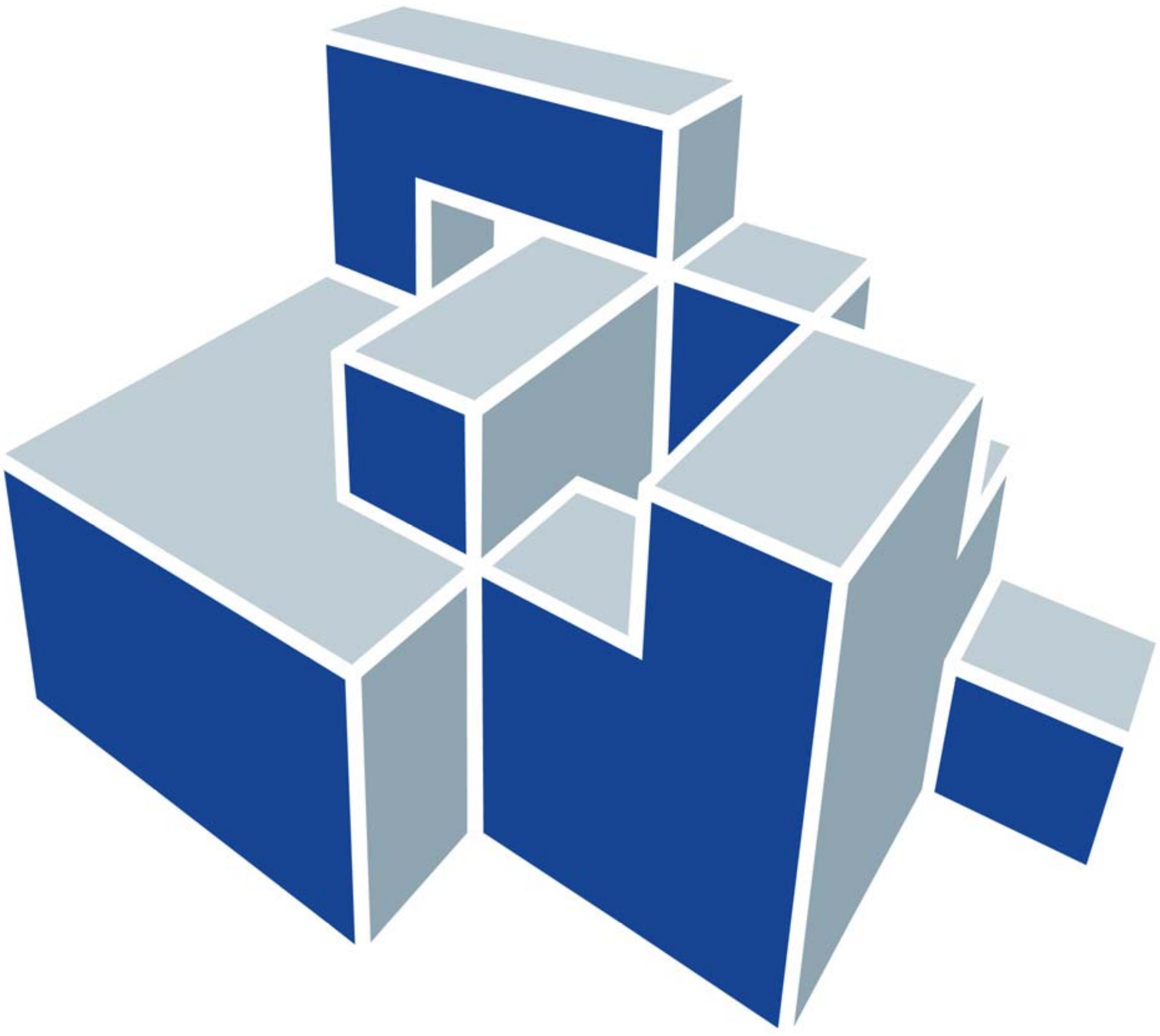
SIEMENS

SIMIT SCE

Basic Library

Reference Manual



SIEMENS

Edition

July 2009

Trademarks

SIMIT® is a registered trademark of Siemens AG.

Other names used in this document can be trademarks, the use of which by third-parties for their own purposes could violate the rights of the owners

Siemens AG
Industrial Solutions and Services
Information Technology Plant Solutions

SIMIT-HB-V7BL-2009-07-en

Contents

Table of Figures

List of Tables

1 PREFACE

1.1 Target group

This manual is intended for anyone who uses the SIMIT simulation system. It describes the structure of the standard library of the basic SIMIT system and the component types contained in the standard library. As the central element of SIMIT, the component types form the basis of the SIMIT models. A good knowledge of how they work is essential in order to create the models and run simulations. This manual provides the necessary information.

It assumes knowledge of the basic SIMIT system and a basic knowledge of the mathematics on which the component types are based in addition to an in-depth knowledge of the use of PCs and the Windows user interface.

1.2 Contents

This manual primarily describes the component types contained in the basic library. An introductory chapter 2 first describes the structure of the basic library and explains the general properties of the component types. This section is needed in order to understand the descriptions of the individual component types contained in the following chapters 3 to 5. We therefore recommend that you read chapter 2 first.

Chapters 3 to 5 may be read independently of one another. Chapter 3 describes the connectors that can be used when modelling with the basic SIMIT system. Chapter 4 contains descriptions of the standard component types in SIMIT, while chapters 5 provides information about the drive component types contained in the basic library.

The descriptions of the individual component types are set out so as to clearly illustrate their primary functions. Special features of the implementation are only explained where needed in order to understand how a component works.

1.3 Symbols

Particularly important information is highlighted in the text as follows:

NOTE

Notes contain important supplementary information about the documentation contents. They also highlight those properties of the system or operator input to which we want to draw particular attention.

CAUTION

This means that the system will not respond as described if the specified precautionary measures are not applied.

WARNING

This means that the system may suffer irreparable damage or that data may be lost if the relevant precautionary measures are not applied.

2 INTRODUCTION

The basic library is part of the “SIMIT Basic” system. It contains elementary functions for modelling plant and machine behaviour with SIMIT. As is normally the case in SIMIT, these functions are provided in the form of component types. This manual describes in detail the individual component types contained in the basic library.

The basic library is divided into four sections:

- connectors,
- standard components, and
- drive components

The component types are stored in three different folders according to this breakdown:

- “CONNECTORS”,
- “STANDARD”, and
- “DRIVES”

2.1 Language

The names of all library components and the associated internal names of connections, parameters, statuses and messages are all in English. This is needed in order to create a unique and interchangeable basic library for both the German and all the foreign-language versions of SIMIT.

2.2 General

Every description of a component type is preceded by its symbol. The symbol of a component type is the central element for modelling on diagrams so, in this manual, it acts as the reference point for the functional description. It contains various elements itself, such as a label or graphic and connections with names. These elements are designed for each component type so that the function of the component and the connections can be intuitively understood when modelling on diagrams.

Other connection designations are used in the description in this manual. These are used as and when the functional description requires. The individual elements of the description of a component type are arranged on the symbol as shown in Figure 2-1, for example.

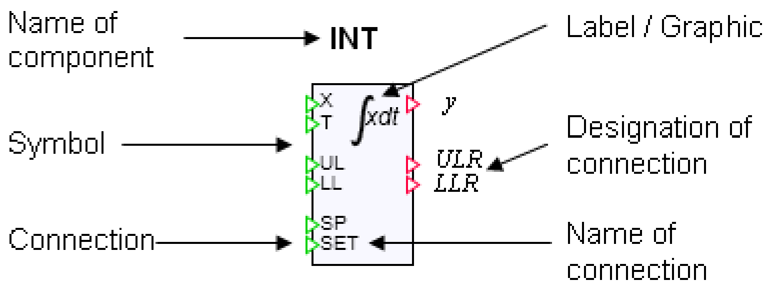


Figure 2-1: Elements of the component type description

A component type can be found in the library under its name. Component types are instantiated as components with their symbol when they are placed on a diagram. Each symbol therefore has a label or graphic that clearly shows the function of the components on diagrams.

Connections are available as inputs or outputs. Inputs are generally arranged on the left and outputs on the right of the component symbol. All inputs and outputs are either binary (logical) or analog inputs or outputs. No complex connection types are used in the components of the basic library. In the explanatory notes, the values of the binary inputs or outputs are designated by zero or one; zero stands for FALSE and one for TRUE.

Inputs and outputs that belong together from the functional viewpoint are arranged opposite one another in the symbol. In the above example, these are input x and output y. Inputs and outputs that belong together from the functional viewpoint are also grouped together and are separated from other groups by spaces. This means that the functional interactions created by interconnecting components can be more easily detected on diagrams. The above sample component illustrates the three following groups:

- Inputs x and T for calculating the integral value at output y
- The limits “UL” and “LL” with the binary feedback b_{UL}, b_{LL}
- Set value “SP” and set command “SET” for setting the integral output y

Connections are given designations in the description when this is needed in order to explain the function. In the above example, the function described by the integral

$$y = \frac{1}{T}\int x \,\mathrm{d}t$$

can be assigned to the connections. Connections are only given names in the symbol if the function of that connection cannot be clearly identified.

