple input analog port and one multiple output analog port of different dimension
MIMOs	Control component with one multiple input analog port and one multiple output analog port of the same dimension
MO	Control component with one multiple output analog port
Mux	Abtract definition of multiplexors
SI2SO	Control component with two single input analog signal port and one single output analog signal port
SI2bSO	Control component with two single input analog signal port and one single output Boolean signal port
SISO	Control component with one single input analog signal port and one single ouput analog port
SO	Control component with one single output analog signal port
Sampler	Control component that samples the input signals
bMI2MOs	Control component with two multiple input Boolean signal ports and one multiple output Boolean signal port of the same dimension
bMIMO	Control component with one multiple input Boolean signal port and one multiple output Boolean signal port of different dimensions
bMIMOs	Control component with one multiple input Boolean signal ports and one multiple output Boolean signal port of the same dimension
bMO	Control component with one multiple output Boolean signal port
bSI2SO	Control component with two single input Boolean signal ports and one single output Boolean signal port
bSISO	Control component with one single input Boolean signal ports and one single output Boolean signal port
bSO	Control component with one single output Boolean signal port
dMIMO	Control component with one multiple input analog signal port and one multiple output analog signal port with different dimension where the input signals are sampled
dMIMOs	Control component with one multiple input analog signal port and one multiple output analog signal port with the same dimension where the input signals are sampled
dSISO	Control component with one single input analog signal port and one single output analog signal port where the input signals are sampled

Operative Components

Component Symbol	Component Name	Description
	Addition	Control component that generates the sum of two input signals
	AnalogSource	Control component that can generate different waveforms
	BooleanToReal	Control component that converts a Boolean signal to an analog signal
	ButterLow	Control component that filters the input signals with a low pass Butterworth filter of any order
	CLK	Control component to generate a boolean-type signal to synchronize digital components.
	Clock	Control component that provides the simulation time
	Cntrl_DigitalPID	Control component that represents a digital proportional integral derivative (PID) controller
	Cntrl_P	Control component that represents a standard proportional (P) controller
	Cntrl_PI	Control component that represents a standard proportional integral (PI) controller
	Cntrl_PID	Control component that represents a standard proportional integral derivative (PID) controller
	Counter_DOWN	Control component that represents a digital down-counter
	Counter_UP	Control component that represents a digital up-counter
	Counter_UP-DOWN	Control component that represents a digital up-counter
	CritDamping	Control component that filters the input signal with an n-th order filter with critical damping
	DeadZone	Control component that provides a region of zero output
	Delay	Control component that delays the input signal by a fixed time interval
	Demux2	Control component that separates the input vector signal into two output signals
	Demux3	Control component that separates the input vector signal into three output signals
	Demux4	Control component that separates the input vector signal into four output signals

Component Symbol	Component Name	Description
	Demux5	Control component that separates the input vector signal into five output signals
	Demux6	Control component that separates the input vector signal into six output signals
	Demux7	Control component that separates the input vector signal into seven output signals
	Demux8	Control component that separates the input vector signal into eight output signals
	Derivative	Control component that computes the time derivative of the input signal
	Digital_ramp	Control component that generates a time-based ramp
	Division	Control component that implements the division function
	EQT	Control component that represents the standard comparison 'equal than'
	FirstOrder-Hold	Control component that implements a first order sample and hold latch
	Gain	Control component that multiplies the input signal by a constant
	Gate_AND	Control component that performs the logical operation AND between the two input signals
	Gate_NOT	Control component that performs the logical operation NOT in the input signal
	Gate_XOR	Control component that performs the logical operation XOR between the two input signals
	Gate_OR	Control component that performs the logical operation OR between the two input signals
	GET	Control component that represents the function 'greater or equal to'
	GT	Control component that represents the function 'greater than'
	Hysteresis	Control component that implements boolean hysteresis.
	Integrator	Control component that integrates the input signal
	Inverse	Control component that calculates the inverse of the input signal
	Logical-Switch	Control component that swithches depending on the boolean port (the middle one)
	LET	Control components that implements the function 'lower or equal to'
	LT	Control components that implements the function 'lower than'
	MathFunc-tion	Control component that performs basic mathematical functions to the input signals
	Maximum	Control component that calculates the maximum value between two input signals
	Minimum	Control component that calculates the minimum value between two input signals
	Module	Control component that calculates the module of a division of two input signals
	Move	Control component that represents the standard arithmetic operation MOVE
	Mux2	Control component that combines two input signals into a vector output port

Component Symbol	Component Name	Description
	Mux3	Control component that combines three input signals into a vector output port
	Mux4	Control component that combines four input signals into a vector output port
	Mux5	Control component that combines five input signals into a vector output port
	Mux6	Control component that combines six input signals into a vector output port
	Mux7	Control component that combines seven input signals into a vector output port
	Mux8	Control component that combines eight input signals into a vector output port
	NEQ	Control component that represents the function 'non equal to'
	Product	Control component that calculates the product between the two input signals
	Random-Source	Control component that generates normally distributed random numbers
	RealToBoolean	Control component that converts a analagog signal into a Boolean signal
	RelationalOp-erator	Control component that performs specified relational operation on the input signal
	Relay	Control component that switches the output signal between two values
	RS_Bistable	Control component that represents a reset dominant biestable
	Saturation	Control component that limits excursion of a signal
	Scaling	Control component to scale the input signal to a range
	Selector	Control component that extracts signals from an input signal port
	SourceChirp	Control component that generates sine wave with increasing frequency
	SourcebStep	Control component that generates Boolean step signals
	SourceDataFile	Control component that generates an output signal interpolating in a table loaded from an external ASCII file
	SourceExp	Control component that generates a rising and falling exponential analog signal
	SourceExpSine	Control component that generates exponentially damped sine analog signals
	SourcebCon-stant	Control component that generates Boolean constant signals
	SourcebPulse	Control component that generates Boolean pulse signals
	SourcebSam-pleTrigger	Control component that generates sample triggers
	SourcebStep	Control component that generates Boolean step signals
	SR_Bistable	Control component that represents a set dominant bistable
	StateSpace	Control component that implements a linear state-space system

Component Symbol	Component Name	Description
	Subtraction	Control component that performs the subtraction between the two input analog signals
	Switch	Control component that switches between two input signals
	TF_1stOrder	Control component that defines a first order transfer function between the input signal and the output signal
	TF_- 2ndOrder	Control component that defines a second order transfer function between the input signal and the ouput signal
	Timer_- TOFF	Control component that represents a timer to delay a signal on deactivation
	Timer_TON	Control component that represents a timer to delay a signal on activation
	Timer_TP	Control component that represents a timer to delay a signal on deactivation
	Transfer- Function	Control component that implements a linear transfer function
	UnitDelay	Control component that delays a signal one sample period
	VarDelay	Control component that delays the input signals by a variable amount of time
	ZeroOrder- Hold	Control component that implements a zero-order hold of one sample period
	ZeroPole	Control component that implements a transfer function specified in terms of poles and zeros
	dFilter	Control component that implements a discrete filter
	dIntegrator	Control component that performs a discrete-time integration of a signal
	dStateSpace	Control component that implements a discrete state-space system
	dTransfer- Function	Control component that implements a discrete transfer function
	dZeroPole	Control component that implements a discrete tranfer function in terms of poles and zeros

3. Abstract Components

3.1 ASensor

3.1.1 Description

This component is used for deriving sensors by inheritance.

3.1.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

3.1.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

3.1.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
bias[n_out]	REAL		Bias	-
gain[n_out]	REAL		Gain	-

3.1.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
var[n_out]	REAL		Measured variable	-

3.1.6 Formulation

The value of the output signals is calculated by applying the sensor gain and the sensor bias to the measured variable:

$$s\_out.signal[i] = gain[i] \cdot \mathrm{var}[i] + bias[i]$$

3.2 Controller

3.2.1 Description

This abstract component is used for deriving controllers of different types by inheritance.

3.2.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.2.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n)	OUT	Controlled output signal
s_set	analog_signal	(n = n)	IN	Set point signal
s_var	analog_signal	(n = n)	IN	Controlled variable signal

3.2.4 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
e[n]	REAL		Input error	-
r[n]	REAL		Set point	-
u[n]	REAL		Output	-
y[n]	REAL		Measured variable	-

3.2.5 Formulation

The set-point and the measured variable are equivalent to the s_set.signal and the s_var.signal respectively:

```
r[i] = s_set.signal[i]
y[i] = s_var.signal[i]
```

The error is calculated as follows:

```
e[i] = r[i] - y[i]
```

The variable u is equivalent to the s_out.signal:

```
u[i] = s_out.signal[i]
```

3.3 Demux

3.3.1 Description

This abstract component is used for deriving demultiplexer of different number of outputs by inheritance.

3.3.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals
n_out	CONST INTEGER	1	Number of outputs

3.3.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n_out * dim_out)	IN	Inlet port
s_out[n_out]	analog_signal	(n = dim_out)	OUT	Outlet ports

3.3.4 Formulation

The elements of the input signal vector is divided in element set of the same dimension depending the number of the output signals. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

s_out[i].signal[j] = element j of the outlet port i

s_in.signal[j] = element j of the inlet port

3.4 MI2MOs

3.4.1 Description

This abstract component is used for deriving components with two multiple input analog ports and one multiple output analog port of the same dimension.

3.4.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.4.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	Inlet signal
s_in_2	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

3.5 MIMO

3.5.1 Description

This abstract component is used for deriving components with one multiple input analog port and one multiple output analog port. The dimension of the input port signals can be different to the dimension of output port signals.

3.5.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

3.5.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n_in)	IN	Inlet signal
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

3.6 MIMOs

3.6.1 Description

This abstract component is used for deriving components with one multiple input analog port and one multiple output analog port. The dimension of the input port signals and the output port signals are the same.

3.6.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.6.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

3.7 MO

3.7.1 Description

This abstract component is used for deriving components with only one multiple output analog port.

3.7.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

3.7.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

3.8 Mux

3.8.1 Description

This abstract component is used for deriving multiplexers of different number of inputs by inheritance.

3.8.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals
n_in	CONST INTEGER	1	Number of inputs

3.8.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet ports
s_out	analog_signal	n=n_in*dim_in	OUT	Outlet port

3.8.4 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

s_in[i].signal[j] = signal j of the input port i

s_out.signal[j] = signal j of the output port

3.9 SI2SO

3.9.1 Description

This abstract component inherited from the abstract component MI2MO. It is used for deriving components with two single input analog ports and one single output analog port.

3.9.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	First inlet signal
s_in_2	analog_signal	(n = n)	IN	Second inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

3.9.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.10 SI2bSO

3.10.1 Description

This abstract class is used for the definition of a control component with two single analog input ports and one single boolean output port.

3.10.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	First inlet signal
s_in_2	analog_signal	(n = n)	IN	Second inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

3.10.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimensions of inputs and outputs

3.11 SISO

3.11.1 Description

This abstract component inherits from the abstract component MIMOs. It is used for deriving components with one single input analog port and one single output analog port.

3.11.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

3.11.3 Closed Parameters

NAME	TYPE	VALUE	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.12 SO

3.12.1 Description

This abstract component inherits from the abstract component MO. It is used for deriving components with only one single output analog port.

3.12.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

3.12.3 Closed Parameters

NAME	TYPE	VALUE	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

3.13 Sampler

3.13.1 Description

This abstract component is used for deriving components that needs to sample the input signals.

3.13.2 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s

3.13.3 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE	Boolean variable to know the sample time instant	-

3.13.4 Formulation

To sample at time 0 the following EcosimPro Language (EL language) WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

The Boolean variable sample is set to FALSE when it is detected that its value is TRUE and it is again set to TRUE after passing dt seconds.

3.14 bMI2MOs

3.14.1 Description

This abstract component is used for deriving components with two multiple input Boolean ports and one multiple output Boolean port. The inlet ports and the outlet ports have the same dimension.

3.14.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.14.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	bool_signal	(n = n)	IN	First inlet signal
s_in_2	bool_signal	(n = n)	IN	Second inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

3.15 bMIMO

3.15.1 Description

This abstract component is used for deriving components with one multiple input Boolean port and one multiple output Boolean port. The dimension of input port signals can be different to the dimension of output port signals.

3.15.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

3.15.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	bool_signal	(n = n_in)	IN	Inlet signal
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

3.16 bMIMOs

3.16.1 Description

This abstract component is used for deriving components with one multiple input Boolean port and one multiple output Boolean port. The dimension of the input port signals and the output port signals are the same.

3.16.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
N	CONST INTEGER	1	Dimension of inputs and outputs

3.16.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	bool_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

3.17 bMO

3.17.1 Description

This abstract component is used for deriving components with one multiple output Boolean port.

3.17.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

3.17.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

3.18 bSI2SO

3.18.1 Description

This abstract component inherits from the abstract component bMI2MOs. It is used for deriving components with two single input Boolean ports and one single output Boolean port.

3.18.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	bool_signal	(n = n)	IN	First inlet signal
s_in_2	bool_signal	(n = n)	IN	Second inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

3.18.3 Closed Parameters

NAME	TYPE	VALUE	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.19 bSISO

3.19.1 Description

This abstract component inherits from the abstract component bMIMOs. It is used for deriving components with one single input Boolean port and one single output Boolean port.

3.19.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	bool_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

3.19.3 Closed Parameters

NAME	TYPE	VALUE	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.20 bSO

3.20.1 Description

This abstract component inherits from the abstract component bMO. It is used for deriving components with one single output Boolean port.

3.20.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

3.20.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.21 dMIMO

3.21.1 Description

This abstract component inherits from the abstract component Sampler. It is used for deriving components with one multiple input analog port and one multiple output analog port. The input signals are sampled at each sample time period dt. The dimension of input port signals can be different to the dimension of output port signals.

3.21.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

3.21.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n_in)	IN	Inlet signal
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

3.21.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s

3.21.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE	Boolean variable to know when the input signals must be sampled	

3.21.6 Formulation

To sample at time 0, the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

3.22 dMIMOs

3.22.1 Description

This abstract component inherits from the abstract component Sampler. It is used for deriving components with one multiple input analog port and one multiple output analog port. The input signals are sampled at each sample time period dt. The dimension of input port signals and the dimension of output port signals are the same.

3.22.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.22.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal ()
s_out	analog_signal	(n = n)	OUT	Outlet signal ()

3.22.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s

3.22.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE		

3.22.6 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

3.23 dSISO

3.23.1 Description

This abstract component inherits from the abstract component dMIMOs. It is used for deriving components with one single input analog port and one single output analog port. The input signal is sampled at each sample time period dt.

3.23.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

3.23.3 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s

3.23.4 Closed Parameters

NAME	TYPE	VALUE	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

3.23.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE		

3.23.6 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

This page intentionally left blank.

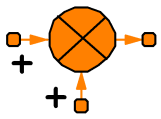
4. CONTROL Components

4.1 Addition

4.1.1 Description

This component is inherited from component MI2MOs. It represents an adder unit tipically employed in control systems. It calculates an output signal that is the algebraic sum of the two input signals.

4.1.2 Symbol



Addition

4.1.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.1.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	Inlet signal
s_in_2	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.1.5 Formulation

The output signals responds instantaneously to changes of the input signals. The equation associated to this component is the following:

FOR(i =0; i<=n; i = i+1)

s_out.signal[i] = s_in_1.signal[i] + s_in_2.signal[i]

4.2 AnalogSource

4.2.1 Description

This component is inherited from component MO. It represents a signal generator and can produce one of the following different waveforms:

- Constant signal
- Sine signal
- Pulse signal
- Step signal
- Sawtooth signal
- Square signal
- Ramp signal
- Signal by linear interpolation with detection of events in a data table
- Signal by linear interpolation without detection of events in a data table
- Signal by spline interpolation in a table of data
- Step signal generated by interpolation in a table of data

This component is vectorized that means that the ouput signal is a vector of signals of the same type and with the same parameters.

4.2.2 Symbol

AnalogSource

4.2.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.2.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.2.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Amp	REAL	1	Signal amplitude or height	-
Offset	REAL	0	Offset of output signal	-
Period	REAL	10	Period of sine, pulse, sawtooth, or square source	s
Phase	REAL	0	Phase of sine source	rad
Tstart	REAL	0	Starting time of signal generation	s
pulseWidth	REAL	0.001	Pulse width of pulse wave	s
rampDuration	REAL	10	Duration of the ramp	s
source	ENUM SourceOption	Source_Constant	Waveform	
tabmehod	ENUM tableMethod	LinearInterpWith-Events	Method to interpolate or connect the table points	
timeTable	TABLE_1D		Table for table source	-

4.2.6 Guidelines

The user defines the type of signals changing the value of the data source. Not all the data are applicable to a specific type of source. The following data are applicable to all the types of sources:

- Amp: Signal amplitude or height
- Offset: Offset of the output signals
- Tstart: Start time of the signal, for time less than this data, the ouput is set to the offset value

The rest of the data is applicable depending on the type of source, this is explained in the description of the corresponding data.

The type of source that the user can define is shown in the following table:

source option	Description
Source_Constant	To generate constant signals
Source_Sine	To generate sinusoidal signals
Source_Pulse	To general pulse signals
Source_Step	To generate step signals
Source_Square	To generate square signals
Source_Ramp	To generate ramp signals
Source_Table	To generate signals by interpolation in a table (linear, splines, steps)

If the user defines the source type called Source_Table, then the user can specify the method of interpolating in the table (or connecting the points of the table) with the enumerative data type named tabmethod. There are four methods listed in the following table:

Method	Description
LinearInterpWithEvents	Linear interpolation with event detection
LinearInterpWithoutEvents	Linear interpolation without event detection
SplineInterp	Spline interpolation
StepConnect	Connection of the table points by means of steps

The output signals will depend on the interpolation method chosen by the user.

4.2.7 Formulation

As we have explained above the output signals depends on the value of the parameter source defined by the user. In the following table the expressions used to calculate the ouput signals are depicted depending on the type of source choosen.

source option	Equation	
Source_Constant	Amp + Offset	
Source_Sine	Amp * sin(2 * PI /Period * (TIME - Tstart) + Phase) + Offset	
Source_Pulse	pulse(TIME-Tstart, Period, pulseWidth) + Offset	
Source_Step	Amp * step(TIME, Tstart) + Offset	
Source_SawTooth	Amp * ramp(TIME-Tstart, Period) / Period + Offset	
Source_Square	Amp * square(TIME-Tstart, Period) + Offset	
Source_Ramp	Amp / rampDuration * (TIME - Tstart) * (1- step(TIME, Tstart + rampDuration)) + Amp * step(TIME, Tstart + rampDuration) + Offset	
Source_Table	tabmehod	Equation
	LinearInterpWithEvents	Amp * timeTableInterp(TIME-Tstart, timeTable) + Offset
	LinearInterpWithoutEvents	Amp * linearInterp1D(timeTable, TIME-Tstart) + Offset
	SplineInterp	Amp * splineInterp1D (timeTable ,TIME-Tstart) + Offset
	StepConnect	Amp * timeTableStep(TIME-Tstart, timeTable) + Offset

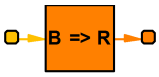
These expressions are applied when the simulation time is greater than Tstart, if not the output signals are equal to Offset.

4.3 BooleanToReal

4.3.1 Description

This component transforms Boolean signals to analog signals. The user defines the value of the output signals when the input signals are TRUE or FALSE.

4.3.2 Symbol



BooleanToReal

4.3.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs signal vectors

4.3.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	bool_signal	(n = n)	IN	Boolean inlet signal
s_out	analog_signal	(n = n)	OUT	Analog outlet signal

4.3.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
ValueFalse	REAL	0	Output analog signal for FALSE Boolean input	-
ValueTrue	REAL	1	Output analog signal for TRUE Boolean input	-

4.3.6 Formulation

If the Boolean input signal is TRUE, the output analog signal takes the value of the data ValueTrue, otherwise the output analog signal takes the value of the data ValueFalse.

$$s\_out.signal[i] = \begin{cases} IF(s\_in.signal[i] = TRUE) & ValueTrue \\ ELSE & ValueFalse \end{cases}$$

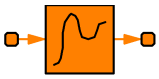
4.4 ButterLow

4.4.1 Description

This component is inherited from component SISO. The component defines the transfer function between the input signal and the output signal as an n_f-th order low pass filter with Butterworth characteristics and cut-off frequency defined by the data freq. It is implemented as a series of second order filters and a first order filter.

Butterworth or maximally flat lowpass filters have a monotonic amplitude frequency response which is maximally flat at zero frequency response and the amplitude frequency response decreases logarithmically with increasing frequency. The butterworth filter has minimal phase shift over the filter's band pass when it is compared to other conventional filters

4.4.2 Symbol



ButterLow

4.4.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_f	CONST INTEGER	2	Order of the filter

4.4.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.4.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
freq	REAL	1	Cut-off frequency	Hz

4.4.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
m	CONST INTEGER	(n_f+1)/2	

4.4.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
D[m]	REAL			
T	REAL			
k1	REAL			
k2[m]	REAL			
poleimag[m]	REAL		Imaginary part of the poles	
polereal[m]	REAL		Real part of the poles	
realpol	REAL			
w	REAL			
w0[m]	REAL			
x1[m]	REAL		States 1 of second order filters	
x2[m]	REAL		States 2 of second order filters	
xr	REAL		State of real pole for uneven order otherwise dummy	
z[m + 1]	REAL			

4.4.8 Formulation

Butterworth poles lie along a circle and are spaced at equal angular distances around a circle.

The nth order Butterworth filter will have n poles and no finite zeros. The pole locations correspond to the Butterworth response are defined by the equation:

$$1 + \left(-s^2\right)^n = 0$$

The pole locations so defined are located on the unit circle in the s plane and are symetrical to the real axis. For example: for an order of the filter of 5, all the poles will be:

-1.0000, -0.8090 ± 0.5878 i, - 0.3090 ± 0.9511, 0.3090 ± 0.9511, 0.8090 ± 0.5878 i, 1.0000

The angular frequency is calculated as follows:

$$\omega = 2 \cdot \pi \cdot f$$

where f is the cut-off frequency.

The imaginary part and the real part of the poles are calculated with the following expressions:

$$FOR(i = 0; i \le m; i++)$$
$$polereal[i] = \cos\left(\pi / 2 + \pi / n_f \cdot (i - 0.5)\right)$$
$$poleimag[i] = \sin\left(\pi / 2 + \pi / n_f \cdot (i - 0.5)\right)$$

where nf is the order of the filter and m is equal to (nf+1)/2

From these value the second order filter coefficients are calculated:

$$FOR(i = 0; i \le m; i++)$$
$$w_o[i] = \left(polereal[i]^2 + poleimag[i]^2\right) \cdot \omega$$
$$D[i] = \frac{-polereal[i]}{w_o[i]} \cdot w$$

where:

wo = Angular frequency in the second order filters

D[i] = Damping in the second order filters

The output signals depends on the order of the filter (even or odd)

Even order

$$FOR(i=0;i\le m;i++)$$
$$\frac{\delta x_1[i]}{\delta t}=x_2[i]$$
$$\frac{\delta x_2[i]}{\delta t}=\omega_o[i]^2\cdot z[i]-2\cdot D[i]\cdot\omega_o[i]\cdot x_2[i]-\omega_o[i]^2\cdot x_1[i]$$
$$z[i+1]=x_1[i]$$
$$FOR\ i=1 \qquad z[1]=s\_in.signal[i]$$
$$s\_out.signal[1]=z[m+1]$$

Odd order

$$FOR(i=0;i\le m-1;i++)$$
$$\frac{\delta x_1[i]}{\delta t}=x_2[i]$$
$$\frac{\delta x_2[i]}{\delta t}=\omega_o[i]^2\cdot z[i]-2\cdot D[i]\cdot\omega_o[i]\cdot x_2[i]-\omega_o[i]^2\cdot x_1[i]$$
$$z[i+1]=x_1[i]$$
$$FOR\ i=1 \qquad z[1]=s\_in.signal[i]$$
$$z[m+1]=z[m]$$
$$\frac{\delta x_r}{\delta t}=\frac{k_1\cdot z[m+1]-x_r}{T}$$
$$s\_out.signal[1]=x_r$$

where k1 is equal to 1 and T is equal to $1/\omega$

4.5 CLK

4.5.1 Description

This component represents a clock generator. It generates a boolean-type clock signal to synchronize digital components of the library. It manages the value of the global variable Clock_signal

4.5.2 Symbol

4.5.3 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Period	REAL	10	Period of pulse	s
pulseWidth	REAL	5	Pulse width	s
Tstart	REAL	0	Starting time of signal generation	s

4.5.4 Guidelines

Circuits using the clock signal for synchronization may become active at either the rising edge, falling edge or, in the case of double data rate, both at the rising and at the falling edges of the clock cycle.