In the above example, variables such as the integration time T that are generally fixed as parameters for the simulation are defined as input variables. They can also be interconnected and thus modified by the interaction between functions in the simulation. For example, components from the standard library can be used to set the integration time to a different value if the input values X lie within a specific range (see Figure 2-2).

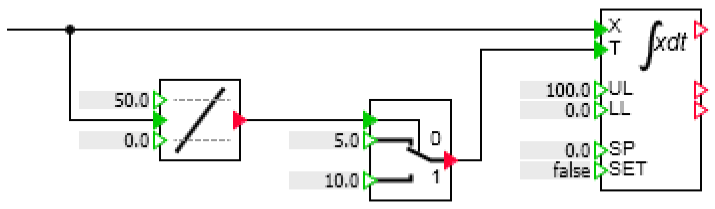


Figure 2-2: Interval-specific changeover of the integration time

Parameters that are only accessible via the property view can also be defined for component types.

2.2.1 Component types with a variable number of inputs

The number of inputs can be varied for the component types “ADD”, “MUL”, “AND”, “OR”, etc. The number is set graphically on the component symbol on diagrams. If the number of inputs can be varied, the cursor changes shape when it is moved over the base or top line of the component symbol on a diagram.

You can set the number of inputs by left-clicking on the base or top line of the symbol, then dragging the line up or down while holding down the left mouse button (see Figure 2-3).

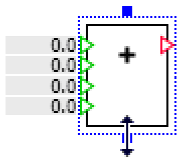


Figure 2-3: Setting the number of inputs on the component symbol

3 CONNECTORS

The “CONNECTORS” folder of the basic library contains connectors for modelling:

- Global connectors
- Input/Output connectors

Other connectors are provided in specific SIMIT modules or libraries. These connectors are located in the “CONNECTORS” folder too.

Global connectors and Input/Output connectors have the following shared characteristics:

The connections of the connector have no type. This means that the connectors assume the connection type of the connected component.

NOTE

A type check, i.e. a check to identify whether the connections connected via connectors are of the same type, is carried out automatically when the simulation is started. If signals with conflicting types are connected, a message will be shown and starting the simulation is cancelled.

The width of the connector symbol on a diagram can be varied and thus matched to the length of the connector name. To set the width of an output or input connector, move the cursor on the left or right-hand edge of the connector. The shape of the cursor then changes. Then hold down the left mouse button and move the edge to the left or right to the desired width.

The connector name and the diagram name, if appropriate, are displayed in the connector symbol on the diagram.

3.1 Global connectors

Global connectors are used just to link models across the boundaries of individual diagrams.

Global connectors may be used as output or as input connectors. The symbol is illustrated in Figure 3-1.

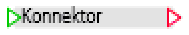


Figure 3-1: Global connector

A link is always established between an output connector and one or more associated input connectors. Links with global connectors simply require a name to be entered for the link. This is the connector name.

NOTE

Please note that global connector names must be unique across the entire simulation project.

The connector name can be entered directly in the symbol. Double click on the connector to open the input box, then press "Return" to confirm your input.

3.2 Input and Output connectors

The components „Input" and „Output" produce the link to signals in the SIMIT gateways. These components can produce a link to signals from any SIMIT gateway.

NOTE

To be consistent with the labeling of input and output siganls in the gateway, the input and output connectors are also named from the viepoint of the controller.

There are input and output connectors („Input" und „Output") as illustrated in Figure 3-3.

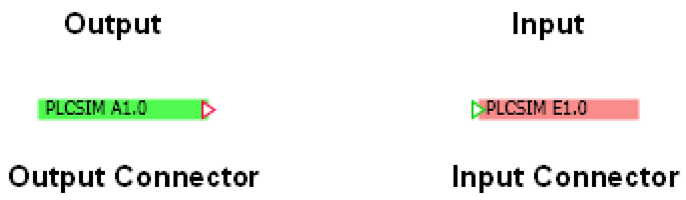
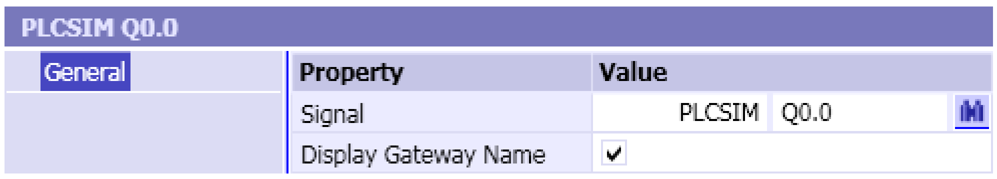


Figure 3-2: Input and Output connectors

The link is established by setting the name of the gateway and the name of the signal in the property view of the connector (Figure 3-3).


PLCSIM Q0.0

General

Property	Value
Signal	PLCSIM Q0.0
Display Gateway Name	✓

Figure 3-3: Parametrization of Input and Output connectors

If you don't want the name of the parametrized gateway to be shown on the diagram, you may deselect the option „Display Gateway Name".

The value of an output connector is set by the gateway. An output connector therefore may be used several times in a project, also on different diagrams. The value of an input connector is set by the simulation project. An input connector therefore must be unique in the whole project.

3.3 Connectors

Two connector components - referred to simply as connectors - are available in the "CONNECTORS" folder of the basic library (Figure 3-4):

- The analog connector component "A" and
- the binary connector component "B".

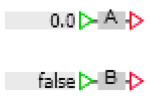


Figure 3-4: Connector components

A connector component transfers the input value to its output without change or delay. They are used, for example, in the following cases:

- Connectors (peripheral connectors or global connectors) cannot be connected directly to each other. As shown in the example in Figure 3-5, connectors can be connected with the aid of connecting components.

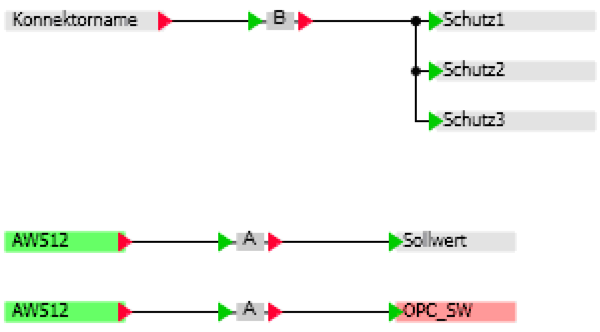
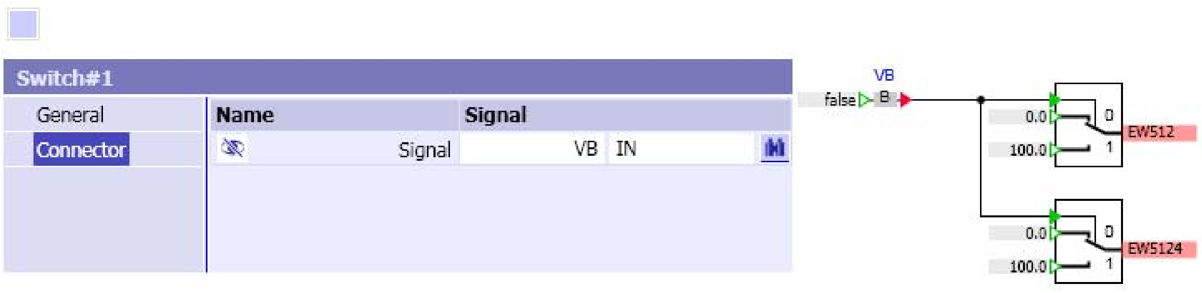


Figure 3-5: Connecting connectors

- An input element, for example a converter, is linked directly to the signal to be set. With the aid of a connector, the input element can influence several components directly. As shown in the example in Figure 3-6, the input signal "IN" of the connector "VB" is linked to the input element.



Name	Signal	
Signal	VB	IN

Figure 3-6: **Multiple connections to an input element**

- Component inputs can be assigned a default value directly. With the aid of a connector, several inputs can be assigned a single default value. As shown in the example in Figure 3-7, a connector is connected to the inputs of the components to be set. The value to be set is then placed on the input of the connector.

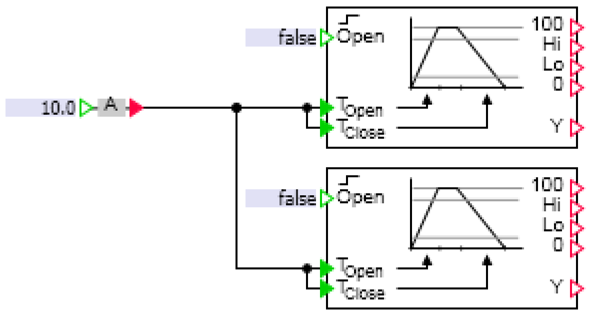


Figure 3-7: Assigning a default value to several inputs

4 STANDARD COMPONENTS

The standard component types in the basic library are contained in the “STANDARD” library folder. The component types are broken down according to function into component types with

- Analog functions
- Binary functions
- Integer functions
- Conversion functions
- Mathematical functions
- Various auxiliary functions.

4.1 Analog functions

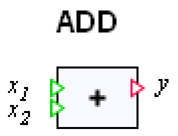
All component types with analog functions are contained in the "AnalogBasic" and "AnalogExtended" subfolders of the standard library. "AnalogBasic" contains the basic analog functions, while "AnalogExtended" holds the extended analog functions.

4.1.1 Basic analog functions

The four basic analog functions of Addition, Subtraction, Multiplication and Division, i.e. the four basic arithmetic operations, are stored as component types in the "AnalogBasic" subfolder of the standard library.

4.1.1.1 ADD – Addition

Symbol



Function

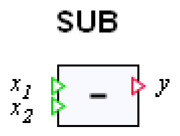
The “ADD” component type maps the sum of the analog values at the n inputs x_1 to x_n onto the output y:

$$y = \sum_{i=1}^{n} x_i = x_1 + x_2 + \ldots + x_n .$$

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

4.1.1.2 SUB - Subtraction

Symbol



Function

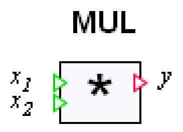
The “SUB” component type maps the difference between the analog values at the inputs x_1 and x_2 onto the output y:

$$y = x_1 - x_2\,.$$

All inputs are set to zero by default.

4.1.1.3 MUL - Multiplication

Symbol



Function

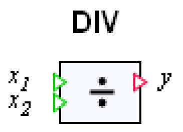
The “MUL” component type maps the product of the analog values at the n inputs x_1 to x_n onto the output y:

$$y = \prod_{i=1}^{n} x_i = x_1 x_2 \;\ldots\; x_n\,.$$

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to one by default.

4.1.1.4 DIV – Division

Symbol



Function

The “DIV” component type maps the quotients of the analog values at the inputs x_1 and x_2 onto the output y:

$$y = \frac{x_1}{x_2}.$$

Input x_1 is set to zero by default, while the divisor input x_2 is set to one by default.

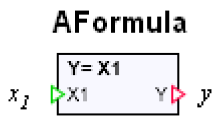
The value of the divisor x_2 must not be zero. When the simulation is run, if the divisor is zero, the error message "DIV: division by zero“ (message category ERROR) is generated and the output y is set to zero.

4.1.2 Extended analog functions

The "AnalogExtended" subfolder of the standard library contains further analog functions in the form of component types.

4.1.2.1 AFormula – analog formula component

Symbol



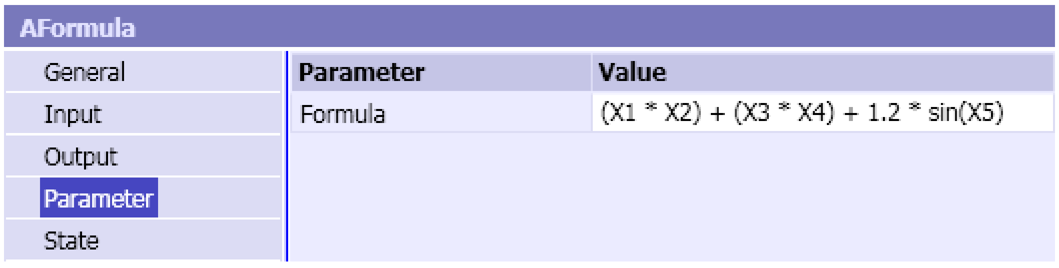
Function

The "AFormula" component allows explicit algebraic functions to be used. This function f calculates a value in relation to the n input values x_i. The function value is assigned to the output y:

$$y = f(x_1, \ldots, x_n).$$

To define the function, enter the desired formula expression in the "Formula“ parameter that calculates the output y in relation to the inputs x_i. First set the required number of inputs and then open the properties dialog for the component in order to enter the formula (see Figure 4-1).

The number n of inputs can be varied between 1 and 32. Only the inputs that are available according to the number currently set may be used in the formula. The formula is displayed at the top of the component symbol. Figure 4-2 shows a component with three inputs by way of example.



Parameter	Value
Formula	(X1 * X2) + (X3 * X4) + 1.2 * sin(X5)

Figure 4-1: Properties dialog for the "AFormula" component type

You can make the component wider to allow longer formulae to be displayed in full. To do this, move the cursor over the right-hand edge of the component, hold down the left mouse button and drag the edge to the desired width.

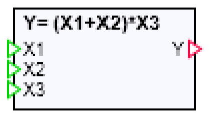


Figure 4-2: "AFormula" component with three inputs

The operators listed in Table 4-1 may be used in formulae.

Operator	**Function**
+	Addition
-	Subtraction
/	Division
*	Multiplication
(	Open bracket
)	Close bracket
Constant	Floating point numbers, also expressed as exponents
Function calls	Standard mathematical functions (see Table 4-2)

Table 4-1: Operators in formulae for the "AFormula" component type

The mathematical functions commonly used in the C/C++ programming language as listed in Table 4-2 are available as standard mathematical functions.

A detailed description of the functions can be found in the documentation for the C/C++ compiler used for SIMIT. The compiler supplied for the SIMIT software supports the standard scope in ANSI C. A description of this standard scope of C can be found on the “Microsoft Developer Network” (MSDN) at “http://msdn.microsoft.com/default.aspx”. Navigate to the relevant pages of the "Visual C++ Libraries Reference".

NOTE

In the formula component there is no check to determine whether all the set inputs are used in the specified formula.

Formula	Function
sqrt(x)	$y = \sqrt{x}$; $x \geq 0$
fabs(x)	$y = \lvert x \rvert$
exp(x)	$y = \mathrm{e}^{x}$
pow(x, y)	$y = \mathrm{x}^{y}$;
log(x)	Natural logarithm: $y = \ln(x)$; $x > 0$
log10(x)	Common logarithm: $y = \lg(x)$; $x > 0$
ceil(x)	Smallest integer greater than or equal to x
floor(x)	Largest integer less than or equal to x
rand()	Random value
sin(x)	$y = \sin(x)$; Angle x in the radian measure
cos(x)	$y = \cos(x)$; Angle x in the radian measure
tan(x)	$y = \tan(x)$; Angle x in the radian measure; $x \neq \pm(2n+1)\pi/2$
asin(x)	$y = \arcsin(x)$; $-\pi/2 \leq y \leq \pi/2$
acos(x)	$y = \arccos(x)$; $0 \leq y \leq \pi$
atan(x)	$y = \arctan(x)$; $-\pi/2 \leq y \leq \pi/2$
atan2(z, x)	$y = \arctan(x/z)$; $-\pi \leq z \leq \pi$
sinh(x)	$y = \sinh(x)$; Angle x in the radian measure
cosh(x)	$y = \cosh(x)$; Angle x in the radian measure
tanh(x)	$y = \tanh(x)$; Angle x in the radian measure

Table 4-2: Mathematical functions in formulae for the "AFormula" component type

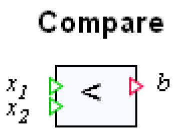
CAUTION

There is also no check in the formula component to determine whether the arguments of the formula have valid values. If a divide by zero occurs during the simulation in the formula calculation, then output y will have the value "**Inf**". If an argument of a formula is not determined during the simulation, as in the case of a division of zero by zero, for example, then output y has the value "**NaN**" (not a number).

These irregular output values will then be propagated in the model in all values that depend on this output. Your simulation will thus enter an undefined state. To avoid this situation, make sure that the inputs of the formula components can only assume values that ensure the validity of the arguments in the formula.

4.1.2.2 Compare functions

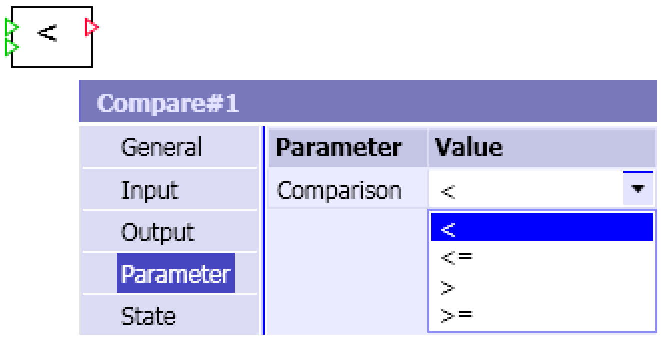
Symbol



Function

The "Compare" component compares analog inputs x_1 and x_2. Binary output b is set to one if the compare expression is true, otherwise b is set to zero.