4.5.5 Formulation

This component is modelled by means of discrete events that change the value of the global variable Clock_-signal to generate a signal with the period and width defined by the user. The defined events are the following:

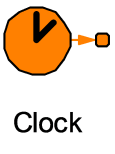
```
WHEN(TIME > Tstart) THEN
   State = TRUE
   Clock_signal = TRUE
END WHEN
WHEN(State) THEN
   State = FALSE AFTER pulseWidth
   Clock_signal = FALSE AFTER pulseWidth
END WHEN
WHEN(State==FALSE AND TIME > Tstart) THEN
   State = TRUE AFTER (Period - pulseWidth)
   Clock_signal = TRUE AFTER (Period - pulseWidth)
END WHEN
```

4.6 Clock

4.6.1 Description

This component is inherited from component MO. It generates output signals equal to the simulation model time plus an offset.

4.6.2 Symbol



4.6.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.6.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal ()

4.6.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Offset	REAL	0	Offset of output signal	-

4.6.6 Guidelines

4.6.7 Formulation

This component has only an equation that makes the value of the output signal equal to the simulation time plus an offset defined by the user:

```
   State = FALSE
   Clock_signal = FALSE
DISCRETE
   WHEN(TIME > Tstart) THEN
      State = TRUE
      Clock_signal = TRUE
   END WHEN
   WHEN(State) THEN
      State = FALSE AFTER pulseWidth
      Clock_signal = FALSE AFTER pulseWidth
   END WHEN
   WHEN(State==FALSE AND TIME > Tstart) THEN
      State = TRUE AFTER (Period - pulseWidth)
      Clock_signal = TRUE AFTER (Period - pulseWidth)
   END WHEN
```

4.7 Cntrl_DigitalPID

4.7.1 Description

This component is inherited from component Controller. It represents a digital controller of the PID type.

4.7.2 Symbol

Cntrl_DigitalPID

4.7.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n)	OUT	Controlled output signal
s_set	analog_signal	(n = n)	IN	Set point signal
s_var	analog_signal	(n = n)	IN	Controlled variable signal

4.7.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
eps[1]	REAL		Minimum step of the output signal	-
eps_in[1]	REAL		Dead band of the input signal	-
input_bits[1]	INTEGER		Size of the input signal	bits
max_input[1]	REAL		High limit of input	-
max_output[1]	REAL		High limit of output	-
min_input[1]	REAL		Low limit of input	-
min_output[1]	REAL		Low limit of output	-
output_bits[1]	INTEGER		Size of the output signal	bits
output_i[1]	REAL		Initial output value	-
period[1]	REAL		Sampling Period	s
prop_gain[1]	REAL		Controller proportional gain	-
rate_time[1]	REAL		Rate or derivative time	-
reset_time[1]	REAL		Reset or integral time	-

4.7.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.7.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
e[n]	REAL		Input error	-
error0[1]	REAL		Input error at time t	-
error1[1]	REAL		Input error at time t-dt	-
error2[1]	REAL		Input error at time t-2*dt	-
frac[1]	REAL		Period fraction	s
input_set[1]	REAL		Quantizied set point signal	-
input_var[1]	REAL		Quantizied variable signal	-
int_out0[1]	REAL		Integral output at time t	
int_out1[1]	REAL		Integral output at time t - dt	
new_out[1]	REAL			
output0[1]	REAL		Output at time t	-
output1[1]	REAL		Output at time t-dt	-
prop_out[1]	REAL		Proportional output	
r[n]	REAL		Set point	-
rate_out[1]	REAL			
start[1]	REAL			
step_size_input[1]	REAL			
step_size_output[1]	REAL			
u[n]	REAL		Output	-
y[n]	REAL		Measured variable	-

4.7.7 Guidelines

4.7.8 Formulation

At each sample time, the errors at previous times are passed to variables that store old values

$$\varepsilon \text{ t-}2\Delta\text{t} = \varepsilon \text{ t-}\Delta\text{t}$$

$$\varepsilon \text{ t-}\Delta\text{t} = \varepsilon$$

The positive and negative input signals are quantizied taking into account the number of bits of the input signal with the function quantzr.

input_pos = quantzr(step_size_input, s_in_pos.signal)

input_neg = quantzr(step_size_input, s_in_neg.signal)

ε t = input_pos - input_neg

A dead band is applied to the input error:

$$\varepsilon^t = \begin{cases} \varepsilon^t - \varepsilon_{db} \text{ if } \varepsilon^t > \varepsilon_{db} \\ \varepsilon^t + \varepsilon_{db} \text{ if } \varepsilon^t < -\varepsilon_{db} \\ 0 \text{ if } -\varepsilon_{db} \le \varepsilon^t \le \varepsilon_{db} \end{cases}$$

The proportional, derivative, and integral part of the controller output are calculated respectively with the following equations:

$$s_{prop} = K_G \cdot \varepsilon^t$$

$$s_{rate} = K_G \cdot T_D \left(\frac{\varepsilon^t - \varepsilon^{t-\Delta t}}{T_S} \right)$$

$$s_{int}^t = s_{int}^{t-\Delta t} + \frac{K_G \, T_S}{2 T_R} \left(\varepsilon^t + \varepsilon^{t-\Delta t} \right)$$

The new output is calculated as:

$$s_{new}^t = s_{prop}^t + s_{rate}^t + s_{int}^t$$

The output is limited to avoid control saturation according to the following formulas:

if $s_{new}^t > s_{\max}$ then

$$s_{out}^t = s_{\max}$$

$$s_{int}^t = s_{int}^{t-\Delta t} + \frac{T_S}{(Ts + T_R)} \left(s_{\max} - s_{int}^{t-\Delta t} \right)$$

end if

if $s_{new}^t < s_{\min}$ then

$$s_{out}^t = s_{\min}$$

$$s_{int}^t = s_{int}^{t-\Delta t} + \frac{T_S}{(Ts + T_R)} \left(s_{\min} - s_{int}^{t-\Delta t} \right)$$

end if

If the output signal is between the limits, a dead band is applied to the control signal:

if $s_{\min} < s_{new}^t < s_{\max}$ then

if $\left| s_{new}^t - s_{out}^{t-\Delta t} \right| > \delta_{db}$ then

$$s_{out}^t = s_{new}^t$$

$$s_{out}^{t-\Delta t} = s_{out}^t$$

end if

end if

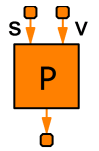
A last correction is applied to the output signal the quantization error that depends on the number of bits of the output signal.

4.8 Cntrl_P

4.8.1 Description

This component is inherited from component Controller. This component type named Cntrl_p, represents a standard proportional (P) control. It obtains an error signal from an input signal and a set-point and produces an output signal consisting of the error signal multiplied by a proportional gain.

4.8.2 Symbol



Cntrl_P

4.8.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.8.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n)	OUT	Controlled output signal
s_set	analog_signal	(n = n)	IN	Set point signal
s_var	analog_signal	(n = n)	IN	Controlled variable signal

4.8.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
k[n]	REAL		Proportional gain of the controller	-
u_max[n]	REAL		High limit of output	-
u_min[n]	REAL		Low limit of output	-

4.8.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
e[n]	REAL		Input error	-
r[n]	REAL		Set point	-
u[n]	REAL		Output	-
v[n]	REAL		Unbounded output	-
y[n]	REAL		Measured variable	-

4.8.7 Guidelines

The controller gain, k, can be a positive number or a negative number. When k>0, the controller is direct-acting. In this case, the controller output decreases if the controlled variable increases. A typical case of direct-acting control is the temperature control of heat exchanger, because if the outlet temperature increases, the controller must close the vapour inlet valve to reduce the heating power.

If k<0, the controller is inverse-acting, and the controller output increases if the controlled variable increases. A typical case is a level control where if the level of liquid in a tank increases due to an increase of the inlet flow, the controller must increase the position of the tank exit valve to re-establish the level.

4.8.8 Formulation

The two input signals to the controller are the set-point and the controlled process variable:

r = s_set.signal

y = s_var.signal

The error signal is equal to:

e = r - y

The unbounded proportional output is obtained multiplying the proportional gain of the controller and the error:

v = k e

The output is limited between u_max and u_min values:

$$u = \begin{cases} u & \textbf{for } u\_\min < u < u\_\max \\ u\_\min & \textbf{for } u < u\_\min \\ u\_\max & \textbf{for } u > u\_\max \end{cases}$$

s_out.signal = u

4.9 Cntrl_PI

4.9.1 Description

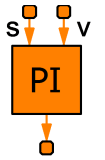
This component is inherited from component Controller. This component type, named Cntrl_pi, represents a standard proportional integral (PI) control. It obtains an error by taking the difference between two input signals, and it produces an output signal with two components: a proportional part and an integral part.

Any controller with integral action can exhibit a non-linear phenomenon called reset windup or integrator build-up when overdriven. If there is a large initial error, the integrator does not respond immediately, but it integrates the error and produces a large signal some time later. The integral control action continues to increase as long as the error has not changed sign even though the controlled output may be close to the desired value. If the error remains large for a sufficiently long time, the integral mode can cause saturation even when the current value of the error is small. Consequently, a large overshoot can result.

Most controllers have an anti-reset windup option that limits the contribution of integral action so that the addition of proportional output plus integral output does not exceed specified limits on either ends of the output range. The component type Cntrl_pi includes the anti-reset windup option of current controllers, a negative feedback that modifies the integrator input to prevent the windup.

The gamma datum is the ratio between the time constant for the antiwindup and the integration time. The time constant for the antiwindup is the time constant of the negative feedback applied to the integrator demand in order to prevent windup. A heuristic rule for tuning gamma is to take gamma = 0.1.

4.9.2 Symbol



Cntrl_PI

4.9.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.9.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n)	OUT	Controlled output signal
s_set	analog_signal	(n = n)	IN	Set point signal
s_var	analog_signal	(n = n)	IN	Controlled variable signal

4.9.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Ti[n]	REAL		Integrator time or reset time	s
beta[n]	REAL	0,1	Weight factor for set point changes in P output	-
end_pos	ENUM EndPosBehaviour	end_-I	End position behaviour	-
gamma[n]	REAL		Ratio between the time constant for the antiwindup and the integration time	-
k[n]	REAL		Proportional gain of the controller	-
u_max[n]	REAL		High limit of output	-
u_min[n]	REAL		Low limit of output	-

4.9.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
e[n]	REAL		Input error	-
es[n]	REAL		Saturation error	-
r[n]	REAL		Set point	-
u[n]	REAL		Output	-
v[n]	REAL		Unbounded output	-
vi[n]	REAL		Integral part of the output	-
vil[n]	REAL		Limited integral part of the output	-
y[n]	REAL		Measured variable	-

4.9.7 Guidelines

Controller Gain, k

The controller gain, k, can be a positive number or a negative number. When k>0, the controller is direct-acting. In this case, the controller output decreases if the controlled variable increases. A typical case of direct-acting control is the temperature control of heat exchanger, ie if the outlet temperature increases, the controller closes the vapour inlet valve to reduce the heating power.

If k<0, the controller is inverse-acting, and the controller output increases if the controlled variable increases. A typical case is a level control where if the level of liquid in a tank increases due to an increase of the inlet flow, the controller must increase the position of the tank exit valve to re-establish the level.

End Position Behaviour, end_pos

This data represents the end position behaviour, it is an enumeration type (see Section 2.1), and can take two values

I (for integral end position behaviour)

PI (for proportional-integral end position behaviour)

In the following example (see Figure 1, Figure 2 & Figure 3), a change of the error in the positive direction leads to a P-jump of the controller output in the case of PI end position behaviour. However, since the error is still negative, the controller integrates again to the end position. The resultant effect, e.g. brief opening of a valve, is in general undesirable, if the error remains negative. With I behaviour in the end position this effect is suppressed, because the controller carries out the P-jump only after a sign change of the error.

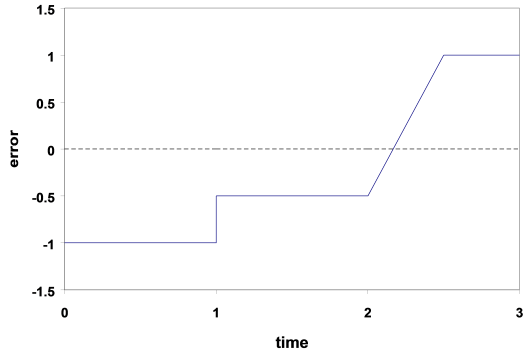


Figure 1-Controller error

The output with PI end position behaviour will be:

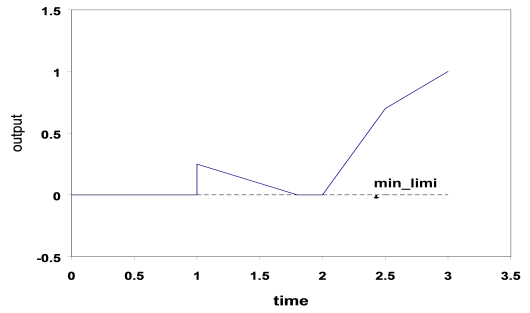


Figure 2-Controller output with PI behaviour

The output with I end position behaviour will be:

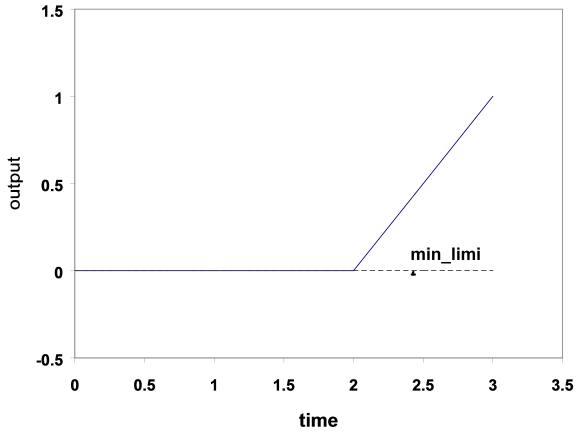


Figure 3-Controller output with I behaviour

Weight Factor for Set-point Changes, beta

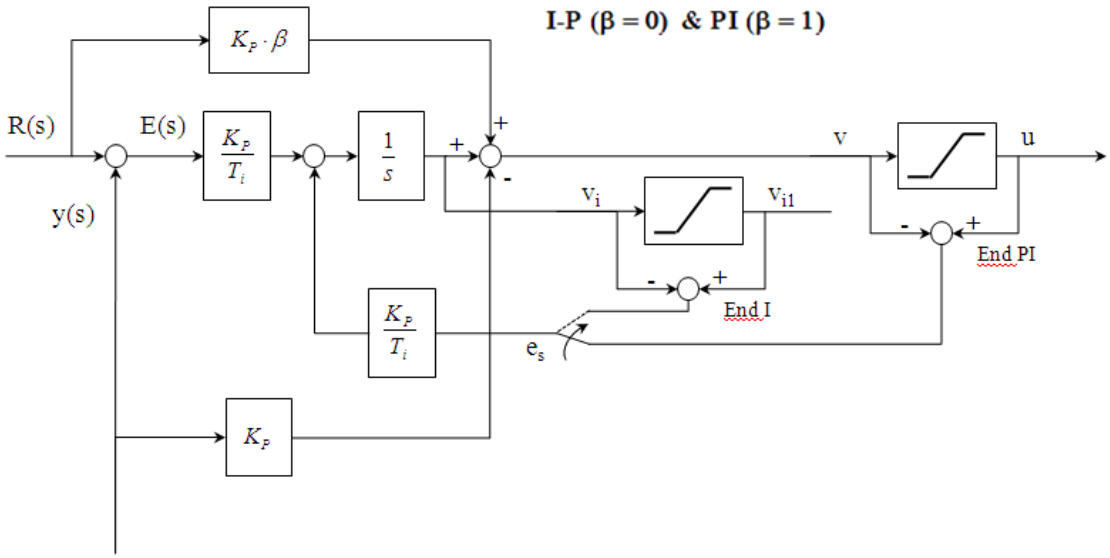
In many current controllers, the proportional part gain only acts over the measured process variables. This avoids high control signals when the set-point is suddenly changed. This controller architecture is called I-P. The following formula covers the two architectures the PI when β=1, and the I-P when β=0

$$u(t) = K\left(\beta r(t) - y(t) + \frac{1}{Ti}\int e(t)\,dt\right)$$

4.9.8 Formulation

Structure of the Controller

The component Cntrl_pi represents a non-interactive control structure PI with a weight factor β to avoid the effect of brusque changes of the set-point, and a feedback loop to avoid the effect of saturation on the integral action whose error signal is applied to the output of the integral action or to the output of the global action depending on whether EndPos behaviour is I or PI respectively and in accordance with the following control diagram (for a justification of this structure, see Appendix A).



Equations

The two input signals to the controller are the set-point and the controlled process variable:

r = s_set.signal

y = s_var.signal

The error signal is equal to:

e = r - y

The feedback into the integrator to avoid saturation is

$$es = \begin{cases} u - v & \textbf{for } end\ position\ behaviour = PI \\ vil - vi & \textbf{for } end\ position\ behaviour = I \end{cases}$$

The integral part of the output is calculated from

$$vi = \int \left(\frac{k}{Ti} \quad e + \frac{es}{Tr} \right) dt$$

The limited integral output is

$$vil = \begin{cases} vi & \textbf{for } u\_\min < vi < u\_\max \\ u\_\min & \textbf{for } vi < u\_\min \\ u\_\max & \textbf{for } vi > u\_\max \end{cases}$$

The unlimited output is the sum of the proportional output and the integral output

v = k (β r - y) + vi

The output is limited between u_max and u_min

$$u = \begin{cases} u & \text{for } u\_min < u < u\_max \\ u\_min & \text{for } u < u\_min \\ u\_max & \text{for } u > u\_max \end{cases}$$

s_out.signal = u

4.10 Cntrl_PID

4.10.1 Description

This component is inherited from component Controller. This component type, named Cntrl_pid, represents a standard proportional integral derivative (PID) control. It obtains an error signal from an input signal and a set-point and produces an output signal with three components: a proportional part, an integral part, and a derivative part.

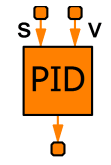
Any controller with integral action can exhibit a non-linear phenomena called reset windup or integrator build-up when overdriven. If there is a large initial error, the integrator does not respond immediately, but it integrates the error and produces a large signal some time later. The integral control action continues to increase as long as the error has not changed sign even if the controlled output may be close to the desired value. If the error remains large for a sufficiently long time, the integral mode can cause saturation even when the current value of the error is small. Consequently, a large overshoot can result.

Most controllers have an anti-reset windup option that limits the contribution of integral action so that the sum of the proportional output and the integral output does not exceed specified limits on either end of the output range. There is a negative feedback that modifies the integrator input error to prevent the windup.

Current PID controllers have a low-pass filter inserted in cascade with the differentiator. The reason for this filter is that the input signal always contains some high frequency noise, which has large slopes compared with the more slowly varying primary signal, and it would dominate the output of the differentiator. The low-pass filter inserted in cascade with differentiator solves the problem.

The component type Cntrl_pid includes the anti-reset windup option and the low-pass filter of current controllers.

4.10.2 Symbol



Cntrl_PID

4.10.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.10.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n)	OUT	Controlled output signal
s_set	analog_signal	(n = n)	IN	Set point signal
s_var	analog_signal	(n = n)	IN	Controlled variable signal

4.10.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Td[n]	REAL		Rate or derivative time	s
Ti[n]	REAL		Integrator time or reset time	s
alpha[n]	REAL	0,1	Derivative filter parameter	-
beta[n]	REAL	0,1	Weight factor for set point changes in P output	-
end_pos	ENUM EndPosBehaviour	end_-I	End position behaviour	-
gamma[n]	REAL		Ratio between the time constant for the antiwindup and the integration time	-
k[n]	REAL		Proportional gain of the controller	-
u_max[n]	REAL		High limit of output	-
u_min[n]	REAL		Low limit of output	-

4.10.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
e[n]	REAL		Input error	-
es[n]	REAL		Saturation error	-
r[n]	REAL		Set point	-
u[n]	REAL		Output	-
v[n]	REAL		Unbounded output	-
vi[n]	REAL		Integral part of the output	-
vil[n]	REAL		Limited integral part of the output	-
y[n]	REAL		Measured variable	-
yf[n]	REAL		Filtered variable	-

4.10.7 Guidelines

Controller Gain, k

The controller gain, k, can be a positive number or a negative number. When k>0, the controller is direct-acting. In this case, the controller output decreases if the controlled variable increases. A typical case of direct-acting control is the temperature control of heat exchanger, ie if the outlet temperature increases, the controller closes the vapour inlet valve to reduce the heating power.

If k<0, the controller is inverse-acting, and the controller output increases if the controlled variable increases. A typical case is a level control where if the level of liquid in a tank increases due to an increase of the inlet flow, the controller must increase the position of the tank exit valve to re-establish the level.

End Position Behaviour, end_pos

This data represents the end position behaviour, and it is of an enumeration type (see Section 2.1), and it can takes two values

end_I (for integral end position behaviour)

end_PI (for proportional-integral end position behaviour)

In the following example (see Figure 1, Figure 2 & Figure 3), a change of the error in the positive direction leads to a P-jump of the controller output in the case of PI end position behaviour. However, since the error is still negative, the controller integrates again to the end position. The resultant effect, e.g. brief opening of a

valve, is in general undesirable, if the error remains negative. With I behaviour in the end position this effect is suppressed, because the controller carries out the P-jump only after a sign change of the error.

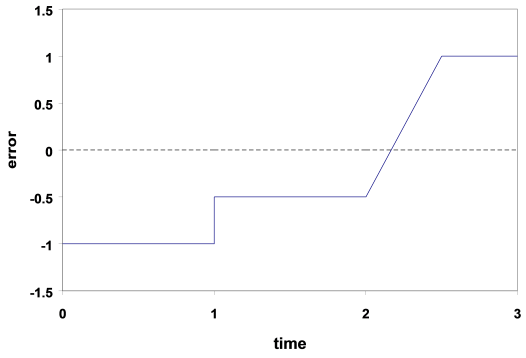


Figure 1-Controller error

The output with PI end position behaviour will be:

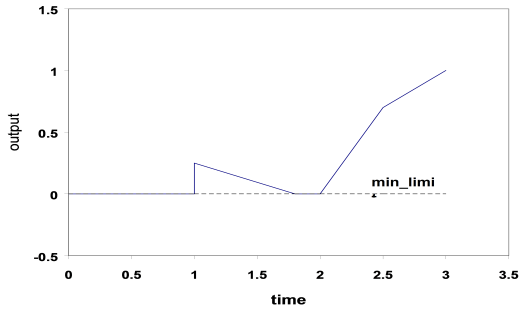


Figure 2-Controller output with PI behaviour

The output with I end position behaviour will be:

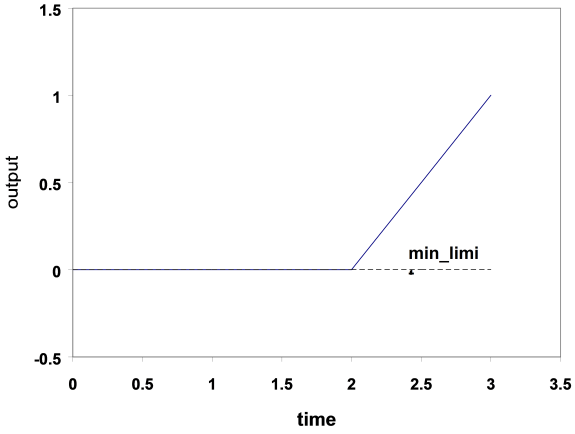


Figure 3-Controller output with I behaviour

Weight Factor for Set-point Changes, beta

In many current controllers, the proportional part gain only acts over the measured process variables. This avoids high control signals when the set-point is suddenly changed. This controller architecture is called I-P. The following formula covers the two architectures the PI when β, and the I-P when β=0

$$u(t) = K\left(\beta r(t) - y(t) + \frac{1}{Ti}\int e(t)\,dt\right)$$

Derivative Filter Factor, alpha

In order to limit high frequency outputs (i.e. noise) the following transfer function is used, which includes a low-pass filter where $\alpha < 1$ is the deriviative filter which takes typical values between 0.05 and 0.1:

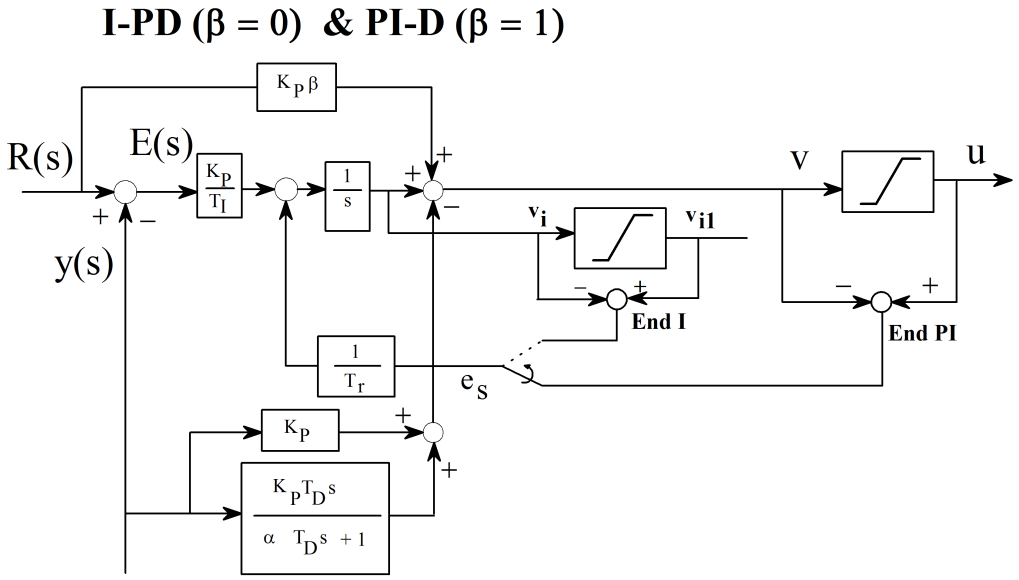
$$\frac{u_d(s)}{E(s)} = \frac{K_P T_D s}{\alpha T_D s + 1}$$

With this filter, the high frequency gain is limited to $\frac{K_P}{\alpha}$

4.10.8 Formulation

Structure of the Controller

The component Cntrl_pid has a non-interactive PID control structure with derivative filter factor α during derivative action to avoid noise amplification at low frequencies, a weight factor β to avoid the effects of brusque changes of the set-point, and a feedback loop to avoid the effects of saturation on the integral action. The anti-windup error signal is applied to the output of the integral action or of the global action depending on whether the end position behaviour is end_I or end_PI respectively, and in accordance with the following control diagram:



Equations

The two input signals to the controller are the set-point and the controlled process variable:

r = s_set.signal

y = s_var.signal

The error signal is equal to:

e = r - y

The correction to the error going into the integrator to avoid saturation is

$$es = \begin{cases} u - v & \textbf{for } end\ position\ behaviour = PI \\ vil - vi & \textbf{for } end\ position\ behaviour = I \end{cases}$$

The integral part of the output is calculated from

$$vi = \int \left(\frac{k}{Ti} \quad e + \frac{es}{Tr} \right) dt$$

The limited integral output is

$$vil = \begin{cases} vi & \textbf{for } u\_\min < vi < u\_\max \\ u\_\min & \textbf{for } vi < u\_\min \\ u\_\max & \textbf{for } vi > u\_\max \end{cases}$$

The output of the derivative filter is calculated from

$$\frac{d\,y_f}{dt} = \frac{\left(y - y_f\right)}{\alpha\, T_d}$$

The unlimited output is the sum of the proportional output, the integral output and the derivative output:

$$v = k\,(\beta\, r - y - T_d \frac{dy_f}{dt}\ ) + v_{i,\lim}$$

The output is limited between u_max and u_min:

$$u = \begin{cases} u & \textbf{for } u\_\min < u < u\_\max \\ u\_\min & \textbf{for } u < u\_\min \\ u\_\max & \textbf{for } u > u\_\max \end{cases}$$

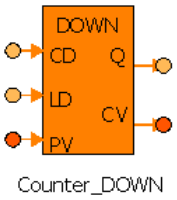
s_out.signal = u

4.11 Counter_DOWN

4.11.1 Description

This component is inherited from component SISO. It represents digital down-counter as defined in IEC 61131-3.

4.11.2 Symbol



4.11.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
CD	bool_signal	(n = 1)	IN	Counter activation
LD	bool_signal	(n = 1)	IN	Load signal
Q	bool_signal	(n = 1)	OUT	Output

4.11.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
PVmin	INTEGER	0	Minimun limit	-

4.11.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
PV	INTEGER	–	Maximum value the counter can count to	–
CV	INTEGER	–	Counter	–
t	INTEGER	0	Internal variable	–

4.11.6 Guidelines

The signal CD is a synchronous entry so the user must define a clock signal in the schematic diagram (CLK component). The CD signal becomes active at the rising edge of the clock signal.

The numerical values of the limit variable PVmin are implementation dependent.

4.11.7 Formulation

The component is implemented as a continuous equation that defines the value of CV as equal to the output and a set of discrete events defined as follows:

```
WHEN(Clock_signal == TRUE)THEN
   PV = s_in.signal[1]
   IF (CD.signal[1]==TRUE AND CV >PVmin AND CV > 0)THEN
      IF (t ==0) THEN
         CV = PV
         t += 1
      ELSE
         CV = CV -1
      END IF
   END IF
   IF(CV <= 0)THEN
      Q.signal[1] =TRUE
```

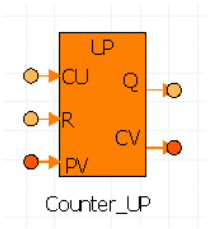
```
   ELSE
      Q.signal[1] = FALSE
   END_IF
END_WHEN
WHEN (LD.signal[1]== TRUE) THEN
   CV= PV
   t = 0
   Q.signal[1] = FALSE
END_WHEN
```

4.12 Counter_UP

4.12.1 Description

This component is inherited from component SISO. It represents a digital up-counter as defined in IEC 61131-3.

4.12.2 Symbol



4.12.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
CU	bool_signal	(n = 1)	IN	Counter activation
RESET	bool_signal	(n = 1)	IN	Reset signal
Q	bool_signal	(n = 1)	OUT	Output

4.12.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
PVmax	INTEGER	100	Limit	-

4.12.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
PV	INTEGER	–	Maximum value the counter can count to	
CV	INTEGER	–	Counter	
T	INTEGER	0	Internal variable	

4.12.6 Guidelines

The signal CU is a synchronous entrance so the user must define a clock signal in the schematic diagram (CLK component). The CU signal becomes active at the rising edge of the clock signal.

The numerical values of the limit variable PVmax are implementation dependent.

4.12.7 Formulation

The component is implemented as a continuous equation that defines the value of CV as equal to the output and a set of discrete events defined as follows:

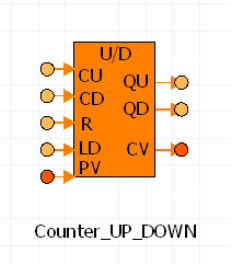
```
WHEN(Clock_signal == TRUE) THEN
   PV = s_in.signal[1]
   IF (CU.signal[1]==TRUE AND CV < PVmax)THEN
      IF (t ==0) THEN
         CV = 0
         t += 1
      ELSE
         CV +=1
      END IF
   END IF
   IF(CV >= PV)THEN
      Q.signal[1] =TRUE
   ELSE
      Q.signal[1] = FALSE
   END IF
END WHEN
WHEN (RESET.signal[1]== TRUE) THEN
   CV = 0
   t = 0
   Q.signal[1] = FALSE
END WHEN
```

4.13 Counter_UP-DOWN

4.13.1 Description

This component is inherited from component SISO. It represents digital up-down-counter as defined in IEC 61131-3.

4.13.2 Symbol



4.13.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
CU	bool_signal	(n = 1)	IN	Up-counter
CD	bool_signal	(n = 1)	IN	Down-counter
RESET	bool_signal	(n = 1)	IN	Reset signal
LD	bool_signal	(n = 1)	IN	Load
QU	bool_signal	(n = 1)	OUT	Output
QD	bool_signal	(n = 1)	OUT	Output

4.13.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
PVmax	INTEGER	100	Upper limit	-
PVmin	INTEGER	0	Lower limit	-

4.13.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
PV	INTEGER	–	Maximum value the counter can count to	
CV	INTEGER	–	Counter	

4.13.6 Guidelines

Signals CU and CD are synchronous so the user must define a clock signal in the schematic diagram (CLK component). The CU signal becomes active at the rising edge of the clock signal.

The numerical values of the limit variables PVmax and PVmin are implementation dependent.

4.13.7 Formulation

The component is implemented as a continuous equation that defines the value of CV as equal to the output and a set of discrete events defined as follows:

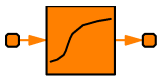
```
WHEN (Clock_signal ==TRUE)THEN
   PV = s_in.signal[1]
   IF (CU.signal[1]== TRUE OR CD.signal[1]==TRUE)THEN
      IF (CU.signal[1]==TRUE AND CV < PVmax) THEN
         CV = CV +1
      ELSEIF (CD.signal[1]==TRUE AND CV >PVmin)THEN
         CV = CV -1
      END IF
   END IF
   IF(CV >= PV)THEN
      QU.signal[1] =TRUE
   ELSE
      QU.signal[1] = FALSE
   END IF
   IF(CV <= 0)THEN
      QD.signal[1] =TRUE
   ELSE
      QD.signal[1] = FALSE
   END IF
END WHEN
WHEN (RESET.signal[1]== TRUE) THEN
   CV= 0
   QU.signal[1] = FALSE
END WHEN
WHEN (LD.signal[1]== TRUE) THEN
   CV = PV
   QD.signal[1] = FALSE
END WHEN
```

4.14 CritDamping

4.14.1 Description

This component is inherited from component SISO. It defines the transfer function between the input signal and the output signal as an n_f-order filter with critical damping characteristics and cut-off frequency freq. It is implemented as a series of first order filters.

4.14.2 Symbol



CritDamping

4.14.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_f	CONST INTEGER	1	Order of filter

4.14.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal ()
s_out	analog_signal	(n = n)	OUT	Outlet signal ()

4.14.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
freq	REAL	1	Cut-off frequency	Hz

4.14.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.14.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
w	REAL			
x[n_f + 1]	REAL		State variables	

4.14.8 Formulation

The angular frequency is calculated with the cut-off frequency (f):

$$\omega = 2 \cdot \pi \cdot f$$

The output signal is given by the following equation sequence where the state variables are computed:

$$x[1] = s\_in.signal[1]$$
$$FOR(i = 1;\quad i \le n_f + 1)$$
$$\frac{\delta x[i]}{\delta t} = \frac{\omega}{2 \cdot \pi} \cdot (x[i-1] - x[i])$$
$$s\_out.signal[1] = x[n_f + 1]$$

where x[i] are the state variables and nf is the order of the filter

4.15 DeadZone

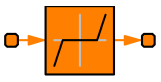
4.15.1 Description

This component is inherited from component MIMOs. It generates zero output signal within a specified region called dead zone. The lower and upper limit of the the dead zone are specified by the user with the data output_max and output_min.

This component generates output signals according to the input and dead zone:

- If the input signal is greater than the lower limit and less than the upper limit, the output signal is zero.
- If the input signal is greater than the upper limit, the output signal is the upper limit minus the input signal.
- If the input signal is less than the lower limit, the output signal is the input signal minus the lower limit
- If the lower and upper limits are equal, the output signal is the value of the input signal minus the dead zone value

4.15.2 Symbol



4.15.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.15.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.15.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
output_max[n]	REAL		Upper limit of the dead zone	
output_min[n]	REAL		Lower limit of the dead zone	

4.15.6 Formulation

The output signal is computed as follows:

$$\begin{aligned} s\_out.signal[i] = IF(s\_in.signal[i] > output\_\max[i]) \quad & s\_in.signal[i] - output\_\max[i] \\ ELSEIF(s\_in.signal[i] < output\_\min[i]) \quad & s\_in.signal[i] - output\_\min[i] \\ ELSE \quad & 0.0 \end{aligned}$$

If the input signal is within the dead zone, i.e. between output_min and output_max, the output signal is zero. Outside this region, the output signal is a linear function of the input signal with a slope of one.

4.16 Delay

4.16.1 Description

This component is inherited from component MIMOs and can be used to simulate a time delay. It delays the input by a specified amount of time specified by the user with the data tdelay.

At the start of the simulation the component outputs the initial value of the input signals until the simulation time exceeds the time specified with tdelay data, then the component outputs the delayed input signals.

4.16.2 Symbol

Delay

4.16.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.16.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.16.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
tdelay[n]	REAL		Delay time	s

4.16.6 Guidelines

The parameter tdelay is a vector whose size is that of the output and input signals. The user can define different delay times for the different output signals.

4.16.7 Formulation

The output signals are computed using the EL function called delay:

$$s\_out.signal[i] \;=\; delay(s\_in.signal[i],\; \max(tdelay[i], 1.0e\text{-}12))$$

The minimum delay time is limited to 1e-12 seconds.

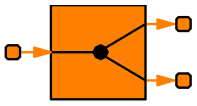
4.17 Demux2

4.17.1 Description

This component is inherited from component Demux. This component separates an input signal vector into two output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of two times the dimension of the output signals.

The input signal is splitted up into two output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.17.2 Symbol



Demux2

4.17.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.17.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.17.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	2	Number of outputs

4.17.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case two.

4.17.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case two. The first output signal is the first set of elements and the second output signal is the second set of elements.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

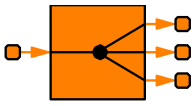
4.18 Demux3

4.18.1 Description

This component is inherited from component Demux. This component separates an input signal vector into three output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of three times the dimension of the output signals.

The input signal is splitted up into three output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.18.2 Symbol



Demux3

4.18.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.18.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.18.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	3	Number of outputs

4.18.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case three.

4.18.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case three. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

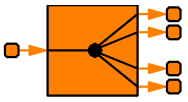
4.19 Demux4

4.19.1 Description

This component is inherited from component Demux. This component separates an input signal vector into four output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of four times the dimension of the output signals.

The input signal is splitted up into four output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.19.2 Symbol



Demux4

4.19.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.19.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.19.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	4	Number of outputs

4.19.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case four.

4.19.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case four. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

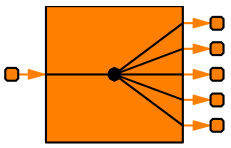
4.20 Demux5

4.20.1 Description

This component is inherited from component Demux. This component separates an input signal vector into five output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of five times the dimension of the output signals.

The input signal is splitted up into five output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.20.2 Symbol



Demux5

4.20.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.20.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.20.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	5	Number of outputs

4.20.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case five.

4.20.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case five. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

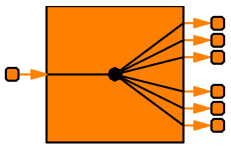
4.21 Demux6

4.21.1 Description

This component is inherited from component Demux. This component separates an input signal vector into six output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of six times the dimension of the output signals.

The input signal is splitted up into six output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.21.2 Symbol



Demux6

4.21.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.21.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.21.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	6	Number of outputs

4.21.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case six.

4.21.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case six. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

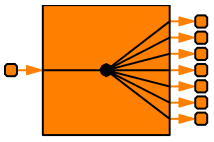
4.22 Demux7

4.22.1 Description

This component is inherited from component Demux. This component separates an input signal vector into seven output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of seven times the dimension of the output signals.

The input signal is splitted up into seven output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.22.2 Symbol



Demux7

4.22.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.22.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.22.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	7	Number of outputs

4.22.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case seven.

4.22.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case seven. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$FOR(j = 1, j \le \dim\_out, j++)$$
$$s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

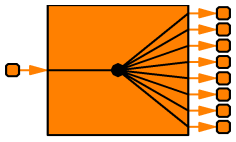
4.23 Demux8

4.23.1 Description

This component is inherited from component Demux. This component separates an input signal vector into eight output signals of the same dimension. The user specifies the dimension of the output signals, then the input signal shall have a dimension of eight times the dimension of the output signals.

The input signal is splitted up into eight output signals. The dimensions of the output signals are explicitly defined by the parameter dim_out.

4.23.2 Symbol



Demux8

4.23.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_out	CONST INTEGER	1	Dimension of outlet signals

4.23.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	n=n_out*dim_out	IN	Inlet port
s_out[n_out]	analog_signal	n = dim_out	OUT	Outlet ports

4.23.5 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	8	Number of outputs

4.23.6 Guidelines

The output signals have the same dimension. The user specifies the dimension of the outlet signal and the dimension of the input signal is determined multiplying the dimension of the output signals by the number of output signals, in this case eight.

4.23.7 Formulation

The elements of the input signal vector is divided in sets of elements of the same dimension depending on the number of the output signals, in this case eight. The first output signal is the first set of elements, the second output signal is the second set of elements and so on.

$$FOR(i = 1, i \le n\_out, i++)$$
$$\quad FOR(j = 1, j \le \dim\_out, j++)$$
$$\quad\quad s\_out[i].signal[j] = s\_in.signal[j + (i-1)\cdot \dim\_out]$$

where:

n_out = number of output signals

dim_out = dimension of the output signals

4.24 Derivative

4.24.1 Description

This component is inherited from component MIMOs. It approximates the derivative of the input signals by calculating:

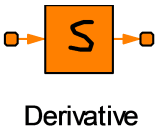
$$\frac{\Delta u}{\Delta t}$$

where Δu is the increment of the input signal in the increment of time Δt.

This transfer function defined in this component between the input signal and the output signal is the following:

$$s\_out.signal[i] = \frac{s}{T_f[i]\cdot s + 1} \cdot s\_in.signal[i]$$

4.24.2 Symbol



4.24.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.24.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.24.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Tf[n]	REAL		Filter time	s

4.24.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
yf[n]	REAL		Auxiliary variable to calculate the approximated derivative	-

4.24.7 Guidelines

The accuracy of the results depends on the size of the filter time Tf. Smaller values of this parameter allow smoother and more accurate output signals.

4.24.8 Formulation

The approximated derivative is calculated with the following expressions:

$$\frac{dy_f}{dt} = \frac{s\_in.signal[i] - y_f}{T_f}$$

$$s\_out.signal[i] = \frac{dy_f}{dt}$$

where:

yf = the auxiliary variable to calculate the approximated derivative.

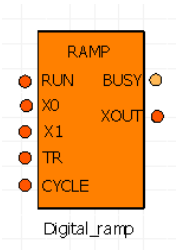
Tf = the time constant

4.25 Digital_ramp

4.25.1 Description

This component represents the time-based ramp function that is described in IEC 61131-3.

4.25.2 Symbol



4.25.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
RUN	analog_signal	(n = 1)	IN	Signal of activation_ 1 = TRUE, 0 = FALSE
X0	analog_signal	(n = 1)	IN	Default value
X1	analog_signal	(n = 1)	IN	Initial value
TR	analog_signal	(n = 1)	IN	Ramp duration
CYCLE	analog_signal	(n = 1)	IN	Sampling time
XOUT	analog_signal	(n = 1)	OUT	Output signal
BUSY	analog_signal	(n = 1)	OUT	Boolean to determine if the ramp is activated

4.25.4 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
XI	DISCR REAL	–	Initial value	–
T	DISCR REAL	–	Time	S
xout	DISCR REAL	–	Output	–
Tstart	DISCR REAL	–	Initial time	s

4.25.5 Formulation

The component is modelled with a continuous equation that defines the value of the output as being equal to the value of the variable xout and a set of discrete events that are the following:

```
WHEN(TIME-Tstart >= CYCLE.signal[1])THEN
   Tstart = TIME
   IF (RUN.signal[1] == 1) THEN
      BUSY.signal[1] = TRUE
      IF (T >= TR.signal[1]) THEN
         xout = X1.signal[1]
      ELSE
         xout = XI + (X1.signal[1] - XI) * T/ TR.signal[1]
         T = T + CYCLE.signal[1]
      END IF
   ELSEIF (RUN.signal[1]==0) THEN
      BUSY.signal[1] = FALSE
      xout = X0.signal[1]
      XI = X0.signal[1]
      T = 0
```

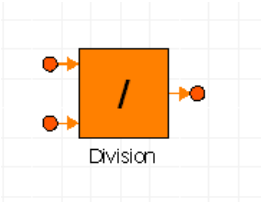
```
   ELSE
      ASSERT (FALSE) FATAL "RUN can only take values 0 and 1"
   END IF
END WHEN
```

4.26 iDivision

4.26.1 Description

This component is inherited from component MI2MOs. It calculates the integer part of the division of two input signals fractional quotients are rounded toward zero to the nearest integers.

4.26.2 Symbol



4.26.3 Formulation

The equation associated with this component is the following:

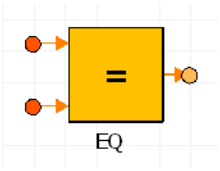
$$s\_out.signal = \text{int}(\frac{s\_in1.signal}{s\_in2.signal})$$

4.27 EQ

4.27.1 Description

This component is inherited from component SI2bSO. It represents the standard comparison function 'equal to'.

4.27.2 Symbol



4.27.3 Formulation

The equation associated with this component is the following:

```
WHEN (abs(s_in_1.signal[1] - s_in_2.signal[1]) > 0) THEN
   s_out.signal[1] = FALSE
END WHEN
WHEN (NOT abs(s_in_1.signal[1] - s_in_2.signal[1]) > 0) THEN
   s_out.signal[1] = TRUE
END WHEN
```

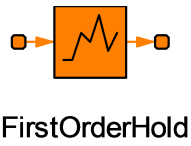
4.28 FirstOrderHold

4.28.1 Description

This component is inherited from component dMIMOs. The output signals of this component is the extrapolation through the values of the last two sampled input signals.