The type of comparison is determined with the "Comparison" parameter. To set this comparison parameter, open the properties dialog for the component (see Figure 4-3).



Parameter	Value
Comparison	<

Figure 4-3: Properties dialog of the "Compare" component type

The following comparisons may be set:

- "Less than" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad x_1 < x_2 \\ 0, & \text{else} \end{cases}$
- "Less than or equal to" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad x_1 \le x_2 \\ 0, & \text{else} \end{cases}$
- "Greater than" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad x_1 > x_2 \\ 0, & \text{else} \end{cases}$ and
- "Greater than or equal to" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad x_1 \ge x_2 \\ 0, & \text{else} \end{cases}$

The selected comparison operator is displayed in the component symbol (see Figure 4-4).

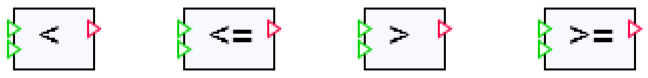


Figure 4-4: Representation of the comparison operator in the symbol

The width of the symbol can be changed in order to be able to offset the “less than or equal to” and “greater than or equal to” comparison operators slight from the right-hand edge of the symbol. To change the width, move the cursor onto the right-hand edge of the symbol. The shape of the cursor changes. Hold down the left mouse button and drag the edge with the mouse to the required position.

In SIMIT, analog variables are mapped onto variables of the type “double”, so it is not meaningful to compare them directly for equality or inequality. Equality of double variables is only possible within the accuracy defined by the computer (machine accuracy). Equality can be checked with the model illustrated in Figure 4-5, for example.

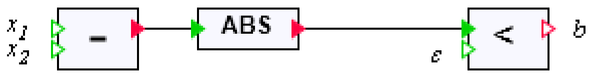


Figure 4-5: Checking for equality with the comparison component

Alternatively, a functionally identical model may be applied, using the “Aformula” formula block, for example (see Figure 4-6).

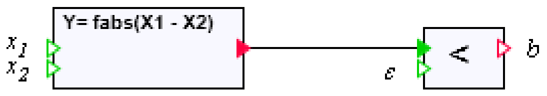


Figure 4-6: Checking for equality with the "AFormula" formula component

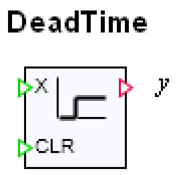
The difference between the two variables x_1 and x_2 is calculated. The amount of this difference is then compared against a definable positive limit ε:

$$b = \begin{cases} 1, & \text{if } |x_1 - x_2| < \varepsilon \\ 0, & \text{else} \end{cases}.$$

To check for inequality, only the “greater than” comparison should be used in the comparison component in both of the illustrated models.

4.1.2.3 DeadTime – Dead time element

Symbol



Function

The "DeadTime" component type makes a dead time element available. The analog value on input x is passed to output y with an adjustable delay.

A step change in the input value x from zero to one thus gives a curve for the output value y as shown in Figure 4-10.

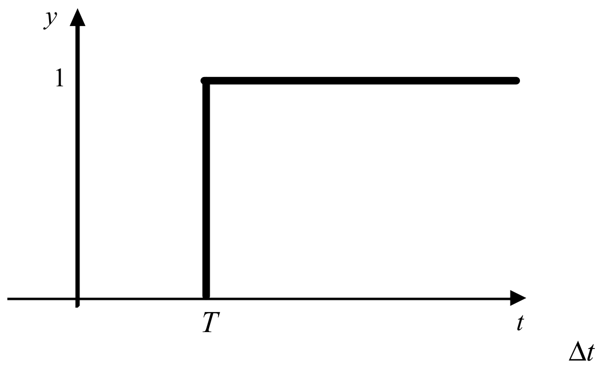


Figure 4-7: Step response of dead time element

The delay time T is adjustable as a "Delay_Cycles" parameter of the component type in whole multiples n of the cycle time Δt : $T = n\Delta t$. It is set to ten by default and may be set to up to 128. The default cycle time of 100 ms therefore produces a delay of one second.

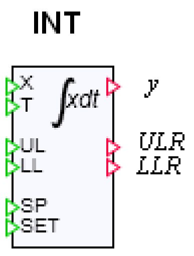
CAUTION

If you change the cycle time Δt in the project properties, you also need to change the selected dead times accordingly.

In the component, the input values are saved and delayed and sent to the output according to the selected dead time. The memory is set up for n values according to the configured number of delay cycles. All storage locations are initialised to zero. The binary input "CLR" can be used to set the output value y and all storage locations n to zero.

4.1.2.4 INT – Integration

Symbol



Function

The “INT” component type forms the integral via the time-specific analog input signal x using

$$y = \frac{1}{T}\int x\,\mathrm{d}t\,.$$

The integral value is assigned to the analog output y.

For a step-shaped input signal of amplitude one this gives a linear ascending output signal as shown in Figure 4-8.

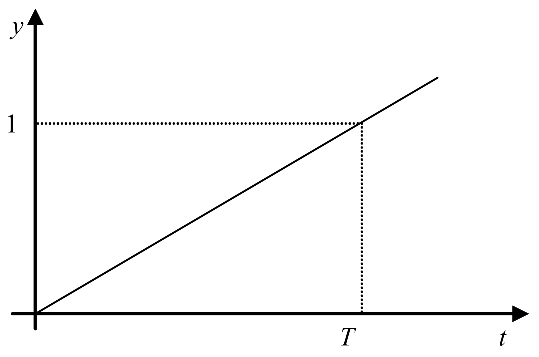


Figure 4-8: Step response of the integration function

The integral value y is limited to an interval defined by the two limit values “UL” (upper limit) and “LL” (lower limit):

$$LL \le y \le UL \,.$$

The binary outputs ULR and LLR indicate when the integral value has reached the lower or upper limit:

$$ULR = \begin{cases} 1, & \text{if} \quad y \ge UL \\ 0, & \text{else} \end{cases} \quad \text{and} \quad LLR = \begin{cases} 1, & \text{if} \quad y \le LL \\ 0, & \text{else} \end{cases}.$$

The lower limit must be less than the upper limit. If this condition is violated, the error message “INT: limits do not match“ (message category ERROR) is generated and output y is set to zero.

The time constant T for the integration must have a positive value. If T is not positive, the error message “INT: zero or negative time constant“ (message category ERROR) is generated and output y is set to zero.

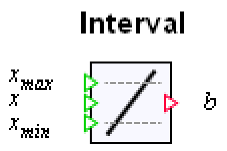
The lower limit is set to zero and the upper limit is set to one hundred by default. The time constant T is set to 1 sec by default. All other inputs are set to zero by default.

The component type only has one parameter - “Initial_Value”. The integration value y is set to this value when the simulation is initialised (started). The parameter is set to zero by default.

The binary input “SET” may be used to set the integration value y to the value at the input “SP”: if “SET” is set to one, then the integration value y is set to equal the value at the input “SP”. Again, the integration value is limited by “LL” and ”UL”.

4.1.2.5 Interval – Interval check

Symbol



Function

The "Interval" component type checks whether an input value x is within the closed interval [x_{min}, x_{max}]. If the input value falls within the set interval, then the binary output is set to one, otherwise it is zero:

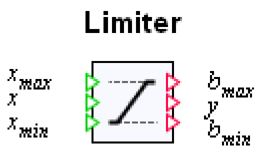
$$b = \begin{cases} 1 & \text{for } x_{\min} \le x \le x_{\max}, \\ 0 & \text{else} \end{cases}.$$

An interval from zero to one hundred is set, i.e. x_{min} is preset to zero and x_{max} is preset to one hundred.

The upper interval limit must not be less than the lower interval limit. When the simulation is run, if the lower interval limit becomes less than the upper limit, then the error message "Limiter: limits do not match“ (message category ERROR) is generated and output b is set to zero (FALSE).

4.1.2.6 Limiter

Symbol



Function

The "Limiter" component type maps an input value limited to the range from x_{min} to x_{max} onto the output y:

$$y = \begin{cases} x_{\max} & \text{for } x \ge x_{\max} \\ x & \text{for } x_{\min} < x < x_{\max} \\ x_{\min} & \text{for } x \le x_{\min} \end{cases}$$

The binary outputs b_{min} and b_{max} are set to one when the limiter takes effect, i.e.

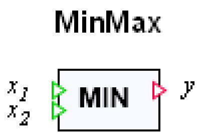
$$b_{\max} = \begin{cases} 1, & \text{if } x \ge x_{\max} \\ 0, & \text{else} \end{cases} \quad \text{and} \quad b_{\min} = \begin{cases} 1, & \text{if } x \le x_{\min} \\ 0, & \text{else} \end{cases}.$$

The limit x_{min} is set to zero and x_{max} is set to one hundred by default.