This component implements a first-order sample and hold that works at the specified sample time dt defined by the user.

4.28.2 Symbol



4.28.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.28.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.28.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s

4.28.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
To	REAL		Current sample time	s
sample	BOOLEAN	FALSE		
signal_1[n]	REAL		Array with values of the last sampled input signal	-
signal_2[n]	REAL		Array with values of the penultimate sampled input signal	-

4.28.7 Guidelines

The sample time dt is the time interval between samples. This data is a scalar and it is used for all the signals.

4.28.8 Formulation

The input signals are sampled in the DISCRETE block of the component with WHEN blocks. The input signals are sampled every time sampling interval dt:

$$
\begin{aligned}
&WHEN(sample == TRUE) \qquad THEN\\
&\qquad sample = FALSE\\
&\qquad sample = TRUE \ \ after \ \ dt
\end{aligned}
$$

where sample is a Boolean variable that indicates to the component when it must sample

$$
\begin{aligned}
&WHEN(sample == TRUE) \qquad THEN\\
&\qquad To = TIME\\
&\qquad FOR(i = 0;\ \ i \le n;\ \ i++)\\
&\qquad\qquad signal\_2[i] = signal\_1[i]\\
&\qquad\qquad signal\_1[i] = s\_in.signal[i]
\end{aligned}
$$

The output signals are computed as follows:

$$s\_out.signal[i] = signal\_1[i] + (signal\_1[i] - signal\_2[i]) \cdot \frac{TIME - To}{dt}$$

where:

signal_1[i] = the value of the last sampled input signal

signal_2[i] = the value of the input signal that was sampled two sample times ago

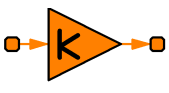
To = the current sample time

dt = the sample time

4.29 Gain

This component is inherited from component MIMOs. It generates the output signals by multiplying the input signals by a contant specified by the user.

4.29.1 Symbol



Gain

4.29.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.29.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.29.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
k[n]	REAL		Gain	-

4.29.5 Guidelines

The gain data k is a vector of the same size as the size of the input and output signals. Each element of the input signal vector is multiplied by the corresponding element of the gain vector to generate the elements of the output signal vector.

4.29.6 Formulation

The output signals are calculating multiplying the input signals by the corresponding contant of the gain vector, as follows:

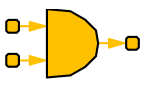
$$s\_out.signal[i] = k[i] \cdot s\_in.signal[i]$$

where k[i] is the gain

4.30 Gate_AND

This component is inherited from component bSI2SO. It defines a logical gate AND. The output signal is TRUE if both input signals are true, otherwise the output signal is FALSE.

4.30.1 Symbol



Gate_AND

4.30.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	bool_signal	(n = n)	IN	Inlet signal
s_in_2	bool_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

4.30.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.30.4 Formulation

The output signal is TRUE only when the two input signals are TRUE.

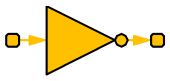
$$\text{IF } (s\_in\_1.signal[j] = TRUE \ \& \ s\_in\_2.signal[j] = TRUE) \Rightarrow \quad s\_out.signal[j] = TRUE$$
$$\text{ELSE} \Rightarrow \quad s\_out.signal[j] = FALSE$$

4.31 Gate_NOT

This component is inherited from component bSISO. It defines a logical gate NOT. This component inverts the value of the input signal.

This component performs the NOT logical operation.

4.31.1 Symbol



Gate_NOT

4.31.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	bool_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

4.31.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.31.4 Formulation

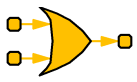
The output signal will be TRUE if the input signal is FALSE. And the output signal will be FALSE if the input signal is TRUE.

$$\text{IF } (s\_in.signal[j] = TRUE) \Rightarrow s\_out.signal[j] = FALSE$$
$$\text{IF } (s\_in.signal[j] = FALSE) \Rightarrow s\_out.signal[j] = TRUE$$

4.32 Gate_OR

This component is inherited from component bSI2SO. It represent a logical operator OR. The output signal is only FALSE if both input signals are FALSE, otherwise the output signal is TRUE.

4.32.1 Symbol



Gate_OR

4.32.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	bool_signal	(n = n)	IN	Inlet signal
s_in_2	bool_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

4.32.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.32.4 Formulation

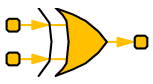
The output signal is FALSE only when the two input signals are FALSE otherwise the output signal is TRUE.

$$\text{IF}(s\_in\_1.signal[j] = TRUE \quad \text{OR} \quad s\_in\_2.signal[j] = TRUE) \Rightarrow s\_out.signal[j] = TRUE$$
$$\text{IF}(s\_in\_1.signal[j] = FALSE \quad \& \quad s\_in\_2.signal[j] = FALSE) \Rightarrow s\_out.signal[j] = FALSE$$

4.33 Gate_XOR

This component is inherited from component bSI2SO. It represent a logical operator XOR. The otuput signal is FALSE if the input signals are both FALSE or both TRUE, otherwise the output signal is TRUE.

4.33.1 Symbol



Gate_XOR

4.33.2 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	bool_signal	(n = n)	IN	Inlet signal
s_in_2	bool_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

4.33.3 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.33.4 Formulation

The output signal is only FALSE if the input signals have the same value, both are FALSE or both are TRUE, otherwise the output signal is TRUE.

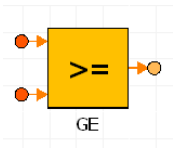
$$\text{IF}(s\_in\_1.signal[j] = TRUE \;\; \& \;\; s\_in\_2.signal[j] = TRUE) \Rightarrow s\_out.signal[j] = FALSE$$
$$\text{ELSEIF}\,(s\_in\_1.signal[j] = FALSE \;\; \& \;\; s\_in\_2.signal[j] = FALSE) \Rightarrow s\_out.signal[j] = FALSE$$
$$ELSE \Rightarrow TRUE$$

4.34 GE

4.34.1 Description

This component is inherited from component SI2bSO. It represents the standard comparison function 'greater than or equal to'.

4.34.2 Symbol



4.34.3 Formulation

The equation associated with this component is the following:

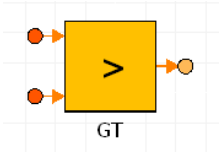
```
WHEN (s_in_1.signal[1] < s_in_2.signal[1]) THEN
   s_out.signal[1] = FALSE
END WHEN
WHEN (NOT s_in_1.signal[1] < s_in_2.signal[1]) THEN
   s_out.signal[1] = TRUE
END WHEN
```

4.35 GT

4.35.1 Description

This component is inherited from component SI2bSO. It represents the standard comparison function 'greater than'.

4.35.2 Symbol



GT

4.35.3 Formulation

The equation associated with this component is the following:

```
WHEN (s_in_1.signal[1] > s_in_2.signal[1]) THEN
   s_out.signal[1] = TRUE
END WHEN
WHEN (NOT s_in_1.signal[1] > s_in_2.signal[1]) THEN
   s_out.signal[1] = FALSE
END WHEN
```

4.36 Hysteresis

4.36.1 Description

This component is inherited from component SI2bSO. It implements Boolean hysteresis on the difference of REAL inputs.

4.36.2 Symbol

Hysteresis

4.36.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
eps	analog_signal	(n = 1)	IN	Epsilon value defined by the user. The values must be possitive.

4.36.4 Formulation

The component is implemented as a set of discrete events defined as follows:

```
WHEN (s_in_1.signal[1]< s_in_2.signal[1] - eps.signal[1] ) THEN
      s_out.signal[1] = FALSE
END WHEN
WHEN (s_in_1.signal[1]> s_in_2.signal[1] + eps.signal[1] ) THEN
   s_out.signal[1] = TRUE
END WHEN
```

4.37 Integrator

This component is inherited from component MIMOs. It represents an integrator unit typically employed in control systems.

The component calculates an output signal that is simply the integral of the input signal.

4.37.1 Symbol

Integrator

4.37.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.37.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.37.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
output_o[n]	REAL		Initial values of output signals	s

4.37.5 Guidelines

This component allows to initialize the output signals by means of the data vector named output_o. The n-th element of the input signal vector is integrated with the initial value specified by the n-th element of the initial value vector named output_o to produce the n-th element of the output signal vector.

4.37.6 Formulation

The derivative of the output signal responds instantaneously to changes of the input signal.

$$\frac{\partial s_{out}}{\partial t} = s_{in}$$

where

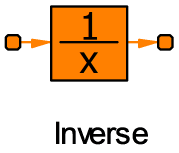
sout = the value of the output signal

sin = the value of the input signal.

4.38 Inverse

This component is inherited from component MIMOs. It calculates the inverse of the input analog signals.

4.38.1 Symbol



Inverse

4.38.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.38.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.38.4 Formulation

The output signals are calculated as the inverse of the input signals:

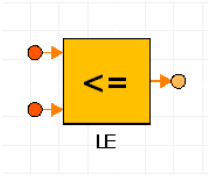
$$s\_out.signal[i] = \frac{1}{s\_in.signal[i]}$$

4.39 LE

4.39.1 Description

This component is inherited from component SI2bSO. It represents the standard comparison function 'lower than or equal to'.

4.39.2 Symbol



4.39.3 Formulation

The equation associated with this component is the following:

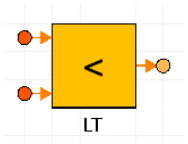
```
WHEN (s_in_1.signal[1] > s_in_2.signal[1]) THEN
   s_out.signal[1] = FALSE
END WHEN
WHEN (NOT s_in_1.signal[1] > s_in_2.signal[1]) THEN
   s_out.signal[1] = TRUE
END WHEN
```

4.40 LT

4.40.1 Description

This component is inherited from component SI2bSO. It represents the standard comparison function 'lower than'.

4.40.2 Symbol



4.40.3 Formulation

The equation associated with this component is the following:

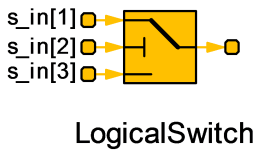
```
WHEN (s_in_1.signal[1] < s_in_2.signal[1]) THEN
   s_out.signal[1] = TRUE
END WHEN
WHEN (NOT s_in_1.signal[1] < s_in_2.signal[1]) THEN
   s_out.signal[1] = FALSE
END WHEN
```

4.41 LogicalSwitch

This component switches depending on the value of the input signal of port s_in[2] between two possible input signals correponding to ports s_in[1] and s_in[3]. If the signal of port s_in[2] is TRUE, the output signal

will be equal to the input signal of port s_in[1] otherwise the output signal will be equal to the input signal of port s_in[3].

4.41.1 Symbol



4.41.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.41.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[3]	bool_signal	(n = n)	IN	Boolean inlet signal
s_out	bool_signal	(n = n)	OUT	Boolean outlet signal

4.41.4 Formulation

The output signals are computed as follows:

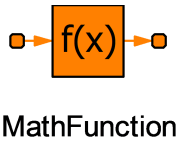
$$IF\ (s\_in[2].signal[i] = TRUE\ ) \quad \Rightarrow\ s\_out.signal[i] = s\_in[1].signal[i]$$
$$ELSE \quad \Rightarrow s\_out.signal[i] = s\_in[3].signal[i]$$

4.42 MathFunction

This component is inherited from component MIMOs. It performs basic mathematical functions to the input signals. The user can specify the mathematical operation with the parameter option. The mathematical functions available in this component are the following:

option	Description
FunAbs	To calculate the absolute value of the input signals
FunSign	To calculate the sign of the input signals
FunSqrt	To calculate the square root of the input signals
FunSin	To calculate the sine of the input signals
FunCos	To calculate the cosine of the input signals
FunTan	To calculate the tangent of the input signals
FunAsin	To calculate the arc-sine of the input signals
FunAcos	To calculate the arc-cosine of the input signals
FunAtan	To calculate the arc-tangent of the input signals
FunSinh	To calculate the hyperbolic sine of the input signals
FunCosh	To calculate the hyperbolic cosine of the input signals
FunTanh	To calculate the hyperbolic tangent of the input signals
FunExp	To calculate the exponential of the input signals
FunLog10	To calculate the base 10 logarithm of the input signals
FunLog	To calculate the natural logartihm of the input signals

4.42.1 Symbol



4.42.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs
option	CONST ENUM MathOption	FunSin	Mathematical function

4.42.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.42.4 Guidelines

The users specificy the mathematical functions they want to apply to the input signals with the parameter option.

The exceptions of the mathematical functions of this component are not protected. The exceptions of the mathematical functions will be detected by the compiler.

4.42.5 Formulation

The mathematical function performed to the input signals depends on the option choosen by the user:

option	Equation applied
FunAbs	abs(s_in.signal[i])
FunSign	sign(s_in.signal[i])
FunSqrt	sqrt(s_in.signal[i])
FunSin	sin(s_in.signal[i])
FunCos	cos(s_in.signal[i])
FunTan	tan(s_in.signal[i])
FunAsin	asin(s_in.signal[i])
FunAcos	acos(s_in.signal[i])
FunAtan	atan(s_in.signal[i])
FunSinh	sinh(s_in.signal[i])
FunCosh	cosh(s_in.signal[i])
FunTanh	tanh(s_in.signal[i])
FunExp	exp(s_in.signal[i])
FunLog10	log10(s_in.signal[i])
FunLog	log(s_in.signal[i])

All these functions (abs, sign, sqrt, sin, cos...) are EL language global functions.

4.43 Maximum

This component is inherited from component MI2MOs. It calculates which is the maximum value between the two input signals.

4.43.1 Symbol

Maximum

4.43.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.43.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	First inlet signal
s_in_2	analog_signal	(n = n)	IN	Second inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.43.4 Formulation

If the first input signal is greater than the second input signal then the value of the output signal will be the value of the first input signal, otherwise the value of the output signal will be the value of the second input signal.

$$IF(s\_in.1.signal[i] > s\_in.2.signal[i]) \quad \Rightarrow s\_out.signal[i] = s\_in.1.signal[i]$$
$$ELSE \quad \Rightarrow s\_out.signal[i] = s\_in.2.signal[i]$$

4.44 Minimum

This component is inherited from component MI2MOs. It calculates which is the minimum value between the two input signals.

4.44.1 Symbol

Minimum

4.44.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.44.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	First inlet signal
s_in_2	analog_signal	(n = n)	IN	Second inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.44.4 Formulation

If the first input signal is lower than the second input signal then the value of the output signal will be the value of the first input signal, otherwise the value of the output signal will be the value of the second input signal.

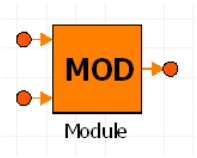
$$IF(s\_in.1.signal[i] < s\_in.2.signal[i]) \quad \Rightarrow s\_out.signal[i] = s\_in.1.signal[i]$$
$$ELSE \quad \Rightarrow s\_out.signal[i] = s\_in.2.signal[i]$$

4.45 Module

4.45.1 Description

This component is inherited from component MI2MOs. It represents an output signal that is the module of a division of two input signals.

4.45.2 Symbol



4.45.3 Formulation

The equation associated with this component is as follows. If s_in_2.signal[i] is not 0 :

```
s_out.signal[i] = s_in_1.signal[i] - int(s_in_1.signal[i]/s_in_2.signal[i]) * s_in_2.
  signal[i]
```

If s_in_2.signal[i] is 0 the output is 0.

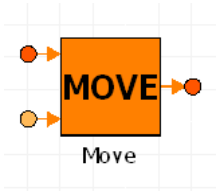
Int() is the function implemented in MATH library that returns the integer of the input value.

4.46 Move

4.46.1 Description

This component is inherited from component SISO. It represents the standard arithmetic operation MOVE (IEC1131-3). It puts the same signal that is connected to the input at the output.

4.46.2 Symbol



4.46.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
EN	bool_signal	(n = 1)	IN	Enable signal

4.46.4 Formulation

The output signals respond instantaneously to changes of the input signals if the enable signal is true. The equation associated with this component is the following:

```
s_out.signal[1] = s_in.signal[1]
```

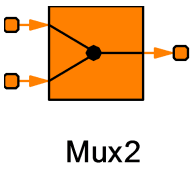
4.47 Mux2

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port

The user specifies the dimension of the input signal ports, then the output signal port shall have a dimension of two times the dimension of the input signal ports.

The two input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.47.1 Symbol



4.47.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.47.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.47.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	2	Number of outputs

4.47.5 Guidelines

All the input signal ports have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case two.

4.47.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

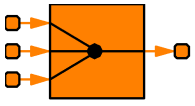
4.48 Mux3

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port

The user specifies the dimension of the input signal port, then the output signal port shall have a dimension of three times the dimension of the input signal ports.

The three input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.48.1 Symbol



Mux3

4.48.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.48.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.48.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	3	Number of outputs

4.48.5 Guidelines

The input signal ports have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case three.

4.48.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$\quad FOR(j=1, j \le \dim\_in, j++)$$
$$\quad\quad s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

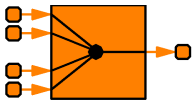
4.49 Mux4

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port

The user specifies the dimension of the input signal port, then the output signal port shall have a dimension of four times the dimension of the input signal ports.

The four input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.49.1 Symbol



Mux4

4.49.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.49.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.49.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	4	Number of outputs

4.49.5 Guidelines

The input signals have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case four.

4.49.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

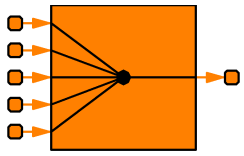
4.50 Mux5

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port

The user specifies the dimension of the input signal port, then the output signal port shall have a dimension of five times the dimension of the input signal ports.

The five input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.50.1 Symbol



Mux5

4.50.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.50.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.50.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	5	Number of outputs

4.50.5 Guidelines

The input signals have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case five.

4.50.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

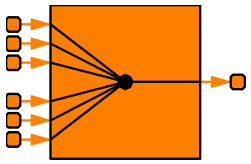
4.51 Mux6

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port

The user specifies the dimension of the input signal port, then the output signal port shall have a dimension of six times the dimension of the input signal ports.

The six input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.51.1 Symbol



Mux6

4.51.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.51.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.51.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	6	Number of outputs

4.51.5 Guidelines

The input signals have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case six.

4.51.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

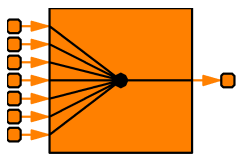
4.52 Mux7

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port.

The user specifies the dimension of the input signal port, then the output signal port shall have a dimension of seven times the dimension of the input signal ports.

The seven input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.52.1 Symbol



Mux7

4.52.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.52.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.52.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	7	Number of outputs

4.52.5 Guidelines

The input signals have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case seven.

4.52.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

dim_in = dimension of the input signal ports

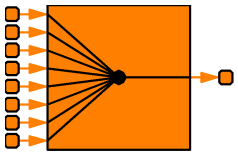
4.53 Mux8

This component is inherited from component Mux. This component combines several input signal ports of the same dimension into only one output signal port

The user specifies the dimension of the input signal port, then the output signal port shall have a dimension of eight times the dimension of the input signal ports.

The eight input signal ports are combined into an only output signal port. The dimensions of the input signal ports are explicitly defined by the parameter dim_in.

4.53.1 Symbol



Mux8

4.53.2 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
dim_in	CONST INTEGER	1	Dimension of input signals

4.53.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in[n_in]	analog_signal	n = dim_in	IN	Inlet signal ports
s_out	analog_signal	n=n_in·dim_in	OUT	Outlet signal port

4.53.4 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	8	Number of outputs

4.53.5 Guidelines

The input signals have the same dimension. The user specifies the dimension of the input signal ports and the dimension of the output signal port is determined multiplying the dimension of the input signals port by the number of input signal port, in this case eight.

4.53.6 Formulation

The output signal vector is built concatenating the elements of the input ports. The first dim_in elements of the output signal port will be the set of elements of the first input signal port, the second dim_in elements of the output signal port will be the set of elements of the second input signal port and so on.

$$FOR(i=1, i \le n\_in, i++)$$
$$FOR(j=1, j \le \dim\_in, j++)$$
$$s\_out.signal[(i-1)\cdot \dim\_in + j] = s\_in[i].signal[j]$$

where:

n_in = number of input signal ports

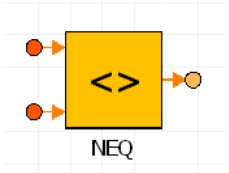
dim_in = dimension of the input signal ports

4.54 NEQ

4.54.1 Description

This component is inherited from component SI2bSO. It represents the standard comparison function 'not equal to'.

4.54.2 Symbol



4.54.3 Formulation

The equation associated with this component is the following:

```
WHEN (abs(s_in_1.signal[1] - s_in_2.signal[1]) > 0) THEN
   s_out.signal[1] = TRUE
END WHEN
WHEN (NOT abs(s_in_1.signal[1] - s_in_2.signal[1]) > 0) THEN
   s_out.signal[1] = FALSE
END WHEN
```

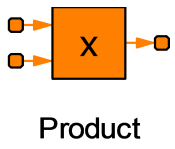
4.55 Product

4.55.1 Description

This component is inherited from component MI2MOs. It represents a multiplier unit tipically employed in control systems.

The component calculates an output signal that is the product of the two input signals.

4.55.2 Symbol



4.55.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.55.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	Inlet signal
s_in_2	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.55.5 Guidelines

4.55.6 Formulation

The output signals respond instantaneously to changes of the input signals. It calculates the output signal elements as the product of the corresponding elements of the two input signal ports.

$$s\_out.signal[i] = s\_in\_1.signal[i] \cdot s\_in\_2.signal[i]$$

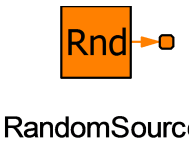
4.56 RandomSource

4.56.1 Description

This component is inherited from component SO. It generates random numbers following various probability distributions depending on the value of the parameter random.

It generates normally distributed random numbers. The user specifies the starting seed value to generate the random numbers with the parameter iseed.

4.56.2 Symbol



RandomSource

4.56.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
iseed	CONST INTEGER	1200	Integer seed
random	CONST ENUM RandomOption	Uniform	Probability distribution

4.56.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.56.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
a	REAL	0	Lower limit of interval on uniform distribution	-
b	REAL	1	Higher limit of interval on uniform distribution	-
dt	REAL	0.1	Time between samples	s
lambda	REAL	1	Mean of exponential distribution	-
mu	REAL	0	Mean of gaussian distribution	-
n	INTEGER	10	Number of tries on binomial distribution	-
order	INTEGER	3	Order of gamma distribution	-
p	REAL	0.5	Success probability on binomial distribution	-
sigma	REAL	1	Typical deviation of gaussian distribution	-
xm	REAL	1	Mean of Poisson distribution	-

4.56.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of the outlet signal port

4.56.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
Sample	BOOLEAN	TRUE	Sampler	TRUE/FALSE
i_array[15]	INTEGER		Auxiliar array of integers	-
r_array[150]	REAL		Auxiliar array	-
x	REAL		Random numbers	-

4.56.8 Guidelines

The users can choose the probability distribution type that they want to use to generate the random numbers with the parameter random. The types of probability distribution available in this component are shown in the following table:

random option	Description
Uniform	Uniform distribution
Binomial	Binomial distribution
Exponential	Exponential distribution
Gamma	Gamma distribution
Gaussian	Gaussian distribution
Poisson	Poisson distribution

Defines a component generating random numbers following various probability distributions, depending on the value of variable random.

random option	Equation
Uniform	if(a < x <b) p(x) = 1 / (b - a) when (a < x < b) else p(x) = 0
Binomial	Pnp(j) = (n over j) * p**j * (1 - p)**(n - j)
Exponential	p(x) = exp(-lambda * x)
Gamma	p(x) = x**(a - 1) * exp(-x) / gamma(a)
Gaussian	p(x) = 1 / sqrt(2 * PI * sigma) * exp(- ((x - mu) / sigma)**2)
Poisson	Px(j) = x**j * exp(-x) / j!