The upper limit must not be less than the lower limit. When the simulation is run, if the upper limit becomes less than the lower limit, then the error message "Limiter: limits do not match“ (message category ERROR) is generated and output y is set to zero.

4.1.2.7 MinMax – minimum and maximum value selection

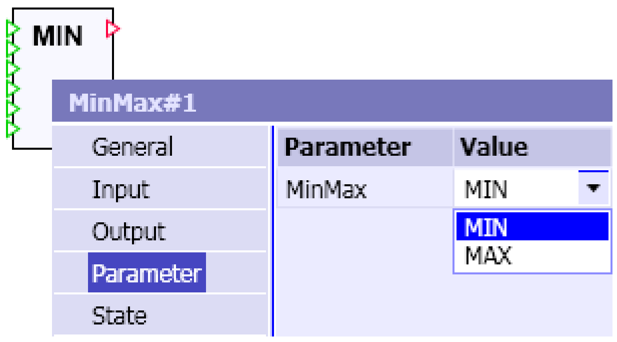
Symbol



Function

The “MinMax” component type maps the minimum or maximum of the n inputs x_1 to x_n onto the output y: The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

The type of mapping is determined with the "MinMax" parameter. To set, open the component properties dialog and set the required value (see Figure 4-9).



MinMax#1

Parameter	Value
MinMax	MIN

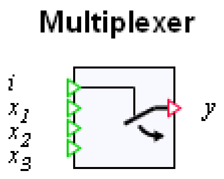
Figure 4-9: Parameter setting for the "MinMax" component type

The "MIN" or "MAX" mapping set for a component is displayed in the component symbol as shown in Figure 4-10.

Figure 4-10: Selection in the symbol for the "MinMax" component type

4.1.2.8 Multiplexer

Symbol



Function

The "Multiplexer" component type switches one of the n different inputs x_i in relation to the value at the selection input i onto the output y:

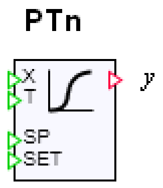
$$y = x_i \quad \text{for } 1 \le i \le n \, .$$

The number n of inputs x_i is variable and can be set to any value between 3 and 32. All inputs are set to zero by default.

The value at the selection input i is limited internally to 1 to n, i.e. for values less than one, i is set to one, while for values greater than n, i is set to the value n. In the default setting, the first input x_i is therefore enabled at the output y.

4.1.2.9 PTn – nth order delay

Symbol



Function

The "PTn" component type provides an n-th order delay.

With a first order delay, the function value y at the output follows the value x at the input with a delay equal to the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \frac{1}{T}(x - y).$$

A step change in the input value x from zero to one thus gives an exponential curve for the output value y (step response) as per

$$y = 1 - e^{-t/T}$$

as illustrated in Figure 4-11.

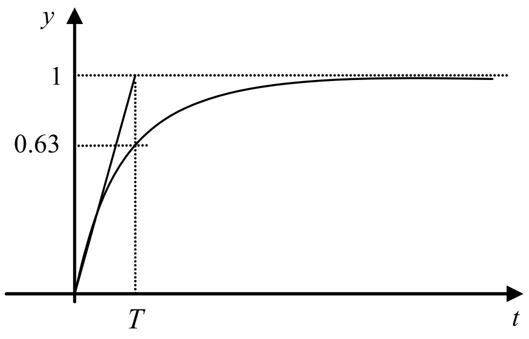


Figure 4-11: Step response for the first order delay function

For higher-order delays, the output value y is obtained by concatenating first-order delays:

$$\frac{\mathrm{d}z_i}{\mathrm{d}t} = \frac{1}{T}(z_{i-1} - z_i),\ i = 1, \dots, n;$$

$$y = z_n .$$

Where $z_i, i = 1,\dots,n,$ are the n states of the n-th order delay. To illustrate, an n-th order delay corresponds to n first order delays connected in series.

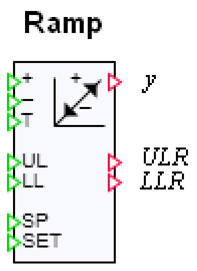
The order n of the delay can be set as a parameter of the component type. It is set to one by default and may be set to up to 32. The "Initial_Value" parameter set to zero is used to initialise the states of the delay function.

To prevent the component behaving in an unstable manner for too short delay times, the delay time constant T is limited to values that are greater than or equal to the cycle time of the component. If the values at input T are less than the cycle time, then the error message "PTn: delay time below cycle time“ (message category ERROR) is generated. The delay time is set to 1 sec by default. All other inputs of the component are set to zero by default.

The binary input “SET” may be used to set the output value y and the status values $z_i, i = 1,...,n,$ to the value at the input “SP”: if “SET” is set to one, then these values are set to equal the value at the input “SP”.

4.1.2.10 Ramp function

Symbol



Function

The "Ramp" component type increments or decrements its function value y in every time step of the simulation by the value

$$\Delta y = \frac{1}{UL - LL} \frac{\Delta t}{T},$$

where Δt is the simulation time step width. The ramp value y is incremented by Δy when the "+" (UP) input is set to one. If the "-" (DOWN) input is set to one, the ramp value y is decremented by Δy. If both the "+" and the "-" input are set to one, then the ramp value y does not change.

The ramp value is limited to an interval defined by the two limit values “UL” (upper limit) and “LL” (lower limit):

$$LL \le y \le UL .$$

The binary outputs ULR and LLR indicate when the ramp value has reached the lower or upper limit:

$$ULR = \begin{cases} 1, & \text{if} \quad y \ge UL \\ 0, & \text{else} \end{cases} \qquad \text{and} \qquad LLR = \begin{cases} 1, & \text{if} \quad y \le LL \\ 0, & \text{else} \end{cases} .$$

The lower limit must be less than the upper limit. If this condition is violated, the error message “RAMP: limits do not match“ (message category ERROR) is generated and output y is set to zero.
The time constant T must have a positive value. If T is not positive, the error message “INT: zero or negative time constant“ (message category ERROR) is generated and output y is set to zero.
The lower limit is set to zero and the upper limit is set to one hundred by default. The time constant is set to 10 sec by default. All other inputs are set to zero by default.

The component type only has one parameter - “Initial_Value”. The ramp function value y is set to this value when the simulation is initialised (started). "Initial_Value" is set to zero by default.

The binary input "SET" may be used to set the ramp value y to the value at the input SP: if SET is set to one, then the ramp value y is set to equal the value at the input SP. Again, the ramp value is limited by “LL” and “UL”.

The component control window can be used to manually set the ramp function value during a simulation. First open the control window (see Figure 4-12).
Use the "MANUAL" button to switch the ramp function to manual mode. The colour of the border around the button changes to red. You can now press the "-" or "+" button to reduce or increase the ramp function value. The current function value is displayed both as a decimal value and as a percentage (see Figure 4-13).

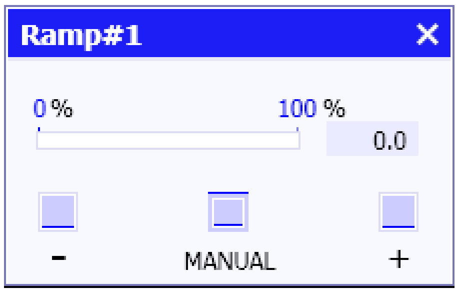


Figure 4-12: Control window for the “Ramp” component type

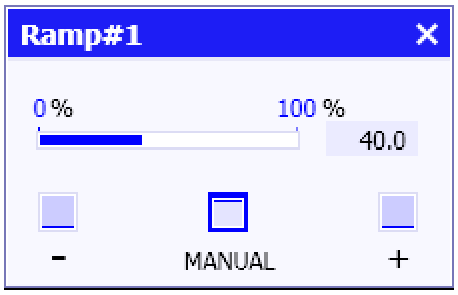
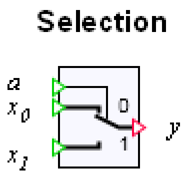


Figure 4-13: Control window for the “Ramp” component in manual mode

4.1.2.11 Selection – analog switch

Symbol



Function

The "Selection" component type switches one of the two inputs x_0 or x_1 through to output y in relation to the value of the binary input a.

If the selection input $a = 0$, then input x_0 is switched through, or if the selection input $a = 1$ then input x_1 is switched through:

$$y = \begin{cases} x_0, & \text{if } a = 0 \\ x_1, & \text{if } a = 1 \end{cases}.$$

All inputs are set to zero by default.

4.2 Integer functions

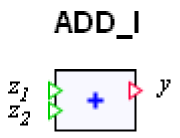
All component types with integer functions are contained in the "IntegerBasic" and "IntegerExtended" subfolders of the standard library. "IntegerBasic" contains the basic integer functions, while "IntegerExtended" holds the extended integer functions.

4.2.1 Basic integer functions

The four basic integer functions of Addition, Subtraction, Multiplication and Division, i.e. the four basic arithmetic operations, are stored as component types in the "IntegerBasic" subfolder of the standard library.

4.2.1.1 ADD_I – Addition

Symbol



Function

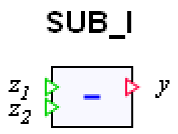
The “ADD_I” component type maps the sum of the analog values at the n inputs x_1 to x_n onto the output y:

$$y = \sum_{i=1}^{n} x_i = x_1 + x_2 + \ ... \ + x_n \, .$$

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

4.2.1.2 SUB_I – Subtraction

Symbol



Function

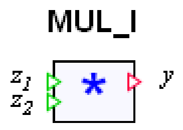
The “SUB_I” component type maps the difference between the integer values at the inputs x_1 and x_2 onto the output y:

$$y = x_1 - x_2 \, .$$

All inputs are set to zero by default.

4.2.1.3 MUL_I – Multiplication

Symbol



Function

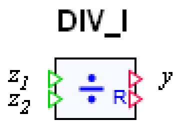
The “MUL_I” component type maps the product of the integer values at the n inputs x_1 to x_n onto the output y:

$$y = \prod_{i=1}^{n} x_i = x_1 x_2 \; \dots \; x_n \, .$$

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to one by default.

4.2.1.4 DIV_I – Integer division

Symbol



Function

The „DIV_I“ component maps the quotient of the integer values at the inputs z_1 and z_2 onto the output y as an integer division with remainder R:

$$z_1 = y \cdot z_2 + R$$

The divident at input z_1 is set to zero by default, the divisor input z_2 is set to one by default.

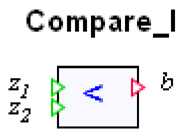
The value of the divisor x_2 must not be zero. When the simulation is run, if the divisor is zero, the error message "DIV: division by zero“ (message category ERROR) is generated and the output y is set to zero.

4.2.2 Extended integer functions

The "IntegerExtended" subfolder of the standard library contains further integer functions in the form of component types.

4.2.2.1 Compare_I – Compare function

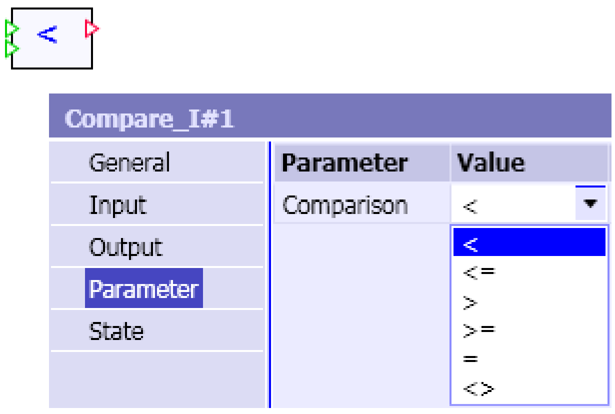
Symbol



Function

The "Compare_I" component compares integer inputs x_1 and x_2. Binary output b is set to one if the compare expression is true, otherwise b is set to zero.

The type of comparison is determined with the "Comparison" parameter. To set this comparison parameter, open the propertiy view (see Figure 4-14).



Compare_I#1

General
Input
Output
Parameter
State

Parameter	Value
Comparison	<

Figure 4-14: Properties dialog of the "Compare_I" component type

The following comparisons may be set:

- "Less than" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad z_1 < z_2 \\ 0, & \text{else} \end{cases}$
- "Less than or equal to" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad z_1 \le z_2 \\ 0, & \text{else} \end{cases}$
- "Greater than" comparison, i.e. $b = \begin{cases} 1, & \text{if} \; z_1 > z_2 \\ 0, & \text{else} \end{cases}$ and
- "Greater than or equal to" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad z_1 \ge z_2 \\ 0, & \text{else} \end{cases}$
- „Equal to" comparison, i.e. $b = \begin{cases} 1, & \text{if} \quad z_1 = z_2 \\ 0, & \text{else} \end{cases}$.
- „Not equal" comparison, i.e. $b = \begin{cases} 0, & \text{if} \quad z_1 \ne z_2 \\ 1, & \text{else} \end{cases}$.

The selected comparison operator is displayed in the component symbol (see Figure 4-15).

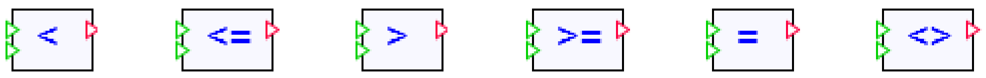
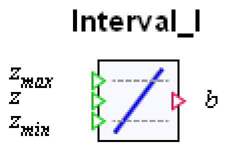


Figure 4-15: Representation of the comparison operator in the symbol

The width of the symbol can be changed in order to be able to offset the “less than or equal to” and “greater than or equal to” comparison operators slight from the right-hand edge of the symbol. To change the width, move the cursor onto the right-hand edge of the symbol. The shape of the cursor changes. Hold down the left mouse button and drag the edge with the mouse to the required position.

4.2.2.2 Interval_I – Interval check

Symbol



Function

The "Interval" component type checks whether an input value z is within the closed interval [z_{min}, z_{max}]. If the input value falls within the set interval, then the binary output is set to one, otherwise it is zero:

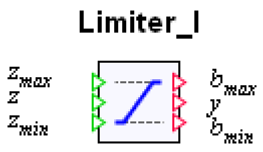
$$b = \begin{cases} 1 & \text{for } z_{\min} \le z \le z_{\max}, \\ 0 & \text{else} \end{cases}.$$

An interval from zero to one hundred is set, i.e. z_{min} is preset to zero and z_{max} is preset to one hundred.

The upper interval limit must not be less than the lower interval limit. When the simulation is run, if the lower interval limit becomes less than the upper limit, then the error message "Limiter: limits do not match“ (message category ERROR) is generated and output b is set to zero (FALSE).

4.2.2.3 Limiter_I

Symbol



Function

The "Limiter" component type maps an input value limited to the range from z_{min} to z_{max} onto the output y:

$$y = \begin{cases} z_{\max} & \text{for } z \ge z_{\max} \\ z & \text{for } z_{\min} < z < z_{\max} \\ z_{\min} & \text{for } z \le z_{\min} \end{cases}$$

The binary outputs b_{min} and b_{max} are set to one when the limiter takes effect, i.e.

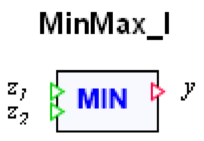
$$b_{\max} = \begin{cases} 1, & \text{if } z \geq z_{\max} \\ 0, & \text{else} \end{cases} \quad \text{and} \quad b_{\min} = \begin{cases} 1, & \text{if } z \leq z_{\min} \\ 0, & \text{else} \end{cases}.$$

The limit z_{min} is set to zero and z_{max} is set to one hundred by default.

The upper limit must not be less than the lower limit. When the simulation is run, if the upper limit becomes less than the lower limit, then the error message "Limiter: limits do not match“ (message category ERROR) is generated and output y is set to zero.

4.2.2.4 MinMax_I – Minimal- und Maximalwertauswahl

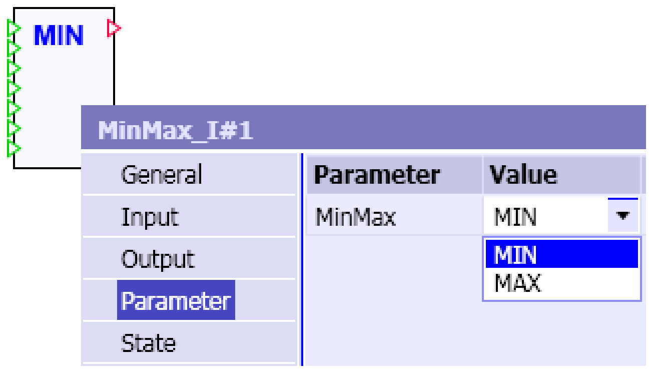
Symbol



Function

The “MinMax” component type maps the minimum or maximum of the n inputs z_1 to z_n onto the output y: The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

The type of mapping is determined with the "MinMax" parameter. To set, open the component properties dialog and set the required value (see Figure 4-16).



Parameter	Value
MinMax	MIN

Figure 4-16: Parameter setting for the "MinMax_I" component type

The "MIN" or "MAX" mapping set for a component is displayed in the component symbol as shown in Figure 4-17.

Figure 4-17: Selection in the symbol for the "MinMax" component type

4.3 Mathematical functions

The “Math” subfolder of the standard library contains the most commonly used mathematical functions in the form of component types: absolute value generation (ABS), square root extraction (SQRT), natural logarithm (LN) and the exponential function (EXP), plus the trigonometric functions sine (SIN), cosine (COS) and tangent (TAN).