Notes:

Random functions need two arrays (one of integer parameters and other ofreal parameters). This arrays are over dimensioned, and their function is:

i_array[15]: Components 1 to 5 are needed by uniform random number generator function and represent the modules and two flags which tell wether the function is called the first time or it has been already called. Components 6 to 10 represent integer parameters of the distribution, such as, for example, the number of tries "n" on Binomial distribution. Components 11 to 15 are flags. This flags may save values obtained from integer parameters of the last call, such as old value of "n" on binomial distribution, or simple flags.

i_array[11]: On gaussian case, which indicates wheter is neccessary to calculate or not.

r_array[150]: Components 1 to 98 are needed by uniform random number generator function to save the history of the generation. Components 99 to 100 are not used. Components 101 to 120 represent real parameters, such as the probability of success "p" on binomial distribution. Components 121 to 150 are flags saving some values obtained from parameters, such as the old value of "p" (r_array[121]) on binomial distribution.

4.56.9 Formulation

This component generates random numbers following various probability distrributions, depending on the value of variable "random".

random option	Equation
Uniform	if(a < x <b) p(x) = 1 / (b - a) when (a < x < b) else p(x) = 0
Binomial	Pnp(j) = (n over j) * p**j * (1 - p)**(n - j)
Exponential	p(x) = exp(-lambda * x)
Gamma	p(x) = x**(a - 1) * exp(-x) / gamma(a)
Gaussian	p(x) = 1 / sqrt(2 * PI * sigma) * exp(- ((x - mu) / sigma)**2)
Poisson	Px(j) = x**j * exp(-x) / j!

Notes:

Random functions need two arrays (one of integer parameters and other ofreal parameters). This arrays are over dimensioned, and their function is:

i_array[15]: Components 1 to 5 are needed by uniform random number generator function and represent the modules and two flags which tell wether the function is called the first time or it has been already called. Components 6 to 10 represent integer parameters of the distribution, such as, for example, the number of tries "n" on Binomial distribution. Components 11 to 15 are flags. This flags may save values obtained from integer parameters of the last call, such as old value of "n" on binomial distribution, or simple flags.

i_array[11]: On gaussian case, which indicates wheter is neccessary to calculate or not.

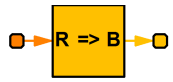
r_array[150]: Components 1 to 98 are needed by uniform random number generator function to save the history of the generation. Components 99 to 100 are not used. Components 101 to 120 represent real parameters, such as the probability of success "p" on binomial distribution. Components 121 to 150 are flags saving some values obtained from parameters, such as the old value of "p" (r_array[121]) on binomial distribution.

4.57 RealToBoolean

4.57.1 Description

This component transforms analog signals to Boolena signals. The user defines the upper and lower limit to switch the Boolean ouput to TRUE and FALSE.

4.57.2 Symbol



RealToBoolean

4.57.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs signal vectors

4.57.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Analog inlet signal
s_out	bool_signal	(n = n)	OUT	Boolean outlet signal

4.57.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
ValueFalse	REAL	0.5	Upper limit for switching the Boolean output to FALSE	-
ValueTrue	REAL	0.5	Lower limit for switching the Boolean output to TRUE	-

4.57.6 Formulation

When the analog signal is greater than the data ValueTrue specified by the user the output Boolean signal will be TRUE and when the analog signal is lower than the data ValueFalse the output Boolean signal will be FALSE:

$$IF\,(s\_in.signal[i] > ValueTrue) \quad \Rightarrow \quad s\_out.signal[i] = TRUE$$
$$IF\,(s\_in.signal[i] < ValueFalse) \quad \Rightarrow \quad s\_out.signal[i] = FALSE$$

4.58 RelationalOperator

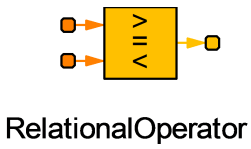
4.58.1 Description

This component is inherited from component SI2bSO. It performs a relational operation on its two input signal ports and generates Boolean output signals according to the relational operation specified by the user with the parameter option. The following table shows the available relational operations and a description of the output signal generated:

Relational Operator option	Output signal Value
Equal	TRUE if the first input signal is equal to the second input signal
NotEqual	TRUE if the first input signal is not equal to the second input signal
GreaterThan	TRUE if the first input signal is greater than the second input signal
GreaterEqual	TRUE if the first input signal is greater than or equal to the second input signal
LessTahn	TRUE if the first input signal is less than the second input signal
LessEqual	TRUE if the first input signal is less than or equal to the second input signal

The first input signal port in the symbol of the component is the upper port and the second input signal port in the simbol of the component is the lower port.

4.58.2 Symbol



4.58.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs
option	CONST ENUM RelationalOption	Equal	Relational Operation

4.58.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	Inlet signal
s_in_2	analog_signal	(n = n)	IN	Inlet signal
s_out	bool_signal	(n = n)	OUT	Outlet signal

4.58.5 Formulation

Depending on the value of the Relational Operator option the mathematical expression applied to the component will be the following:

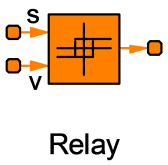
Relational Operator option	Output signal Value
Equal	IF(s_in_1.signal[i] == s_in_2.signal[i]) s_out.signal[i] = TRUE ELSE s_out.signal[i] = FALSE
NotEqual	IF(s_in_1.signal[i] != s_in_2.signal[i]) s_out.signal[i] = TRUE ELSE s_out.signal[i] = FALSE
GreaterThan	IF(s_in_1.signal[i] $>$ s_in_2.signal[i]) s_out.signal[i] = TRUE ELSE s_out.signal[i] = FALSE
GreaterEqual	IF(s_in_1.signal[i] $\geq$ s_in_2.signal[i]) s_out.signal[i] = TRUE ELSE s_out.signal[i] = FALSE
LessThan	IF(s_in_1.signal[i] $<$ s_in_2.signal[i]) s_out.signal[i] = TRUE ELSE s_out.signal[i] = FALSE
LessEqual	IF(s_in_1.signal[i] $\leq$ s_in_2.signal[i]) s_out.signal[i] = TRUE ELSE s_out.signal[i] = FALSE

4.59 Relay

4.59.1 Description

This component is inherited from component Controller. It allows the output signal to switch between two specified values. When the relay is ON state, it remains in this state until the difference between the setpoint signal and controlled variable signal drops below the specified value of the e_off data. And when the relay is in OFF state, it remains in this state until the difference between the setpoint signal and the controlled variable signal is greater than the specified value of the e_on data.

4.59.2 Symbol



4.59.3 Parameters

y	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.59.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n)	OUT	Controlled output signal
s_set	analog_signal	(n = n)	IN	Set point signal
s_var	analog_signal	(n = n)	IN	Controlled variable signal

4.59.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
e_off[n]	REAL		Error for switching to OFF state	-
e_on[n]	REAL		Error for switching to ON state	-
u_off[n]	REAL		Value of controller output when OFF	-
u_on[n]	REAL		Value of controller output when ON	-

4.59.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
e[n]	REAL		Input error	-
r[n]	REAL		Set point	-
state[n]	ENUM state_type		State of controller	ON/OFF
u[n]	REAL		Output	-
y[n]	REAL		Measured variable	-

4.59.7 Guidelines

If the users specify the e_on value greater than the e_off value, they are going to model hysteresis, whereas if they specify equal values for these data, they are going to model a switch with a threshold at that value.

4.59.8 Formulation

When the relay is ON the output signal is equal to the u_on data value, and when the relay is OFF the output signal is equal to the u_off data value.

The component obtains the exact time at which the state changes for each element of the array of the controlled process variables by means of two WHEN statements:

$$\text{WHEN } (e[i] > e\_on[i]) \Rightarrow state[i] = ON$$

$$\text{WHEN } (e[i] < e\_off[i]) \Rightarrow state[i] = OFF$$

The two input signals to the Relay component are the set-point and the controlled process variable:

r[i] = s_set.signal[i]

y[i] = s_var.signal[i]

The error signal is calculated as follows:

e[i] = r[i] - y[i]

The output is a function of the variable state:

$$u[i] = \begin{cases} u\_on[i] \text{ for } state[i] = ON \\ u\_off[i] \text{ for } state[i] = OFF \end{cases}$$

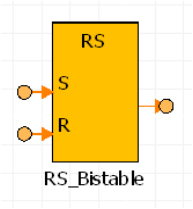
s_out.signal[i] = u[i]

4.60 RS_Bistable

4.60.1 Description

This component is inherited from component bMI2MOs. It represents a reset dominant bistable as defined in IEC 61131-3.

4.60.2 Symbol



4.60.3 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
init_val	BOOLEAN	FALSE	Initial output value	-

4.60.4 Formulation

```
WHEN((s_in_1.signal[1] == TRUE OR s_out.signal[1] == TRUE) AND s_in_2.signal[1] == FALSE) THEN
   s_out.signal[1] = TRUE
END WHEN
WHEN(s_in_1.signal[1] == FALSE AND s_out.signal[1] == FALSE AND s_in_2.signal[1] == FALSE)
  THEN
   s_out.signal[1] = FALSE
END WHEN
WHEN (s_in_2.signal[1] == TRUE) THEN
   s_out.signal[1] = FALSE
END WHEN
```

4.61 Saturation

4.61.1 Description

This component is inherited from component MIMOs. It imposes upper and lower bounds or limits on the output signal.

This component passes the input signal as output signal as long as the input signal is within the specified upper and lower bounds. If not, the defined limits are passes as the output signal.

4.61.2 Symbol

Saturation

4.61.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.61.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.61.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
output_max[n]	REAL		Upper limit of input signal	
output_min[n]	REAL		Lower limit of input signal	

4.61.6 Guidelines

If the users set the upper and lower limit to the same value, the output signals are going to be always equal to this value.

4.61.7 Formulation

The output signals are computed as follows:

$$s\_out.signal[i] = \begin{cases} IF(s\_in.signal[i] > output\_max[i]) & output\_max[i] \\ IF(s\_in.signal[i] < output\_min[i]) & output\_min[i] \\ ELSE & s\_in.signal[i] \end{cases}$$

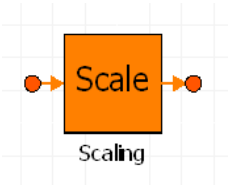
The component passes the input signals as the output signals when the input signals are within the upper and lower limits, if not the output signals will be equal to the corresponding limit.

4.62 Scaling

4.62.1 Description

This component represents a scaling of the input signal to a range defined by the user.

4.62.2 Symbol



4.62.3 Data

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
in_min	REAL	0	Minimum of input range	
in_max	REAL	100	Maximum of input range	
out_min	REAL	0	Minimum of output range	
out_max	REAL	100	Maximum of output range	

4.62.4 Formulation

The equation that defines the output of the component is the following:

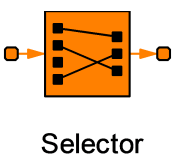
$$s\_out.signal = \frac{(s\_in.signal - in_{\min}) \cdot (out_{\max} - out_{\min})}{in_{\max} - in_{\min}} + out_{\min}$$

4.63 Selector

4.63.1 Description

This component is inherited from component MIMO. It extracts signals from the input port and transfers them to the output port.

4.63.2 Symbol



4.63.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of the outputs

4.63.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n_in)	IN	Inlet signal
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.63.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
SelElements[n_out]	INTEGER		Elements to be included in the output vector	

4.63.6 Guidelines

The user specifies with the vector called SelElements which input signals are taken and in which order they are transferred to the output signal vector. The dimension of the vector SelElements has to match with the dimension of the output signal port.

The component checks that the value of the elements defined in the SelElements vector has to be lower than or equal to the dimension of the input signal port and greater than zero.

4.63.7 Formulation

The input signals transferred to the output signal vector are specified by the integer vector SelElements:

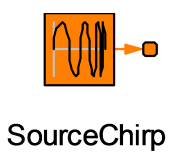
$$s\_out.signal[i] = s\_in.signal[SelElements[i]]$$

4.64 SourceChirp

4.64.1 Description

This component is inherited from component MO. It generates a sine wave whose frequency increases at a linear rate with time. The user specifies the parameters for the sine wave and the initial signal frequency and the frequency linear rate with time.

4.64.2 Symbol



4.64.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.64.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.64.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Amp	REAL	1	Signal amplitude	-
Offset	REAL	0	Offset of output signal	-
Phase	REAL	0	Signal phase	rad
Tstart	REAL	0	Starting time of signal generation	s
frec_o	REAL	0.1	Initial signal frequency	Hz
frec_slope	REAL	0.1	Frequency linear rate with time	1/s^2

4.64.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
frec	REAL		Signal frecuency	Hz

4.64.7 Guidelines

The user can use this component for spectral analysis of nonlinear systems.

4.64.8 Formulation

The output signals of this component are generated as follows:

$$s\_out.signal[i] = \begin{array}{ll} IF(TIME < Tstart) & Offset \\ ELSE & Offset + A \cdot \sin\left(2 \cdot \pi \cdot f \cdot \left(TIME - Tstart\right) + \phi\right) \end{array}$$

where:

Offset = the offset of the output signal

A = the signal amplitude

Tstart = the starting time of the signal generation

ϕ = the signal phase

f = the signal frequency. This frequency is linearly variable with time:

$$\frac{\partial f}{\partial t} = \begin{array}{ll} IF(TIME < Tstart) & 0.0 \\ ELSE & f_{slope} \end{array}$$

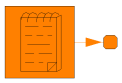
fslope = the frequency linear rate with time

4.65 SourceDataFile

4.65.1 Description

This component is inherited from component SO. It generates an analog signal interpolating in a table that is read from a ASCII file specified by the user.

4.65.2 Symbol



SourceDataFile

4.65.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = 1)	OUT	Outlet signal

4.65.4 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
tabmethod	ENUM	LinearInterpWith-Events	Method to interpolate or connect the table points	-
nt	INTE-GER	1	Number of the column of time data in the file	-
ny	INTE-GER	2	Number of the column of the dependent variable in the file	-
filename	STRING		Pathname of the ASCII file, for example: C:\\DataFiles\\myTable.txt	-

4.65.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
timeTable	TABLE_1D		Name of the table to filled in	-

4.65.6 Guidelines

If the file path is absolute, it is important to ensure that the slash symbols in the pathname are doubled in the string data called filename. For example:

"C:\\DataFiles\\myTable.txt"

4.65.7 Formulation

In the INIT block of the component, the EL function EcoReadTAble1D is called and loads the nt column of the file in the independent variable of the table timeTable, and the ny column of the file in the dependent variable of the table timeTable. The call of the function is the following:

EcoReadTable1D(filename ,nt, ny, timeTable)

The way of interpolating in the table is specified by the user with the enumerative data named tabmethod. We have defined four ways of "interpolating" in the table:

Method	Description
LinearInterpWithEvents	Linear interpolation with event detection
LinearInterpWithoutEvents	Linear interpolation without event detection
SplineInterp	Spline interpolation
StepConnect	Connection of the table points by means of steps

Then the output signal will depend on the interpolation method choosen by the user:

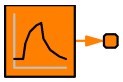
$$
\begin{aligned}
s\_out.signal[i] = {} & IF\ (tabmethod = LinearInterpWithEvents) && timeTableInterp(TIME, timeTable) \\
& ELSEIF\ (tabmethod = LinearInterpWithoutEvents) && linearInterp1D(timeTable, TIME) \\
& ELSEIF\ (tabmethod = SplineInterp) && splineInterp1D(timeTable, TIME) \\
& ELSE && timeTableStep(TIME, timeTable)
\end{aligned}
$$

4.66 SourceExp

4.66.1 Description

This component is inherited from component MO. It generates rising and falling exponential signals.

4.66.2 Symbol



SourceExp

4.66.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.66.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.66.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Offset	REAL	0	Offset of output signal	-
Tstart	REAL	0.5	Starting time of signal generation	s
fallTimeConst	REAL	0.1	Fall time constant	s
outMax	REAL	1	Height of output for infinite riseTime	-
riseTime	REAL	0.5	Rise Time	s
riseTimeConst	REAL	0.1	Rise time constant	s

4.66.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
y_riseTime	REAL		Maximum height reached by the exponential signal	

4.66.7 Formulation

The output signals are calculated with the following expression:

$$s\_out.signal[i] = \begin{cases} IF(TIME < Tstart) & Offset \\ ELSEIF(TIME < Tstart + t_{rise}) & Offset + s\_out_{\max} \cdot \left(1 - e^{-\left(\frac{TIME - Tstart}{K_r}\right)}\right) \\ ELSE & Offset + y_r \cdot e^{-\left(\frac{TIME - Tstart - t_{rise}}{K_f}\right)} \end{cases}$$

where:

Offset = the signal offset

s_outmax = the height of the output signal for infinity rise time (trise)

Kr = the rise time constant

Kf = the fall time constant

trise = the rise time

yr = the maximum height reached by the exponential signal:

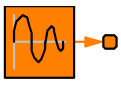
$$y_r = s\_out_{\max} \cdot \left(1 - e^{-\left(\frac{t_{rise}}{K_r}\right)}\right)$$

4.67 SourceExpSine

4.67.1 Description

This component is inherited from component MO. It generates exponentially damped sine signals.

4.67.2 Symbol



SourceExpSine

4.67.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.67.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.67.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Amp	REAL	1	Signal amplitude	-
Damping	REAL	1	Damping coefficients of sine waves	1/s
Offset	REAL	0	Offset of output signal	-
Phase	REAL	0	Signal phase	rad
Tstart	REAL	0	Starting time of signal generation	s
frec	REAL	1 / (2 * PI)	Signal frecuency	Hz

4.67.6 Formulation

The output signal is determined by:

$$s\_out.signal[i] = IF(TIME < Tstart) \quad Offset$$
$$ELSE \quad Offset + A \cdot e^{-(TIME-Tstart)\delta} \cdot \sin(2 \cdot \pi \cdot f \cdot (TIME - Tstart) + \phi)$$

where:

Offset = offset of the output signal

A = the signal amplitude

Tstart = the starting time of the signal generation

ϕ = the signal phase

f = the signal frequency

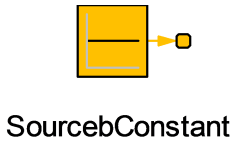
δ = the damping coefficient of the sine wave

4.68 SourcebConstant

4.68.1 Description

This component is inherited from component bMO. It generates constant Boolean signals. The component outputs the value defined by the user in the Boolean data k.

4.68.2 Symbol



4.68.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.68.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

4.68.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
k	BOOLEAN	TRUE	Constant output Boolean value	-

4.68.6 Formulation

The output signals will take the value of the Boolean data k

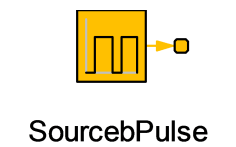
$$s\_out.signal[i] = \begin{cases} IF(k == TRUE) & TRUE \\ ELSE & FALSE \end{cases}$$

4.69 SourcebPulse

4.69.1 Description

This component is inherited from component bMO. It generates Boolean pulse signals.

4.69.2 Symbol



4.69.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.69.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

4.69.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Period	REAL	10	Period of the pulse	s
Tstart	REAL	5	Starting time of signal generation	s
pulseWidth	REAL	5	Pulse width	s

4.69.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
State	BOOLEAN		Auxiliar Boolean variable	-

4.69.7 Guidelines

The users specify the pulse width, the period of the pulse and the starting time for the signal generation. Before the starting time data Tstart, the outlet signals generated are FALSE.

4.69.8 Formulation

The first TRUE pulse occurs when the simulation time is greater than the paremeter named Tstart.

$$WHEN(TIME > Tstart)$$
$$State = TRUE$$
$$s\_out.signal[i] = TRUE$$

The duration of the TRUE pulses is defined with the data named pulseWidth and they are generated with a period of time defined by the data named Period. The modelling of this component is given by the following discrete sentences:

$$WHEN(State)$$
$$State = FALSE \ \ AFTER \ \ pulseWidth$$
$$s\_out.signal[i] = FALSE \ \ AFTER \ \ pulseWidth$$

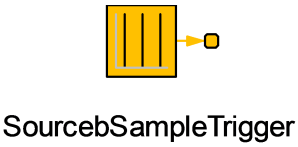
$$WHEN(State = FALSE \ \ AND \ TIME > Tstart)$$
$$State = TRUE \ \ AFTER \ (Period - pulseWidth)$$
$$s\_out.signal[i] = TRUE \ \ AFTER \ (Period - pulseWidth)$$

4.70 SourcebSampleTrigger

4.70.1 Description

This component is inherited from component bMO. It generates Boolean sample trigger signals.

4.70.2 Symbol



4.70.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.70.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

4.70.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Period	REAL	2	Period of sample	s
Tstart	REAL	1	Starting time of signal generation	s

4.70.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
state	BOOLEAN		Auxiliar Boolean variable	-

4.70.7 Guidelines

The users specify the starting time for the signal generation and the period at which the users want to sample the signals. Before the starting time the outlet signals generated are FALSE.

4.70.8 Formulation

The first sample trigger pulse occurs when the simulation time is greater than the paremeter named Tstart.

$$WHEN(TIME > Tstart)$$
$$State = TRUE$$
$$s\_out.signal[i] = TRUE$$

TRUE sample trigger pulse signals will be generated every time period defined with the data Period. The modelling of this component is given by the following discrete sentences:

$$WHEN(State == TRUE)$$
$$s\_out.signal[i] = TRUE$$
$$State = FALSE$$
$$State = TRUE \ \ AFTER \ \ Period$$

$$WHEN(State == FALSE)$$
$$s\_out.signal[i] = FALSE \;\; AFTER \; 10^{-6}$$

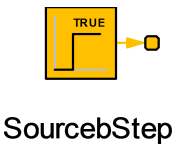
where State is an auxiliary Boolean variable.

4.71 SourcebStep

4.71.1 Description

This component is inherited from component bMO. It generates Boolean step signals.

4.71.2 Symbol



SourcebStep

4.71.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_out	CONST INTEGER	1	Dimension of outputs

4.71.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_out	bool_signal	(n = n_out)	OUT	Outlet signal

4.71.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
Tstart	REAL	1	Starting time of signal generation	s

4.71.6 Guidelines

The users specify the starting time for the signal generation. Before this starting time the outlet signals generated are FALSE.

4.71.7 Formulation

This component generates a TRUE signal when the time is greater than the starting time set by the user, otherwise the output signal will be FALSE.

$$WHEN(TIME > T_{start}) \quad s\_out.signal[i] = TRUE$$

where:

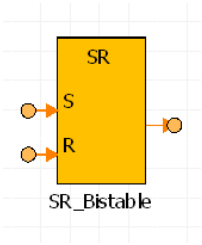
Tstart = the starting time for the generation of the output TRUE signal

4.72 SR_Bistable

4.72.1 Description

This component is inherited from component bMI2MOs. It represents a set dominant bistable as defined in IEC 61131-3.