4.3.1 ABS – absolute value

Symbol

Function

The “ABS” component maps the absolute value (amount) of input x onto output y:

$$y = |x| .$$

4.3.2 SQRT – square root

Symbol

Function

The “SQRT” component maps the square root of input x onto output y:

$$y = \sqrt{x} .$$

The radicand x must not be negative. If the radicand becomes negative when the simulation is run, the error message "SQRT: invalid argument“ (message category ERROR) is generated and output y is set to zero.

4.3.3 EXP – exponential function

Symbol

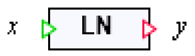
Function

The “EXP” component maps the exponential value of input x onto output y:

$$y = \mathrm{e}^{x} .$$

4.3.4 LN – natural logarithm

Symbol



Function

The “LN” component maps the natural logarithm of input x onto output y:

$y = \ln(x)$.

The argument x must be positive. If the argument becomes less than or equal to zero when the simulation is run, the error message "LN: invalid argument“ (message category ERROR) is generated and output y is set to zero. The argument x is set to one by default.

4.3.5 SIN – sine function

Symbol

Function

The “SIN” component maps the sine value of input x onto output y:

$y = \sin(x)$.

The unit of measure for the argument x (angle) can be set in radians (rad) or degrees (deg) using the "Unit" parameter. The default unit of measure is degrees (deg).

4.3.6 COS – cosine function

Symbol

Function

The “COS” component maps the cosine value of input x onto output y:

$y = \cos(x)$.

The unit of measure for the argument x (angle) can be set in radians (rad) or degrees (deg) using the "Unit" parameter. The default unit of measure is degrees (deg).

4.3.7 TAN – tangent function

Symbol

Function

The “TAN” component maps the tangent value of input x onto output y:

$y = \tan(x)$.

The unit of measure for the argument x (angle) can be set in radians (rad) or degrees (deg) using the "Unit" parameter. The default unit of measure is degrees (deg).

4.4 Binary functions

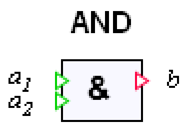
All of the functions for processing binary signals are contained in the "BinaryBasic" and "BinaryExtended" folders. "BinaryBasic" contains the basic binary functions, while "BinaryExtended" holds the extended binary functions.

4.4.1 Basic binary functions

The three basic binary operations of conjunction (AND), disjunction (OR) and negation (NOT), plus equivalence (XNOR) and non-equivalence (XOR), are stored as component types in the "BinaryBasic" folder.

4.4.1.1 AND – conjunction

Symbol



Function

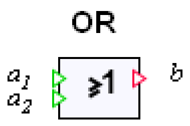
The “AND” component type maps the n binary values at the inputs a_i as a conjunction, i.e. using a logical (boolean) AND function, onto the output b:

$$b = \bigcap_{i=1}^{n} a_i$$

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to one by default.

4.4.1.2 OR – disjunction

Symbol



Function

The “OR” component type maps the n binary values at the inputs a_i as a disjunction, i.e. using a logical (boolean) OR function, onto the output b:

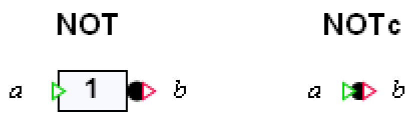
$$b = \bigcup_{i=1}^{n} a_i$$

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

4.4.1.3 NOT, NOTc – negation

Negation is available in two components “NOT” and “NOTc”. These differ only in terms of the symbols used – their functions are totally identical.

Symbol



Function

Output b is equal to the negated input a, i.e.

$$b = \overline{a} \ .$$

As can be seen in the parts of a model in Figure 4-18, when the “NOTc” component is used, the negation can be applied clearly and compactly to the inputs or outputs of components.

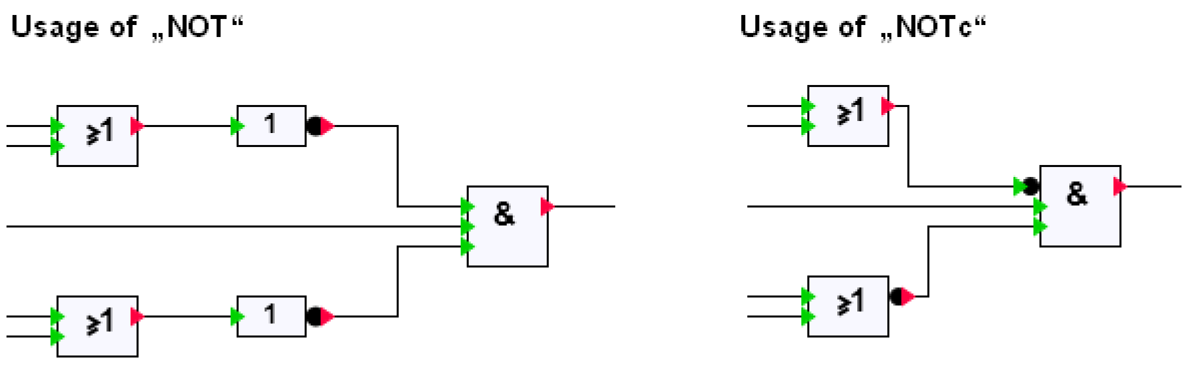
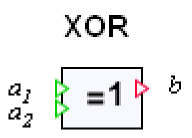


Figure 4-18: Modelling with the component types "NOT" and "NOTc"

4.4.1.4 XOR – non-equivalence

Symbol



Function

The “XOR” component type maps the n binary values at the inputs a_i as a non-equivalence, i.e. using a logical (boolean) exclusive or function, onto the output b: Thus, for two inputs

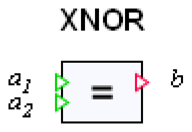
$$b = a_1 \otimes a_2 \ .$$

Output b is one if only one of the inputs a_i is equal to one. In all other cases, the output assumes the value zero.

The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

4.4.1.5 XNOR – equivalence

Symbol



Function

The “XNOR” component type maps the n binary values at the inputs a_i using a binary (Boolean) exclusive nor function (equivalence function) onto the output b. Thus, for two inputs

$$b = \overline{a_1 \otimes a_2}\ .$$

As an inverse XOR function, output b is zero if only one of the inputs a_i is equal to one. In all other cases, the output assumes the value one.

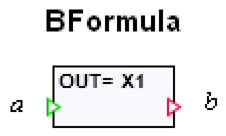
The number of inputs n is variable and can be set to any value between 2 and 32. All inputs are set to zero by default.

4.4.2 Extended binary functions

The "BinaryExtended" subfolder of the standard library contains further binary (logic) functions that go beyond the elementary binary functions in the form of component types.

4.4.2.1 BFormula – binary formula component

Symbol

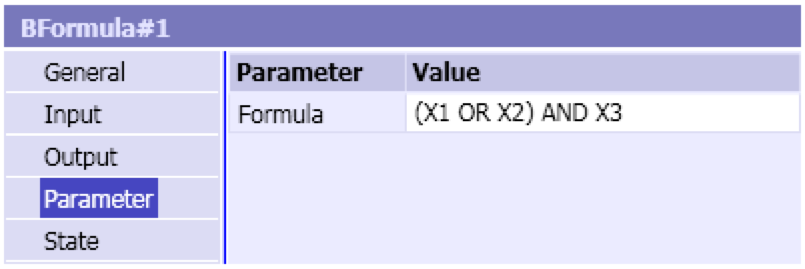


Function

The "BFormula" component allows explicit logic functions to be used. This logic function f calculates a binary value in relation to the n values at the inputs a_i. The function value is assigned to the output b:

$$b = f(a_1, \dots, a_n)$$

To define the function, enter the desired logic formula expression in the "Formula“ parameter that calculates the output value b in relation to the input values a_i. First set the required number of inputs and then open the properties dialog for the component in order to enter the formula (see Figure 4-19).



BFormula#1

Parameter	Value
Formula	(X1 OR X2) AND X3

Figure 4-19: Properties dialog for the "BFormula" component type

The number of inputs can be varied between 1 and 32. Only the inputs that are available according to the number currently set may be used in the formula. As shown in Figure 4-20, the formula is displayed on the diagram in the component symbol.

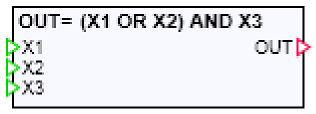


Figure 4-20: "BFormula" component with three inputs

You can make the component wider to allow longer formulae to be displayed in full. To do this, move the cursor over the right-hand edge of the component, hold down the left mouse button and drag the edge to the desired width.

The operators listed in Table 4-3 may be used in formulae.

Operator	Function
AND	Conjunction (AND function)
OR	Disjunction (OR function)
NOT	Negation
(	Open bracket
)	Close bracket

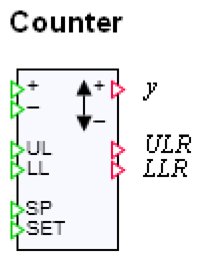
Table 4-3: Permitted operators in formulae for the "BFormula" component type

NOTE

In the formula component there is no check to determine whether all the set inputs are used in the specified formula.