4.72.2 Symbol



4.72.3 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
init_val	BOOLEAN	FALSE	Initial output value	-

4.72.4 Formulation

The logic algorithm implemented is the following:

```
WHEN(s_in_2.signal[1] == TRUE OR (s_out.signal[1] == TRUE AND s_in_1.signal[1] == FALSE)) THEN
   s_out.signal[1] = TRUE
END WHEN
WHEN(s_in_2.signal[1] == FALSE AND s_in_1.signal[1] == FALSE ) THEN
   s_out.signal[1] = FALSE
END WHEN
WHEN(s_in_2.signal[1] == FALSE AND s_out.signal[1] == FALSE ) THEN
   s_out.signal[1] = FALSE
END WHEN
```

4.73 StateSpace

4.73.1 Description

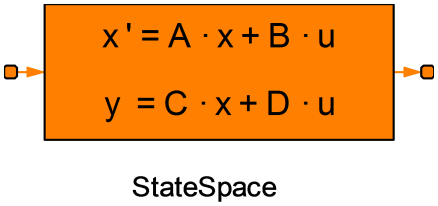
This component is inherited from component MIMO. It defines the relationship between the input port signals and the output port signals in a state space form.

It implements a system whose behavior is defined by the following equation system:

$$x' = A \cdot x + B \cdot u$$
$$y = C \cdot x + D \cdot u$$

where x is the state vector, u is the input vector and y is the output vector.

4.73.2 Symbol



4.73.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER		Number of states
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

4.73.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n_in)	IN	Inlet signal
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.73.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
A[n,n]	REAL		Matrix A	
B[n,n_in]	REAL		Matrix B	
C[n_out,n]	REAL		Matrix C	
D[n_out,n_in]	REAL		Matrix D	
x_o[n]	REAL		Initial condition vector	

4.73.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
x[n]	REAL		States	

4.73.7 Guidelines

The dimension of the outputs can be different to the dimension of the inputs. The matrix has the following dimensions:

A must be an n by n matrix, where n is the number of states.

B must be an n by n_in matrix, where n_in is the number of inputs

C must be a n_out by n matrix, where n_out is the number of outputs

D must be a n_out by n_in matrix

The initial conditions of state variables are defined by means of the vector x_o.

4.73.8 Formulation

The output signals are calculated according to the following equations:

$$\frac{\partial x_i}{\partial t} = \sum_{j=1}^{j=n} A_{i,j} \cdot x_j + \sum_{j=1}^{j=n_{in}} B_{i,j} \cdot s_{in,j}$$

$$s_{out,i} = \sum_{j=1}^{j=n} C_{i,j} \cdot x_j + \sum_{j=1}^{j=n_{in}} D_{i,j} \cdot s_{in,j}$$

where:

sin, j = input signals

sout,j = output signals

xj = state variables

A, B, C and D = state matrices

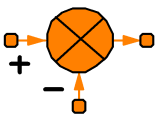
4.74 Subtraction

4.74.1 Description

This component is inherited from component MI2MOs.

This component calculates the output signals as the substraction between the analog signals of the first input port and the analog signals of the second input port.

4.74.2 Symbol



Subtraction

4.74.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.74.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in_1	analog_signal	(n = n)	IN	First inlet analog signal port
s_in_2	analog_signal	(n = n)	IN	Second inlet analog signal port
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.74.5 Guidelines

The output signal responds instantaneously to changes in the input signals

4.74.6 Formulation

The output is computed as the subtraction between the two input signals

s_out.signal[j] = s_in_1.signal[j] - s_in_2.signal[j]

4.75 Switch

4.75.1 Description

This component propagates one of the analog inputs to the analog output depending on the value of the Boolean input (the middle port). If the signal of the second input port (middle port) is TRUE the component propagates the first input signal (upper port), otherwise, it propagates the third input signal (lower port).

4.75.2 Symbol

Switch

4.75.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.75.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_b_in	bool_signal	(n = n)	IN	Boolean inlet signal
s_in[2]	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.75.5 Guidelines

4.75.6 Formulation

If the s_b_in port signal is TRUE, the output signal is set to the value of s_in[1] port signal, otherwise it is set to the value of s_in[2] port signal.

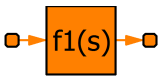
$$s\_out.signal[i] = \begin{cases} IF(s\_b\_in.signal[i]) & s\_in[1].signal[i] \\ ELSE & s\_in[2].signal[i] \end{cases}$$

4.76 TF_1stOrder

4.76.1 Description

This component is inherited from component MIMOs. It represents a first order transfer function between the input port signals and the output port signals.

4.76.2 Symbol



TF_1stOrder

4.76.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER		Number of states
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

4.76.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet analog signal port
s_out	analog_signal	(n = n)	OUT	Outlet analog signal port

4.76.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
gain[n]	REAL		Gain	-
output_o[n]	REAL		Initial output	-
tau[n]	REAL		Time constant	s

4.76.6 Guidelines

The users can use this component to represent a first order lag.

4.76.7 Formulation

The outlet signals are calculated as follows:

$$\frac{d\,s_{out}}{dt} = \frac{k \cdot s_{in} - s_{out}}{\tau}$$

where:

sin = the value of the input signal

sout = the value of the output signal

τ = the value of the time constant

k = the gain

4.77 TF_2ndOrder

4.77.1 Description

This component is inherited from component MIMOs. It represents a second order transfer function between the input port signals and the output port signals.

$$y = \frac{k}{\left(\frac{s}{\omega}\right)^2 + 2 \cdot \delta \cdot \left(\frac{s}{\omega}\right) + 1} \cdot u$$

where:

y = the outlet signal

u = the input signal

k = the gain

ω = the angular frequency

δ = the damping

4.77.2 Symbol

TF_2ndOrder

4.77.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER		Number of states
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

4.77.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.77.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
damp[n]	REAL		Damping	-
frec[n]	REAL		Angular frecuency	Hz
gain[n]	REAL		Gain	-
output_o[n]	REAL		Initial values of output	

4.77.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
z[n]	REAL		Array of dynamic variables	-

4.77.7 Guidelines

The users can use this component to represent a second order lag.

4.77.8 Formulation

The outlet signals are calculated as follows:

$$y'' = \omega^2 \cdot \left(k \cdot u - \left(\frac{2 \cdot \delta}{\omega} \right) \cdot y' - y \right)$$

where:

y″ = the second derivative of the outlet signal

y′ = the first derivative of the outlet signal

y = the outlet signal

u = the input signal

k = the gain

ω = the angular frequency

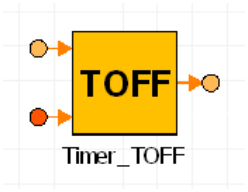
δ = the damping

4.78 Timer_TOFF

4.78.1 Description

This component is inherited from component bSISO. It represents a timer to delay a signal on deactivation.

4.78.2 Symbol



4.78.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
ET	analog_signal	(n = 1)	OUT	Output

4.78.4 Data

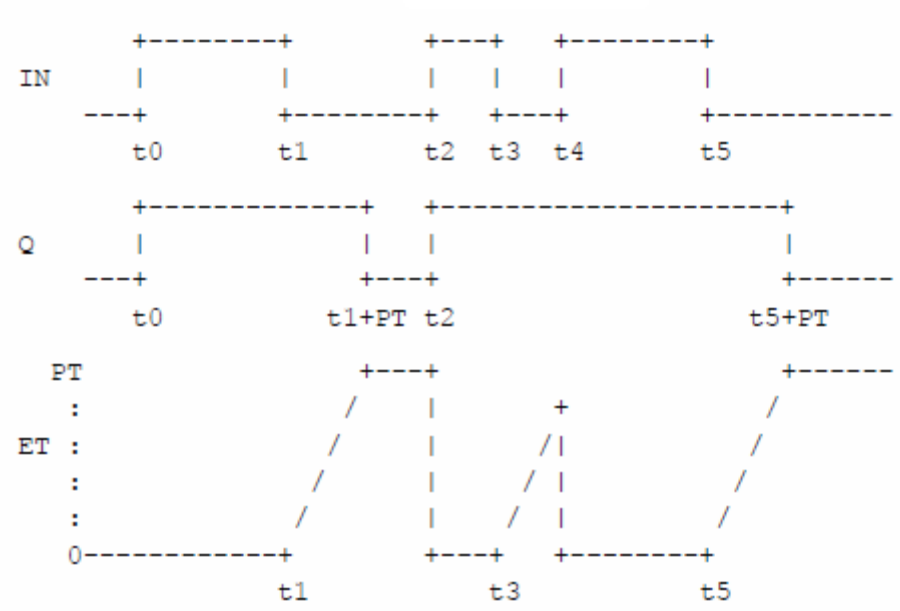
NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
PT	INTEGER	2	Time	s

4.78.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
TSTART	REAL	–	Start time	s
count	BOOLEAN	–	Internal variable	–

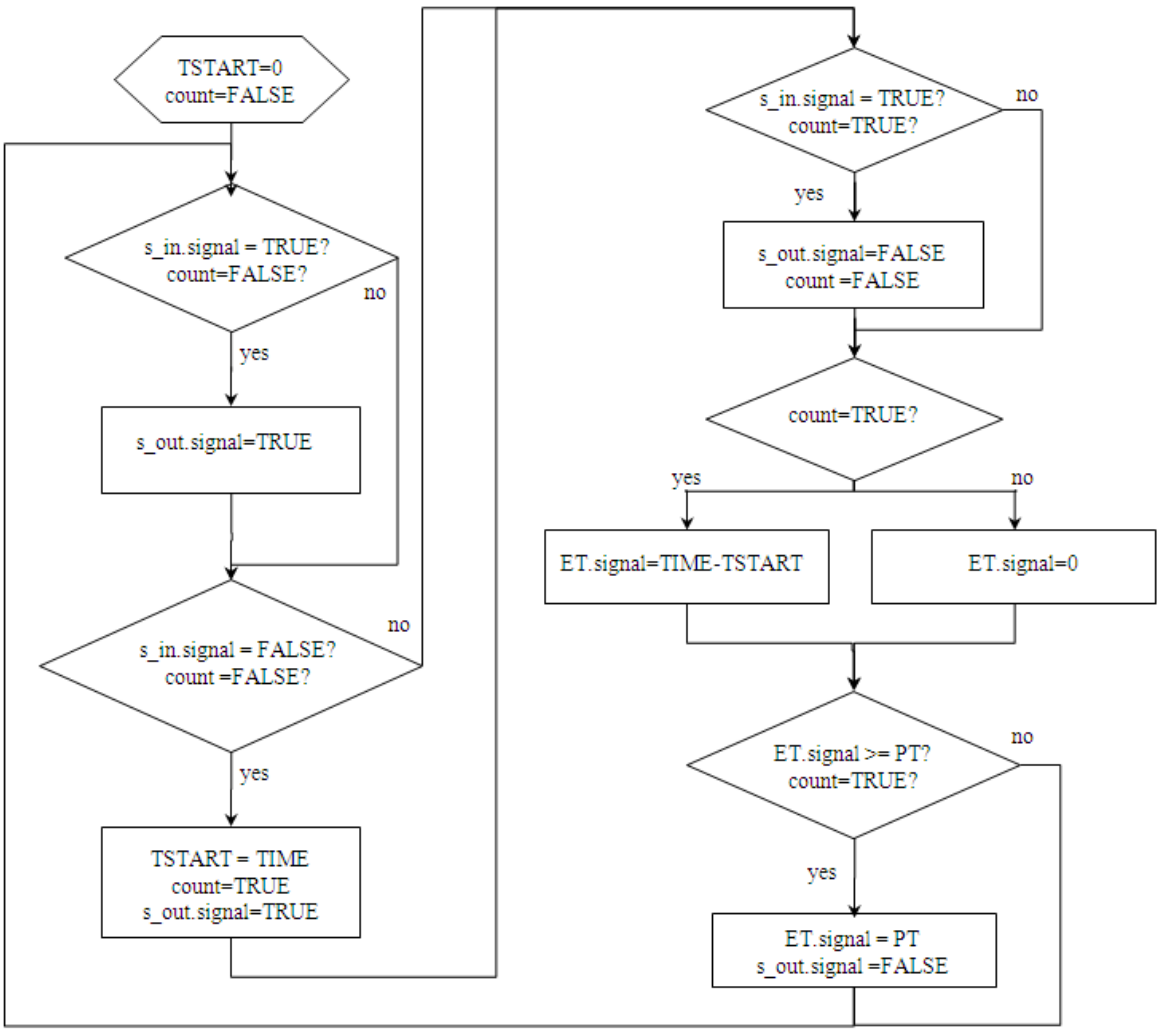
4.78.6 Guidelines

The timing diagram of the output as a function of the inputs is the following:



4.78.7 Formulation

The logic algorithm implemented is the following:



4.79 Timer_TON

4.79.1 Description

This component is inherited from component bSISO. It represents a timer to delay a signal on activation.

4.79.2 Symbol

Timer_TON

4.79.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
ET	analog_signal	(n = 1)	OUT	Output

4.79.4 Data

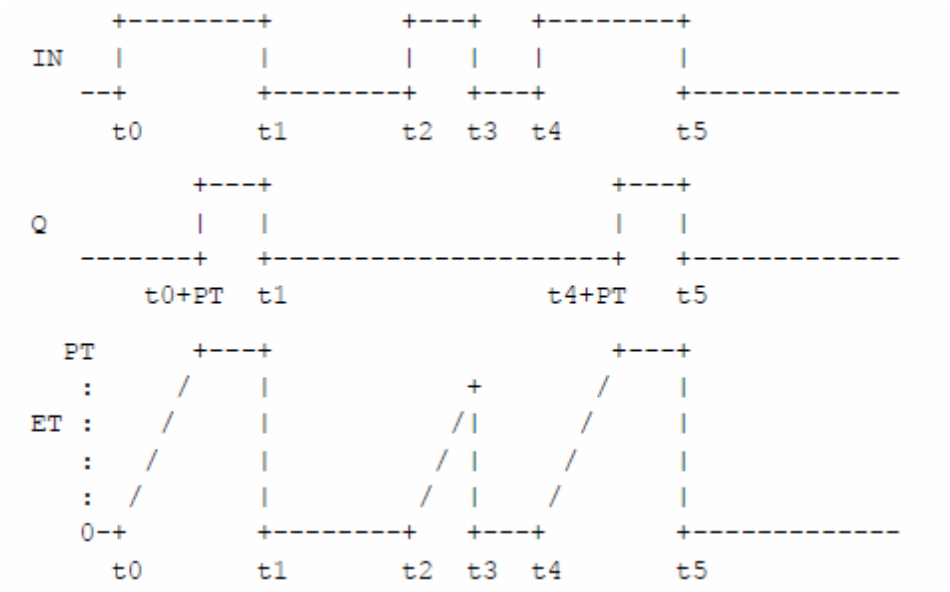
NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
PT	INTEGER	2	Time	s

4.79.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
TSTART	REAL	–	Start time	s
count	BOOLEAN	–	Internal variable	–

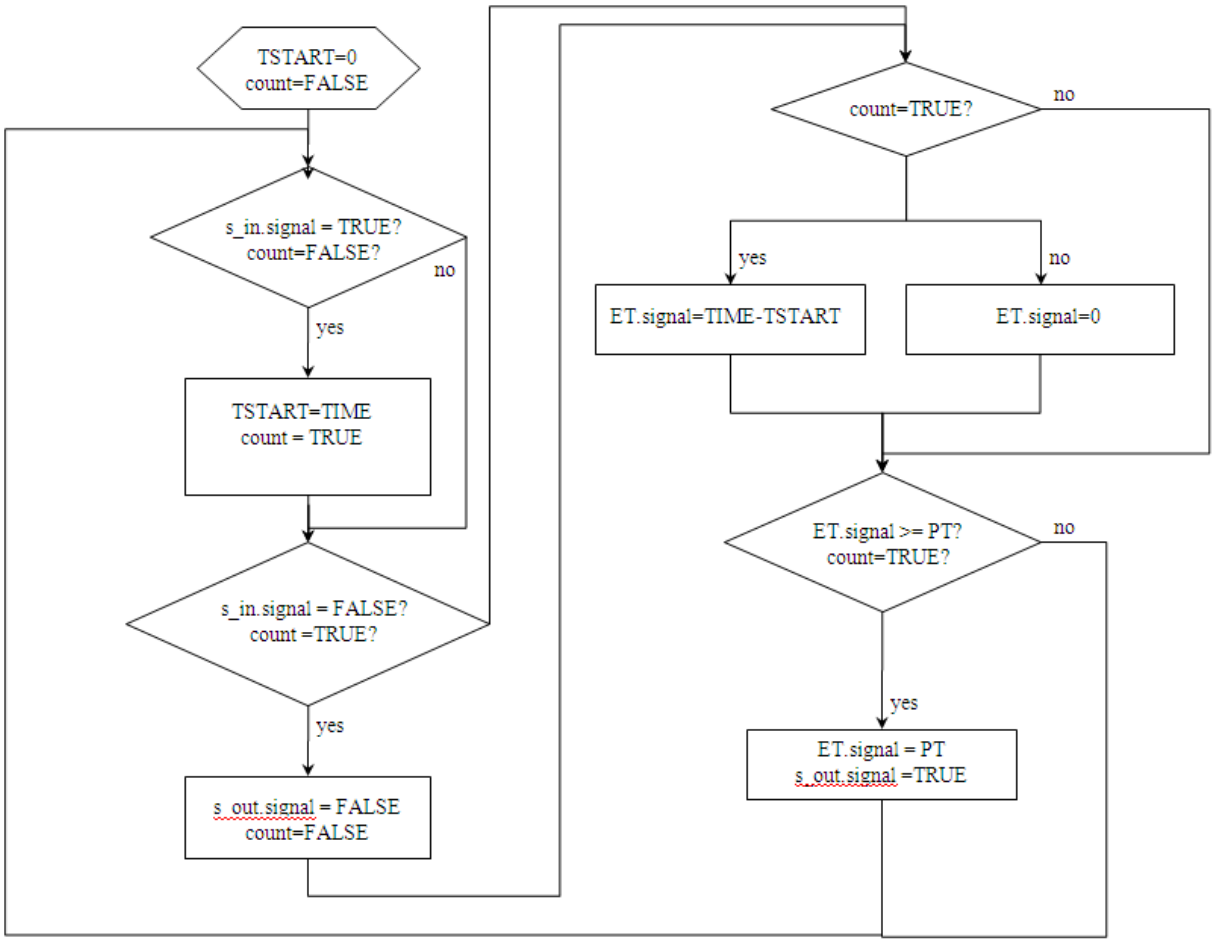
4.79.6 Guidelines

The timing diagram of the output as a function of the inputs is the following:



4.79.7 Formulation

The logic algorithm implemented is the following:

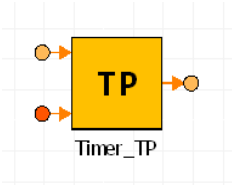


4.80 Timer_TP

4.80.1 Description

This component is inherited from component bSISO. It represents a timer to delay a signal on deactivation as defined in IEC 61131-1.

4.80.2 Symbol



4.80.3 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
ET	analog_signal	(n = 1)	OUT	Output

4.80.4 Data

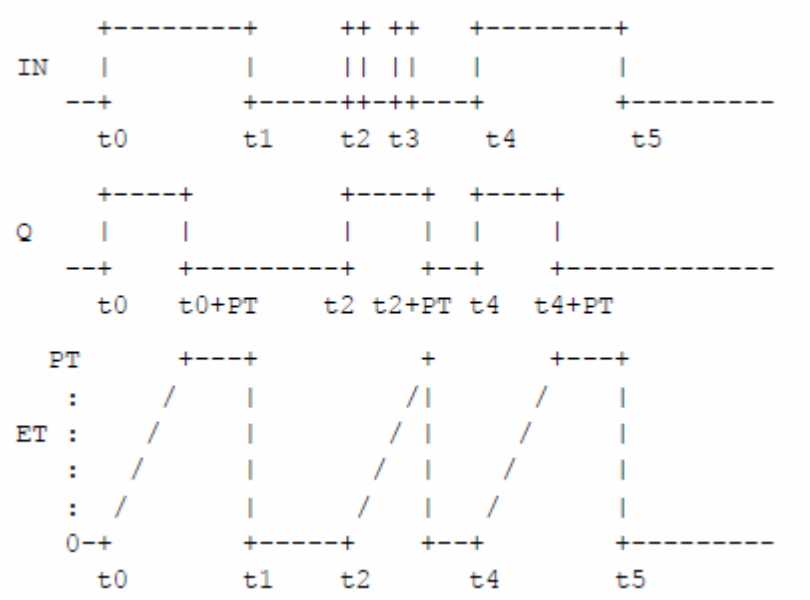
NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
PT	INTEGER	2	Time	s

4.80.5 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
TSTART	REAL	–	Start time	
count	BOOLEAN	–	Internal variable	

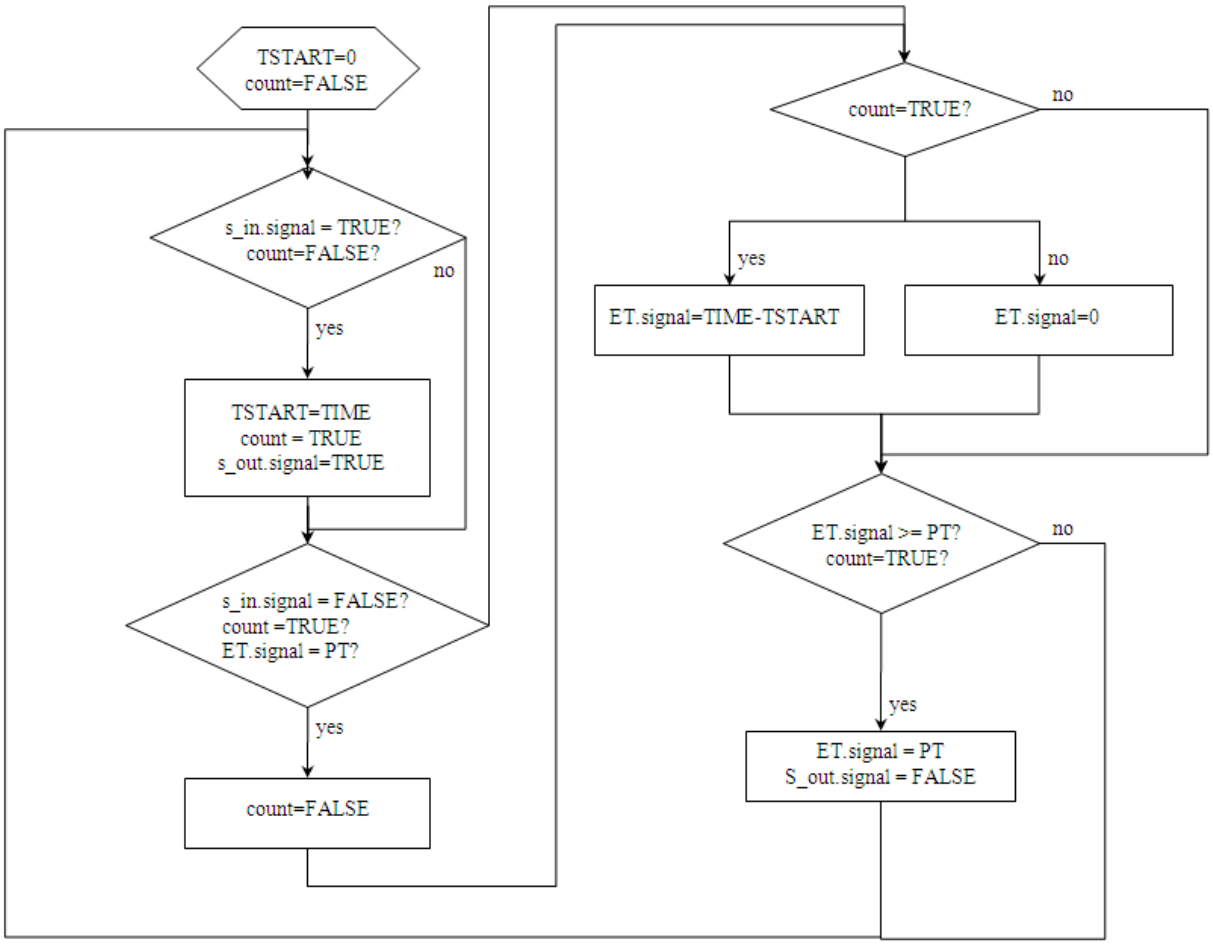
4.80.6 Guidelines

The timing diagram of the output as a function of the inputs is the following:



4.80.7 Formulation

The logic algorithm implemented is the following:



4.81 TransferFunction

4.81.1 Description

This component is inherited from component SISO.

This component type, named TransferFcn, provides a representation of linear transfer functions in the form of a ratio of polynomials in the Laplace operator, s.

There is no limit on the numerator and denominator orders, provided that the overall order of the numerator is less than or equal to the order of the denominator.

Transfer Function Theory

If the differential equation that describes a block of the system is known and is linear, the ratio of the Laplace transform of the output to the Laplace transform of the input is called the transfer function.

Let us suppose that the following linear differential equation with constant coefficients relates the input x to the output y of one block of the system.

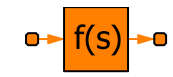
$$q_3 \frac{d^3y}{dt^3} + q_2 \frac{d^2y}{dt^2} + q_1 \frac{dy}{dt} + q_o y = p_1 \frac{dx}{dt} + p_o x$$

The Laplace transform (transfer function) will be:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{p_1 s + p_o}{q_3 s^3 + q_2 s^2 + q_1 s + q_o}$$

All initial conditions are taken as zero.

4.81.2 Symbol



TransferFunction

4.81.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_den	CONST INTEGER	1	Order of the denominator polynomial
n_num	CONST INTEGER	0	Order of the numerator polynomial

4.81.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.81.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
output_o	REAL	0	Initial value of the output	-
p[n_num + 1]	REAL		Coefficients of numerator polynomial: p[1]*sˆn+...+p[n+1]	-
q[n_den + 1]	REAL		Coefficients of denominator polynomial: q[1]*sˆn+...+q[n+1]	-

4.81.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.81.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
j	INTEGER			
z[n_den]	REAL		Array of dynamic variables	-

4.81.8 Guidelines

The users have to take care and set an order of the numerator that must be lower than the order of the denominator. The highest order coefficient in the denominator can not be zero.

The user provides the value of the order of the numerator polynomial, n_num, the value of the order of the denominator polynomial, n_dem, and the values of the coefficients of the numerator and denominator polynomials p[n_num+1] and q[n_dem+1], respectively.

The transfer function polynomials are in the form of the constant term first, and the higher order terms after, with any missing term being given as a zero coefficient. For example, the transfer function, $\frac{s}{s^2+4}$ is defined by:

TransferFunction (nnum=1, ndem=2) transfun

(p = {0, 1} ,

q = {4, 0, 1} ,

output_o = 0)

where:

p = the coefficients of the numerator polynomial

q = the coefficients of the denominator polynomial

4.81.9 Formulation

Let us assume that the transfer function is:

$$\frac{y(s)}{x(s)} = \frac{p_m s^m + p_{m-1} s^{m-1} + \cdots + p_1 s + p_0}{q_n s^n + q_{n-1} s^{n-1} + \cdots + q_1 s + q_0}$$

with x(t) being the input signal and y(t) being the output signal.

We will see how the solution of the above problem can be built from the solution of this other problem:

$$\frac{y_0(s)}{x(s)} = \frac{1}{q_n s^n + q_{n-1} s^{n-1} + \cdots + q_1 s + q_0} \qquad (1)$$

that represents the following ordinary differential equation:

$$q_n \frac{d^n y_0}{dt^n} + q_{n-1} \frac{d^{n-1} y_0}{dt^{n-1}} + \cdots + q_1 \frac{dy_0}{dt} + q_0 y_0 = x(t) \qquad (2)$$

The solution of this ordinary differential equation can be rewritten as a set of n coupled first order differential equations by making the following change of variables:

$$y_0(t) = z_1$$

$$\frac{dy_0(t)}{dt} = \frac{dz_1(t)}{dt} = z_2$$

$$\frac{d^2 y_0(t)}{dt^2} = \frac{dz_2(t)}{dt} = z_3$$

$$\frac{d^{n-1}y_0(t)}{dt^2} = \frac{dz_{n-1}(t)}{dt} = z_n$$

$$\frac{d^n y_0(t)}{dt^n} = \frac{dz_n(t)}{dt}$$

which yields, making substitutions in (2), the following differential equation:

$$q_n \frac{dz_n(t)}{dt} + q_{n-1}z_n(t) + \cdots + q_2 z_3(t) + q_1 z_2(t) + q_0 z_1(t) = x(t)$$

The above equations can be taken as a system of n first order differential equations:

$$\frac{d\,z_1}{dt} = z_2$$

$$\frac{d\,z_2}{dt} = z_3(t)$$

.

.

.

$$\frac{d\,z_n}{dt} = \frac{1}{q_n}\left[x(t) - q_{n-1}z_n(t) - q_{n-2}z_{n-1}(t) - \cdots - q_2 z_3(t) - q_1 z_2(t) - q_0 z_1(t)\right]$$

This system can be solved using any numerical resolution method and the results would be:

$$z_n(t), \quad z_{n-1}(t), \quad \cdots, \quad z_3(t), \quad z_2(t) \quad and \quad z_1(t)$$

And for our first problem (I): y0(t) = z1(t)

Now the solution of the complete equation

$$q_n \frac{d^n y}{dt^n} + q_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + q_1 \frac{dy}{dt} + q_0 y = p_m \frac{d^m x}{dt^m} + p_{m-1}\frac{d^{m-1}x}{dt^{m-1}} + \cdots + p_1 \frac{dx}{dt} + p_0 x$$

can be written as a sum of the solutions for the different right terms because of the linearity of the equation

y(t) = A0 y0(t)+A1y1(t) + ... + Am-1 ym-1(t)+Am ym(t)

where y0 (t), y1(t), ..., ym-1(t), and ym(t) are the solutions for the following equations:

$$
\begin{array}{ccccccccccc}
q_n\dfrac{d^n y_0(t)}{dt^n} & + & q_{n-1}\dfrac{d^{n-1} y_0(t)}{dt^{n-1}} & +\cdots+ & q_1\dfrac{dy_0(t)}{dt} & + & q_0 y_0(t) & = & x(t) \\
q_n\dfrac{d^n y_1(t)}{dt^n} & + & q_{n-1}\dfrac{d^{n-1} y_1(t)}{dt^{n-1}} & +\cdots+ & q_1\dfrac{dy_1(t)}{dt} & + & q_0 y_1(t) & = & \dfrac{dx(t)}{dt} \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots \\
q_n\dfrac{d^n y_{m-1}(t)}{dt^n} & + & q_{n-1}\dfrac{d^{n-1} y_{m-1}(t)}{dt^{n-1}} & +\cdots+ & q_1\dfrac{dy_{m-1}(t)}{dt} & + & q_0 y_{m-1}(t) & = & \dfrac{d_{m-1}x(t)}{dt^{m-1}} \\
q_n\dfrac{d^n y_m(t)}{dt^n} & + & q_{n-1}\dfrac{d^{n-1} y_m(t)}{dt^{n-1}} & +\cdots+ & q_1\dfrac{dy_m(t)}{dt} & + & q_0 y_m(t) & = & \dfrac{d_m x(t)}{dt^m}
\end{array}
$$

The solution of the first equation has been already calculated, and it is

yo(t) = z1

Making the derivative of the first equation, it is possible to see the solution of the second equations:

$$\frac{d}{dt}\left[q_n\frac{d^n y_0}{dt^n}+q_{n-1}\frac{d^{n-1}y_0}{dt^{n-1}}+\cdots+q_1\frac{dy_0}{dt}+q_0 y_0\right]=d\frac{x(t)}{dt}$$

$$q_n\frac{d^n}{dt^n}\left[\frac{dy_0}{dt}\right]+q_{n-1}\frac{d^{n-1}}{dt^{n-1}}\left[\frac{dy_0}{dt}\right]+\cdots+q_1\frac{d}{dt}\left[\frac{dy_0}{dt}\right]+q_0\left[\frac{dy_0}{dt}\right]=\frac{dx(t)}{dt}$$

It can be checked that $\dfrac{dy_0}{dt}$ satisfies the second equations, so

$$y_1=\frac{dy_0}{dt}=z_2$$

And finally:

y0 = z1(t)

y1 = z2(t)

ym-1 = zm-1(t)

ym = zm(t)

zm-1(t) has been obtained with the solution of our first differential equation system, because:

$$y(t)=p_0 z_1(t)+p_1 z_2(t)+\cdots+p_{m-1}z_m(t)+p_m\frac{dz_m(t)}{dt}$$

$\dfrac{dz_m(t)}{DT}$ **will be** $z_{m+1}(t)$ if $m<n$ **or** $\dfrac{dz_m(t)}{dt}$ **if** $m=n$

$$\frac{dz_n(t)}{dt}=\frac{x(t)-q_{n-1}z_n(t)-\cdots-q_0 z_1(t)}{B_n}$$

$$\dot{z}_{nd-1} = \frac{x(t) - \sum_{i=1}^{nd-2} z_i \cdot q_i}{q_{nd}}$$

$$\dot{z}_{nd} = 0$$

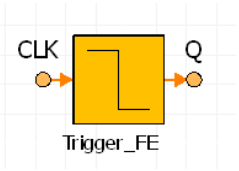
$$output\_var = \begin{cases} \sum_{i=1}^{nn-1} p_i \cdot z_i + p_{nn} \cdot z_{nn} & \textbf{if } nn < nd \\ \sum_{i=1}^{nn-1} p_i \cdot z_i + p_{nn} \cdot \dot{z}_{nn-1} & \textbf{if } nn = nd \end{cases}$$

4.82 Trigger_FE

4.82.1 Description

This component is inherited from component bSISO. It represents a component that detects falling edges in boolean signals.

4.82.2 Symbol



4.82.3 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
CLK_OLD	BOOLEAN	FALSE	Last value of the input signal	
frac	REAL	–	Number of checked time intervals	
start	REAL	–	Start time	

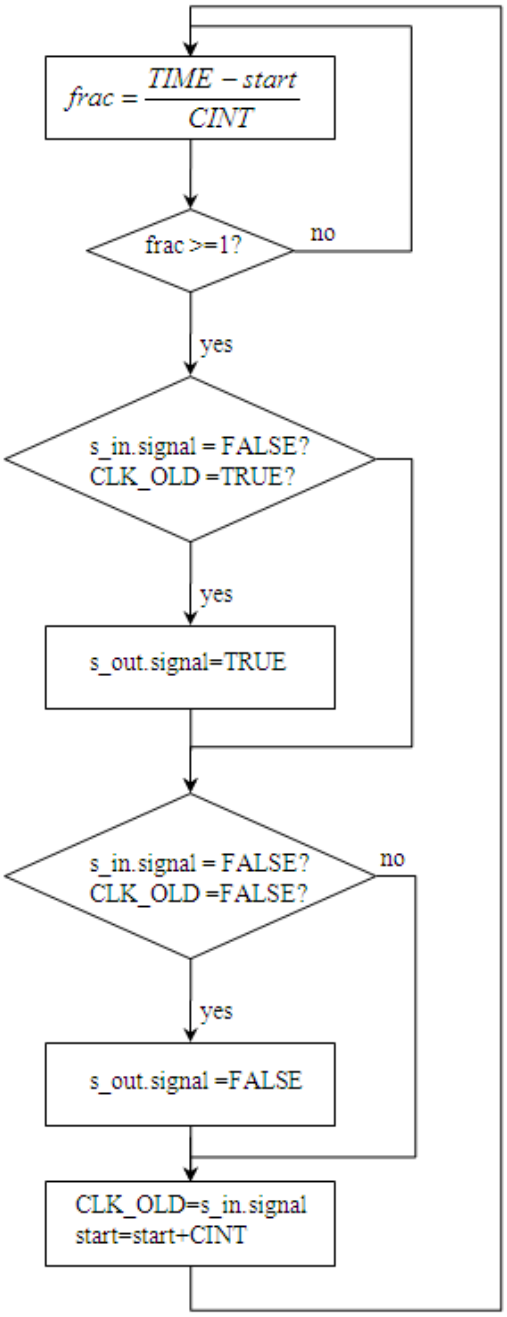
4.82.4 Guidelines

The behaviour of the component corresponds to the following rule:

- The Q output of an F_TRIG function block shall stand at the boolean-value 1 from one execution of the function block to the next, following the 1 to 0 transition of the CLK input, and shall return to 0 at the next execution.

4.82.5 Formulation

The logic algorithm implemented is the following:

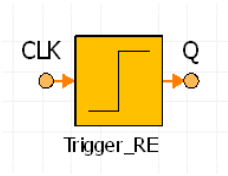


4.83 Trigger_RE

4.83.1 Description

This component is inherited from component bSISO. It represents a component that detects rising edges in boolean signals.

4.83.2 Symbol



4.83.3 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
CLK_OLD	BOOLEAN	–	Last value of the input signal	
frac	REAL	–	Number of checked time intervals	
start	REAL	–	Start time	

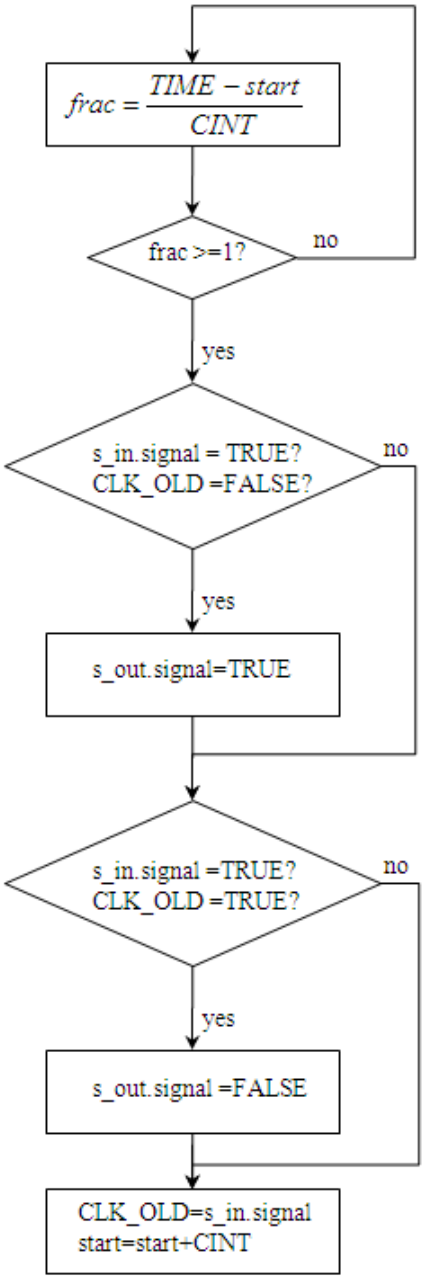
4.83.4 Guidelines

The behaviour of the component corresponds to the following rule:

- The Q output of an R_TRIG function block shall stand at the boolean-value 1 from one execution of the function block to the next, following the 0 to 1 transition of the CLK input, and shall return to 0 at the next execution.

4.83.5 Formulation

The logic algorithm implemented is the following:

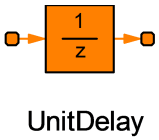


4.84 UnitDelay

4.84.1 Description

This component is inherited from component dMIMOs. It delays the input signal one sample period. The output signal will be the input signal of the previous sample time. This component is equivalent to the z-1 discrete-time operator.

4.84.2 Symbol



4.84.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.84.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.84.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s
y_o[n]	REAL		Initial condition	

4.84.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE		

4.84.7 Guidelines

If the users desire an undelayed sample and hold function, then they must use a ZeroOrderHold component, or if a delay of greater than one sample period time is desired, they must use the dTransferFunction component.

4.84.8 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

When the sample is TRUE the new output signal is calculated as the input signal delayed a time period (dt). The when block to compute this is shown next:

```
WHEN (sample == TRUE) THEN
   FOR(i IN 1, n)
         s_out.signal[i] = s_in.signal[i] AFTER dt
   END FOR
END WHEN
```

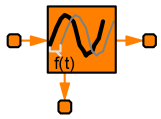
4.85 VarDelay

4.85.1 Description

This component is inherited from component MIMOs. It can be used to simulate a variable time delay. The component has two inputs: the first input is the signal that passes through the component and the second input is the time delay.

The time delay is limited by a maximum delay time data called tdelay_max.

4.85.2 Symbol



VarDelay

4.85.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.85.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_TimeDelay	analog_signal	(n = n)	IN	Time delay port
s_in	analog_signal	(n = n)	IN	Inlet signals
s_out	analog_signal	(n = n)	OUT	Outlet signals

4.85.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
tdelay_max[n]	REAL		Maximum delay time	s

4.85.6 Formulation

The output signals are computed delaying the input signals with the signals of the port s_TimeDelay. The value of the signals of the s_TimeDelay is limited by a maximum delay time defined by the data tdelay_max:

s_out.signal[i] = delay(s_in.signal[i], bound(s_TimeDelay.signal[i], 1e-12, tdelay_max[i])

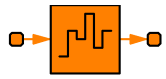
4.86 ZeroOrderHold

4.86.1 Description

This component is inherited from component dMIMOs. It implements a sample-and-hold function that operates at the specified sample time.

This component can be used for discretizing one or more signals. The output signals are identical to the sampled input signal at the sample time instants and the component holds the output signals at the value of the last sample instant during the sample period.

4.86.2 Symbol



4.86.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of inputs and outputs

4.86.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.86.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s

4.86.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE		

4.86.7 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
```

```
END WHEN
```

When the sample time instant takes place the input signals are sampled and the output signals are equal to the input signals at that sample time instant:

```
WHEN (sample == TRUE) THEN
   FOR (i IN 1, n)
      s_out.signal[i] = s_in.signal[i]
   END FOR
END WHEN
```

4.87 ZeroPole

4.87.1 Description

This component is inherited from component SISO. It implements a system with the specified zeros, poles and gain in terms of the Laplace operator s.

A transfer function can be expressed in factored or zero-pole-gain form:

$$K \cdot \frac{(s - Z(1)) \cdot (s - Z(2)) \cdots (s - Z(n))}{(s - P(1)) \cdot (s - P(2)) \cdots (s - P(n))}$$

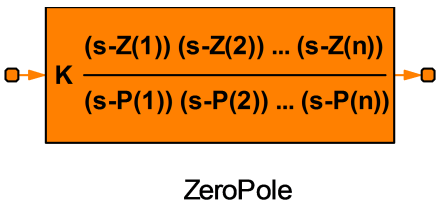
where:

Z represents the zeros

P represents the poles

K represents the gain

4.87.2 Symbol



4.87.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
npoles	CONST INTEGER	1	Number of poles
nzeros	CONST INTEGER	1	Number of zeros

4.87.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.87.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
K	REAL	1	Gain	
option	ENUM InitOption	InitialStates	Initialization of discrete component: states or output	
poles[npoles,2]	REAL		Values of the poles: real part, imaginary part	
x_o[npoles]	REAL		Array with initial values of the states	
y_o	REAL	0	Initial value for the output	
zeros[nzeros,2]	REAL		Values of the zeros: real part, imaginary part	

4.87.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.87.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
j	INTEGER		Index	
p[nzeros + 1]	REAL		Coefficients of the numerator in increasing powers of s	
q[npoles + 1]	REAL		Coefficients of the denominator in increasing powers of s	
x[npoles + 1]	REAL		State variables	

4.87.8 Guidelines

The user can specify how to initialized the component with the data named option:

option	Description
InitialStates	Initialization of the state variables of the component
InitialOutput	Initialization of the output signal of the component

4.87.9 Formulation

The derivative of the state variable j is equal to the state variable j+1:

$$x(j+1) = \frac{\partial x(j)}{\partial t}$$

The value of the input signals must be equal to the sum of the state variables multiplied by the corresponding denominator coefficient

$$s\_in.signal[1] = \sum_{1}^{npoles+1} x[i] \cdot q[i]$$

And the value of the output signas must be equal to the sum of the states variables multiplied by the corresponding numerator coefficient and the gain:

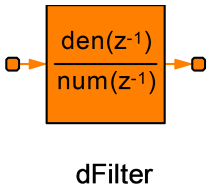
$$s\_out.signal[1] = \sum_{1}^{nzeros+1} x[i] \cdot p[i]$$

4.88 dFilter

4.88.1 Description

This component is inherited from component dSISO. It implements a discrete filter.

4.88.2 Symbol



4.88.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_den	CONST INTEGER	1	Order of denominator polinomial (1/z)
n_num	CONST INTEGER	0	Order of numerator polinomial (1/z)

4.88.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.88.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s
pdata[n_num+1]	REAL		Coefficients of numerator polinomial in ascending powers of 1/z	
qdata[n_den+1]	REAL		Coefficients of denominator polinomial in ascending powers of 1/z	

4.88.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.88.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
p[n_den + 1]	REAL		Coefficients of numerator polynomial in ascending powers of 1/z	
sample	BOOLEAN	FALSE		
x[n_den + 1]	REAL		Values of states	

4.88.8 Guidelines

The users specify the coefficients of the numerator and denominator polynomials in ascending power of z-1 as vectors using the pdata and qdata parameters.

The order of the denominator must be greater than or equal to the order of the numerator.

This component represents the method typically used by signal processing engineers who describe systems using polynomials in z-1 (the delay operator).

4.88.9 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

The coefficients of numerator polynomial in ascending powers of z-1 are computed as follows:

$$FOR(j=1; j \le n_{num}+1; j+1)$$
$$p[j] = pdata[j]$$
$$FOR(j=n_{num}+2; j \le n_{den}+1; j+1)$$
$$p[j] = 0$$

When the sample time instant takes place the output signal is calculated as follows:

The state variable x[1] is calculated before the output signal by:

$$x[1] = \frac{s\_in.signal[1] - \sum_{j=2}^{n\_den+1} x[j] \cdot qdata[j]}{qdata[1]}$$

$$s\_out.signal[1] = \sum_{j=1}^{n\_num+1} x[j] \cdot p[j]$$

And the remainder of states variables are calculated as:

$$FOR(j=1; j \le n\_den; j++)$$
$$x[n\_den+2-j] = x[n\_den+1-j]$$

4.89 dIntegrator

4.89.1 Description

This component is inherited from component dMIMOs. It represents a discrete integration scheme equivalent to the following z-domanin transfer function:

$$y = \frac{1}{z-1} \cdot u$$

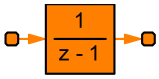
where y is the output signal and u is the input signal.

This component can be used in place of the continuous Integrator component when building a purely discrete system.

The user can select the following integration method to calculate the output signals:

- ForwardEuler
- BackwardEuler
- Trapezoidal

4.89.2 Symbol



dIntegrator

4.89.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
Method	CONST ENUM dIntegMethod	ForwardEuler	Integration method
n	CONST INTEGER	1	Dimension of inputs and outputs

4.89.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal ()
s_out	analog_signal	(n = n)	OUT	Outlet signal ()

4.89.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s
y_o[n]	REAL		Initial condition for the output	

4.89.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.89.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE		
x[n]	REAL		Values of states	

4.89.8 Guidelines

The user chooses the numerical integration method with the construction parameter Method:

ForwardEuler: Forward Euler method also known as Forward Rectangular, or left-hand approximation is a first order single-step method.

BackwardEuler: Euler backward integration method is also a first order single-step method.

Trapezoidal: The trapezoidal rule integration method is a second order single-step method.

4.89.9 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

The output signals are calculated at each sample time instant depending on the integration method selected by the user:

Integration Method	Equations
ForwardEuler	$s\_out.signal[i] = x[i]$ $x[i] = s\_out.signal[i] + dt \cdot s\_in.signal[i]$
BackwardEuler	$IF(TIME = 0)$ $\quad s\_out.signal[i] = x[i]$ $ELSE$ $\quad s\_out.signal[i] = x[i] + dt \cdot s\_in.signal[i]$ $x[i] = s\_out.signal[i]$
Trapezoidal	$IF(TIME = 0)$ $\quad s\_out.signal[i] = x[i]$ $ELSE$ $\quad s\_out.signal[i] = x[i] + \frac{dt}{2} \cdot s\_in.signal[i]$ $x[i] = s\_out.signal[i] + \frac{dt}{2} \cdot s\_in.signal[i]$

4.90 dStateSpace

4.90.1 Description

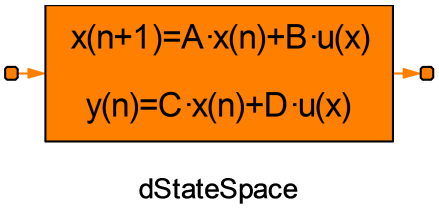
This component is inherited from component dMIMO. It represents a discrete time approximation of a state space system.

It implements a system whose behavior is define by the following equation system:

$$x(n+1) = A \cdot x(n) + B \cdot u(n)$$
$$y(n) = C \cdot x(n) + D \cdot u(n)$$

where x is the state vector, u is the input vector and y is the output vector.

4.90.2 Symbol



4.90.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_states	CONST INTEGER		Number of states
n_in	CONST INTEGER	1	Dimension of inputs
n_out	CONST INTEGER	1	Dimension of outputs

4.90.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n_in)	IN	Inlet signal
s_out	analog_signal	(n = n_out)	OUT	Outlet signal

4.90.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
A[n_states,n_states]	REAL		Matrix A of state space model	
B[n_states,n_in]	REAL		Matrix B of state space model	
C[n_out,n_states]	REAL		Matrix C of state space model	
D[n_out,n_in]	REAL		Matrix D of state space model	
dt	REAL	0.1	Sample time	s
x_o[n_states]	REAL		Initial conditions for output array	

4.90.6 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
sample	BOOLEAN	FALSE		
x[n_states]	REAL		States	
xn[n_states]	REAL		New states	

4.90.7 Guidelines

The dimension of outlet can be different to the dimension of the inlet. The matrix have the following dimensions:

- A must be an n_states by n_states matrix, where n_states is the number of states.
- B must be a n_states by n_in matrix, where n_in is the number of inputs
- C must be a n_out by n_states matrix, where n_out is the number of outputs
- D must be a n_out by n_in matrix

The initial conditions of state variables are defined by means of the vector x_o.

4.90.8 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

When the sample time instant takes place the output signal is calculated as follows:

$$FOR(i=1;i\le n\_out;i++)$$

$$s\_out.signal[i]=\sum_{j=1}^{n\_states}C[i,j]\cdot x[j]+\sum_{j=1}^{n\_in}D[i,j]\cdot s\_in.signal[j]$$

The states variables are computed by:

$$FOR(i=1;i\le n\_states;i++)$$

$$x_t[i]=\sum_{j=1}^{n\_states}A[i,j]\cdot x_{t-\delta t}[j]+\sum_{j=1}^{n\_in}B[i,j]\cdot s\_in.signal[j]$$

where:

xt = the new value of the state variables for the current sample time

xt - δt = the value of the state variables in the previous sample time

4.91 dTransferFunction

4.91.1 Description

This component is inherited from component dSISO. It represents a discrete transfer function described by the following equation:

$$y(z)=\frac{p[1]\cdot z^{n\_num}+p[2]\cdot z^{n\_num-1}+\ldots+p[n+1]}{q[1]\cdot z^{n\_den}+q[2]\cdot z^{n\_den-1}+\ldots+q[n+1]}\cdot u(z)$$

where n_num and n_den are the degree of the numerator and denominator polynomials respectively.

4.91.2 Symbol

dTransferFunction

4.91.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_den	CONST INTEGER	1	Order of denominator polinomial
n_num	CONST INTEGER	0	Order of numerator polinomial

4.91.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.91.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
dt	REAL	0.1	Sample time	s
option	ENUM InitOption	Initial-States	Initialization of discrete component: states or output	
pdata[n_num + 1]	REAL		Coefficients of numerator polinomial in descending power of z: p[1]*z^n+...+p[n+1]	
qdata[n_den + 1]	REAL		Coefficients of denominator polinomial in descending power of z: q[1]*z^n+...+q[n+1]	
x_o[n_den]	REAL		Initial conditions of the states	
y_o	REAL	0	Initial condition for the output	

4.91.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.91.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
j	INTEGER			
sample	BOOLEAN	FALSE		
x[n_den + 1]	REAL		Values of states	

4.91.8 Guidelines

The number of coeffcients of the data vectors pdata and qdata are (n_num+1) and (n_den+1) respectively. The order of the denominator must be greater than or equal to the order of the numerator.

This component represents the method typically used by control engineers representing systems as polynomials in z or s.

4.91.9 Formulation

To sample at time 0 the following WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
   sample = TRUE AFTER dt
END WHEN
```

When the sample time instant takes place the output signal is calculated as follows:

The state variable x[1] is calculated before the output signal by:

$$x[1] = \frac{s\_in.signal[1] = \sum_{j=2}^{n\_den+1} x[j] \cdot qdata[j]}{qdata[1]}$$

$$s\_out.signal[1] = \sum_{j=1}^{n\_num+1} x[n\_den + 2 - j] \cdot p[n\_num + 2 - j]$$

And the remainder of states variables are calculated as:

$$FOR(j = 1; j \le n\_den; j++)$$
$$x[n\_den + 2 - j] = x[n\_den + 1 - j]$$

4.92 dZeroPole

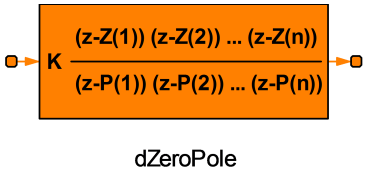
4.92.1 Description

This component is inherited from component dSISO. It implements a discrete system with the specified zeros, poles and gain in terms of the delay operator z as the following equation shows:

$$y(z) = K \cdot \frac{(z - zeros(1)) \cdot (z - zeros(2)) \cdot \ldots \ (z - zeros(n))}{(z - poles(1)) \cdot (z - poles(2)) \cdot \ldots \ (z - poles(n))} \cdot u(z)$$

where zeros represents the zeros vector, poles represents the poles vector and K represents the gain.

4.92.2 Symbol



4.92.3 Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n_poles	CONST INTEGER	1	Number of poles
n_zeros	CONST INTEGER	1	Number of zeros

4.92.4 Ports

NAME	TYPE	PARAMETERS	DIRECTION	DESCRIPTION
s_in	analog_signal	(n = n)	IN	Inlet signal
s_out	analog_signal	(n = n)	OUT	Outlet signal

4.92.5 Data

NAME	TYPE	DEFAULT	DESCRIPTION	UNITS
K	REAL	1	Gain array	
dt	REAL	0.1	Sample time	s
option	ENUM InitOption	Initial-States	Initialization of discrete component: states or output	
poles[n_poles,2]	REAL		Values of the poles	real part, imaginary part
x_o[n_poles]	REAL		Initial conditions of the states	
y_o	REAL	0	Initial condition for the output	
zeros[n_zeros,2]	REAL		Values of the zeros	real part, imaginary part

4.92.6 Closed Parameters

NAME	TYPE	DEFAULT	DESCRIPTION
n	CONST INTEGER	1	Dimension of input and ouput signals

4.92.7 Variables

NAME	TYPE	INITIAL	DESCRIPTION	UNITS
j	INTEGER			
p[n_zeros + 1]	REAL		Coefficients of numerator polinomial in increasing power of z	
q[n_poles + 1]	REAL		Coefficients of denominator polinomial in increasing power of z	
sample	BOOLEAN	FALSE		
x[n_poles + 1]	REAL		Values of the states	

4.92.8 Guidelines

The numerator cannot be of highter order than denominator.

4.92.9 Formulation

To sample at time 0 the following EcosimPro WHEN block is used:

```
WHEN(TIME == 0) THEN
   sample = TRUE AFTER 0.
END WHEN
```

The following WHEN block is used to compute when the the sample time instant takes place.

```
WHEN (sample == TRUE) THEN
   sample = FALSE AFTER 0.
```

```
   sample = TRUE AFTER dt
END WHEN
```

The user can select the way to initialize the component with the data option, either initializing the output signal (InitialOutput option) or initializing the states variables (InitialStates option).

When the sample time instant takes place the output signal is calculated as follows:

The state variable x[1] is calculated before the output signal by:

$$x[1] = \frac{s\_in.signal[1] - \sum_{j=2}^{n\_poles+1} x[j] \cdot q[n\_poles + 2 - j]}{q[n\_poles + 1]}$$

$$s\_out.signal[1] = K \cdot \left(\sum_{j=1}^{n\_zeros+1} x[n\_poles + 2 - j] \cdot p[j] \right)$$

And the remainder of states variables are calculated as:

$$FOR(j = 1; j \leq n\_poles; j++)$$
$$x[n\_poles + 2 - j] = x[n\_poles + 1 - j]$$

This page intentionally left blank.

5. Appendix A: Theory of PID Controllers

5.1 Introduction

Linear PID controllers have two main advantages which make them the most commonly used: their robustness and the intuitive relationships between their parameters and the system response.

The basic algorithm of the PID controllers are the following:

$$u(t)=K_P\,(e(t)+\frac{1}{T_I}\int e(t)dt+T_D\,\frac{de(t)}{dt}) \quad = P + I + D$$

where:

u(t) = the control signal or the manipulated variable

e(t) = the error signal (r(t) - y(t))

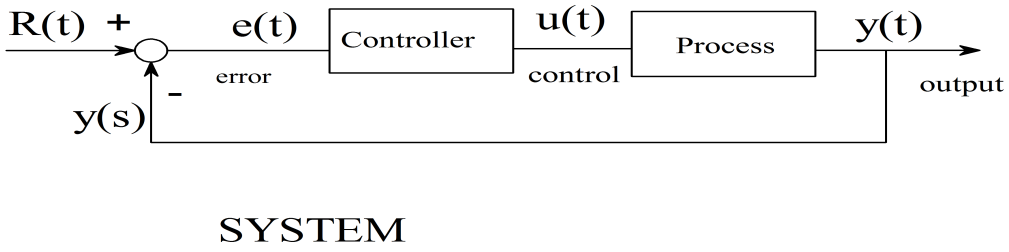


Figure 5-1. Basic Diagram of a Feedback Control System

Definitions:

- Proportional Action (P) is the action produced by a proportional control signal on the deviation of the output signal with regard to the set-point
- Integral action (I) is the action produced by an integral control signal on the time that the output is different from the set-point
- Derivative action (D) is the action produced by a proportional control signal on the speed at which the output signal varies with respect to the set-point
- Gains: proportional (kp); integral (ki = kp / Ti) and derivative (kd = kp Td) are the proportionality constants of the different controller actions

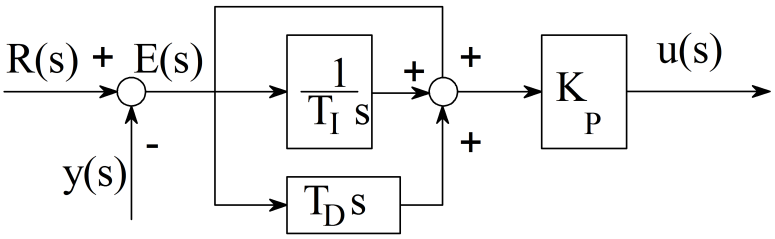
- Proportional range (BP=100/KP) is the change expressed as a % that the process output (measured variable) has to undergo for the output of the proportional block to exceed its minimum value (0%) or its maximum value (100%)
- Integral Time Constant (TI) is the time that must elapse for the integral action to match (to equal or to repeat) the proportional action
- Reset frequency (1/TI) is the number of times that the integral action matches the proportional action in one minute
- Derivative Time Constant (TD) is the time that must elapse for the derivative action to match the proportional action

When the proportional gain is very large, or when the proportional range is very small or when the integral time constant is very small, the controller can become unstable.

Derivative action is not recommendable when a lot of noise is present, as it is amplified.

5.2 Types of PID Structures

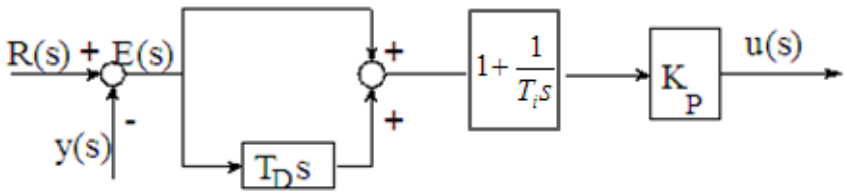
In industrial controllers it is common to find different versions of the PID control algorithm. Any such algorithm can be considered to belong to one of the three groups of PID controllers shown in Figure 5-2 (non-interactive, interactive or parallel).



Non Interactive PID

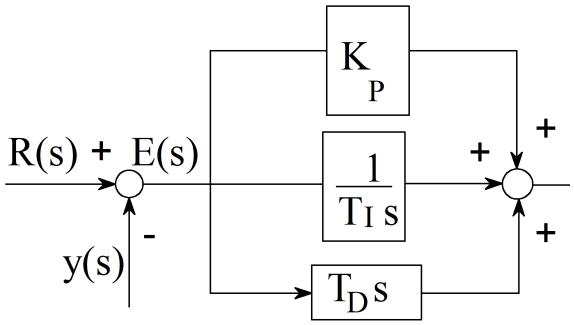
$$u(s) = K_p \left(1 + \frac{1}{T_I s} + T_D s\right) \cdot E(s)$$

Non-Inte4ractive PID



Interactive PID

Interactive PID $$u(s)=K_p\,(1+\frac{1}{T_I s})(1+T_D\,s)\cdot E(s)$$



Parallel PID $$u(s)=(K_P+\frac{1}{T_I s}+T_D\,s)\,E(s)$$

Figure 5-2. PID Control Algorithms

5.3 Filter in the Derivative Action

To limit the output of high frequencies (ie noise), the derivative action filter α is used so that the new filtered derivative action becomes:

$$\frac{u_d(s)}{E(s)}=\frac{K_P\,T_D\,s}{\alpha\,T_D\,s+1}$$

which limits the output of high frequencies to $\frac{K_P}{\alpha}$

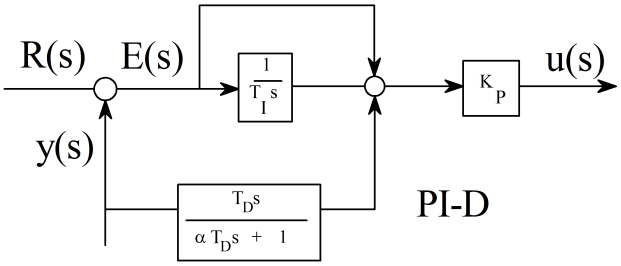
For an interactive PID, the resulting transfer function is:

$$u(s)=K_P\,(1+\frac{1}{T_I\,s})(1+\frac{T_D\,s}{1+\alpha\,T_D\,s})E(s)$$

5.4 Control Structures with Derivative Filter

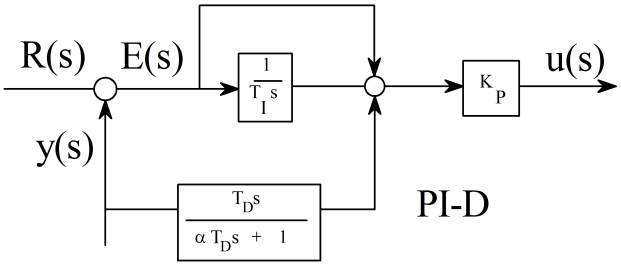
The PID control system allows great flexibility, not only in the control algorithm but also in the handling of the reference signal. Some manufacturers include this last characteristic in their industrial controllers, which allows a choice between different control structures or between different equations.

Figure 5-3 shows the three possible control structures, where the derivative action incorporates the derivative filter. In the PID structure, the three control actions (proportional, integral and derivative) act on the error signal. In the PI-D structure, derivative action only acts on the process output; in this way, overly high control signals with respect to the PID structure are avoided when the set-point changes. In the I-PD structure brusque changes in the control signal due to proportional action are avoided, as only the integral part acts on the error signal.



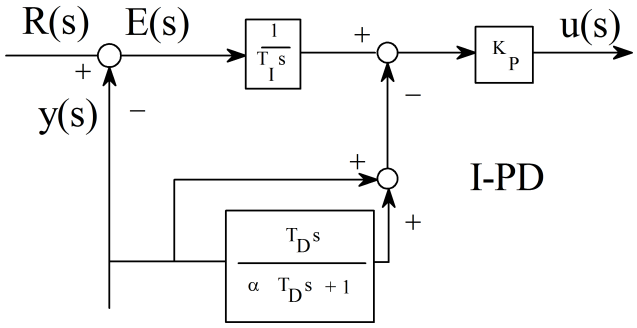
PID Structure

$$u(t)=K_P\left(e(t)+\frac{1}{T_I}\int e(t)\,dt-T_D\,\frac{dy(t)}{dt}\right)$$



PI-D Structure

$$u(t)=K_P\left(e(t)+\frac{1}{T_I}\int e(t)\,dt-T_D\,\frac{dy(t)}{dt}\right)$$



I-PD Structure

$$u(t)=K_P\left(-y(t)+\frac{1}{T_I}\int e(t)\,dt-T_D\frac{dy(t)}{dt}\right)$$

Figure 5-3. Non-Interactive PID Control Structures

There is a general equation which includes the PI-D and I-PD structures for the case of non-interactive control. This equation is:

$$u(t)=K_p\left(\beta\cdot r(t)-y(t)+\frac{1}{T_I}\int e(t)\,dt-T_D\frac{dy(t)}{dt}\right)$$

β is the filter for brusque changes in the set-point

If β = 1 PI-D Controller

If β = 0 I-PD Controller

The adoption of the PI-D type controller has the following effects on the PID:

Longer increase time

Greater elongation

Lengthening of transient

Reduction of the initial value of the control signal

Minimum improvement in evolution

The adoption of the I-PD type controller has the following effects on the PID:

Longer increase time

Less elongation

Same settling time

Apt for systems that are subject to large changes in set-point

5.5 The Windup Problem

Most current controllers incorporate strategies which desaturate the integral part (anti-reset windup). Basically there are three ways of avoiding or reducing saturation as much as possible:

- Conditional integration methods
- Integral followup method
- Input limitation methods
- Integral Followup Method

This method is the most efficient and the only one which we will analyse here. It is based on feeding back to the controller the difference between the control signal which it generates, and the saturated control signal received by the actuator; in this way the difference between both signals is used by the controller to bring the control signal to the same value as the saturated signal. The method is so efficient because it allows a great reduction in the saturation time of the integral part, but it needs to know the physical limits of the actuator at all times, either by direct measurement or by incorporating a model of the actuator within the actual controller; said model may be a simple limiter.

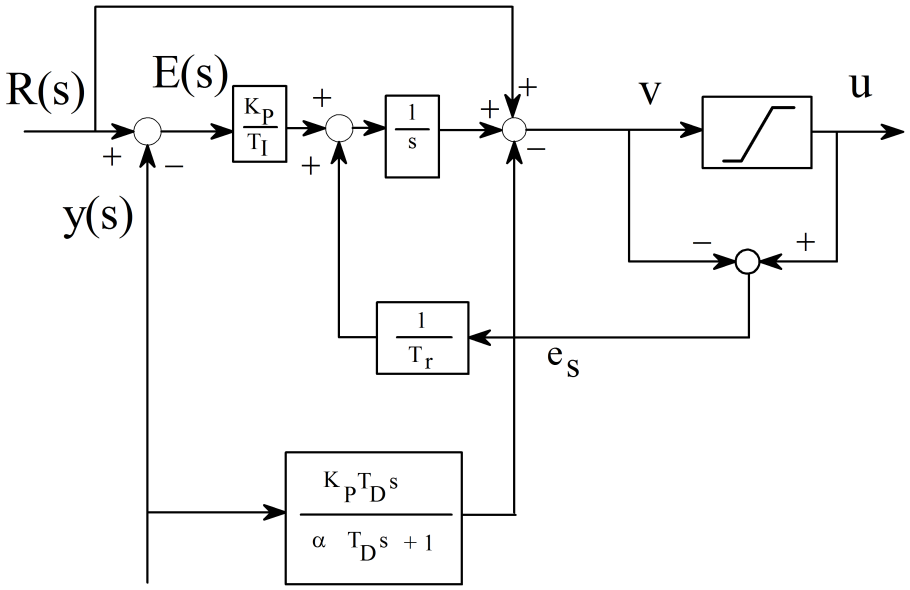


Figure 5-4. Anti-Reset Windup Using the Integral Followup Method

5.6 Description of the Cntrl_pid Component

The CONTROL library includes the Cntrl_pi and Cntrl_pid components to represent PI and PID controllers respectively. In the case of the Cntrl_PID controller there is a non-interactive PID control structure with derivative filter factor α to avoid noise amplification at low frequencies during derivative action, a weight factor β to avoid the effects of brusque changes of the set-point, and a feedback loop to avoid the effects of saturation on the integral part, whose error signal is applied to the output of the integral action or of the global action depending on whether the behaviour at the end position is end_I or PI respectively and in accordance with the following diagram:

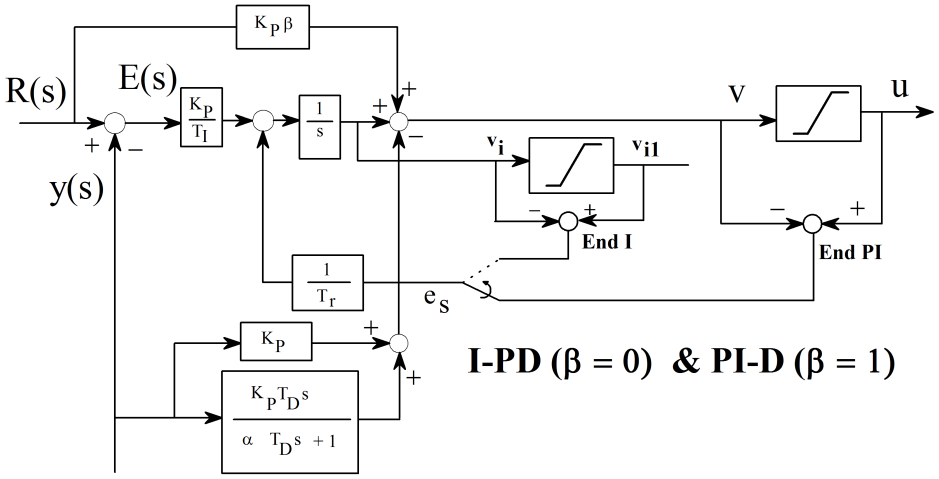


Figure 5-5. Structure of Component Cntrl_pid

This page intentionally left blank.

This page intentionally left blank.

EA Internacional
Magallanes, 3. 28015 Madrid. SPAIN.
E-mail: info@ecosimpro.com
Web: www.ecosimpro.com

EMPRESARIOS AGRUPADOS

Edition 2015