4.4.2.2 Counter – Up and Down counters

Symbol



Function

The "Counter" component is used to count the changes in binary signals. When the binary value at the "**+**" input changes from zero to one, the counter value is incremented at output y. When the binary value at the "**-**" input changes from zero to one, the counter value is decremented at output y.

The values by which the output is decremented or incremented can be set as a parameter ("Decrement" or "Increment"). Both parameters can be modified on-line, i.e. while the simulation is running. You can set any analog value for the decrement and increment. Both parameters are set to one by default.

The counter value is limited to an interval defined by the two limit values "UL" (upper limit) and "LL" (lower limit):

$$LL \le y \le UL\,.$$

The binary outputs ULR and LLR indicate when the counter value has reached the lower or upper limit:

$$ULR = \begin{cases} 1, & \text{if} \quad y \ge UL \\ 0, & \text{else} \end{cases} \qquad \text{and} \qquad LLR = \begin{cases} 1, & \text{if} \quad y \le LL \\ 0, & \text{else} \end{cases}.$$

The lower limit must be less than the upper limit. If this condition is violated, the error message "Counter: limits do not match“ (message category ERROR) is generated and the counter value at output y is set to zero.

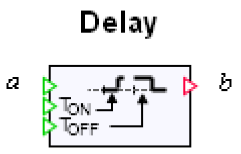
The binary input SET may be used to set the counter value y to the value at the input SP: if SET is set to one, then counter value y is set to equal the value at the input SP. Again, the counter value is limited by LL and UL.

The "Initial_Value" parameter is used to set the counter value when the simulation is initialised (started). "Initial_Value" is set to zero by default.

The limits are set to zero for the lower limit and to one hundred for the upper limit by default. All other inputs are set to zero by default.

4.4.2.3 Delay – On-Off delay

Symbol



Function

With "Delay", the binary signal b at the output is matched to the binary signal a at the input with a delay.

If the signal at the input changes from zero to one, the output is set to one once the On delay time T_{ON} has elapsed. If the input signal is reset to zero before the On delay has elapsed, then the output signal remains unchanged at zero. If the signal at the input changes from one to zero, the output is set to zero once the Off delay time T_{OFF} has elapsed. If the output signal is reset to one before the Off delay has elapsed, then the output signal remains unchanged at one. This interaction is illustrated in Figure 4-21.

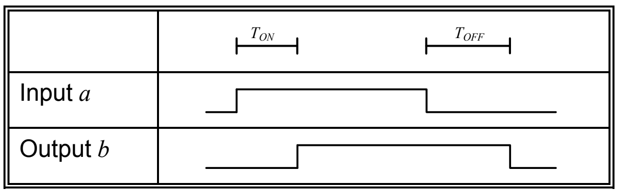
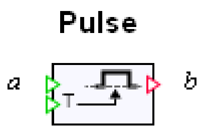


Figure 4-21: Signal curves at the input and output of the "Delay" component type

The delay times must not assume a negative value. If a delay assumes a negative value, then the error message "Delay: on-delay time negative value“ or "Delay: off-delay time negative value“ (message category ERROR) is generated and the corresponding delay is set to zero.

4.4.2.4 Pulse

Symbol



Function

The “Pulse” component type sets the output b when an edge change from zero to one occurs at the input a. The output b is reset once the time T has elapsed. A pulse with pulse width T is thus generated at output b (see Figure 4-22). The pulse width can be set via the signal at analog input T.

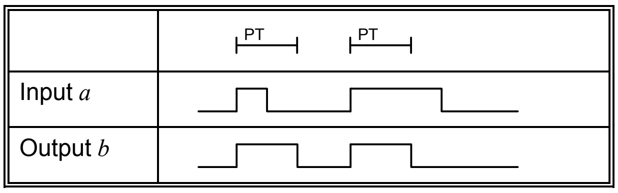


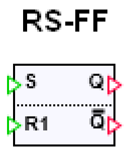
Figure 4-22: Signal curves at the input and output of the "Pulse" component type

The pulse width T must not assume a negative value, otherwise the error message “Pulse: pulse width negative value“ (message category ERROR) is generated and output b is set to zero. For pulse widths that are smaller than the simulation cycle time, the output pulse is set to the duration of one simulation cycle.

If the "Retriggerable" parameter, which can be modified on-line, is set to one, the output pulse is restarted whenever the input signal changes from zero to one. The parameter is set to zero by default.

4.4.2.5 RS_FF – flipflop with preferred state of “Reset”

Symbol



Function

The "RS_FF" component type is the simplest type of flipflop available: an RS flipflop. If the value at the set input S is equal to one, then the output value Q is set to one. If the input value at the reset input $R1$ is set to one, then the output is reset to zero. The reset input is dominant, i.e. if both inputs are set to one, then the output Q is reset to zero. The output $\overline{Q}$ always assumes the inverse value of output Q.

Table 4-4 lists the possible states of the component type.

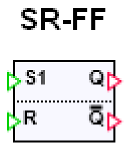
Input S	Input $R1$	Output Q	Output $\overline{Q}$
0	0	unchanged	unchanged
0	1	0	1
1	0	1	0
1	1	0	1

Table 4-4: State table for the “RS_FF” component type

When the simulation starts, the output Q is set to the value of the "InitialState" parameter. "InitialState" is set to zero by default.

4.4.2.6 SR_FF – flipflop with preferred state of “Set”

Symbol



Function

The SR_FF component simulates an SR flipflop. If the value at the set input $S1$ is equal to one, then the output value Q is set to one. If the input value at the reset input R is set to one, then the output is reset to zero. The set input is dominant, i.e. if both inputs are set to one, then the output Q is reset to one. The output $\overline{Q}$ always has the inverse value of output Q.

Table 4-5 lists the possible states of the component type.

When the simulation starts, the output Q is set to the value of the "InitialState" parameter. "InitialState" is set to zero by default.

Input S	Input $R1$	Output Q	Output $\overline{Q}$
0	0	unchanged	unchanged
0	1	0	1
1	0	1	0
1	1	1	0

Table 4-5: State table for the “SR_FF” component type

4.5 Converting values

The “Conv” folder of the standard library contains component types for converting signals:

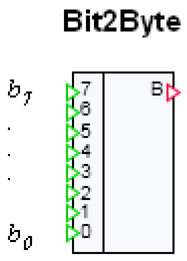
- “Bit2Byte” for converting bits to a byte value
- “Byte2Bit” for converting bytes to bits
- “Byte2Word” for converting bytes to a word
- “Word2Byte” for converting a word into bytes
- “Byte2DWord” for converting bytes to a double word
- “DWord2Byte” for converting a double word into bytes

At present, SIMIT only provides analog and binary signals, so all non-binary signals can only be used in the form of an analog signal. Byte, word and double word values are thus always assigned to analog signals.

Consequently, the inputs or outputs of the components for converting from or to bytes, words and double words can only take the form of analog input or outputs.

4.5.1 Bit2Byte – converting bits into bytes

Symbol



Function

The "Bit2Byte" component type converts the binary input values b_i , $i = 0, ..., 7,$ into a byte value B at the analog output as per

$$B = \sum_{i=0}^{7} b_i \cdot 2^i \ .$$

In the conversion, b_0 thus represents the least significant bit and b_7 the most significant bit.

The individual bits b_i may be set individually while the simulation is running in the component control window. To set a bit, open the component control window. Then select the bits that you wish to set manually by pressing the button to the left of those bits. You can then set and reset each bit by pressing the button on the right, i.e. toggle between zero and one. In the control window in Figure 4-23, bits b_0, b_3 and b_7 are selected for setting, while bit b_3 has already been set.

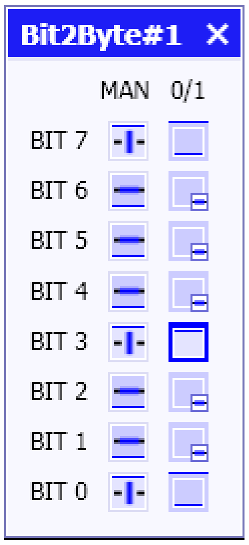
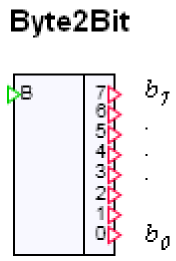


Figure 4-23: Control window for the "Bit2Byte"component type

Unset bits are identified by a light blue border, while set bits have a dark blue border.

4.5.2 Byte2Bit – converting bytes into bits

Symbol



Function

The "Bit2Byte" component type converts a byte value at the analog input B into binary values b_i , $i = 0, ... ,7$, at the outputs. In the conversion, b_0 is assigned the least significant bit and b_7 the most significant bit of the byte value.

The input is limited to the range of values of one byte, i.e. to the range from zero to 255. When the simulation is run, if the input value is not within this range, the message "Byte2Bit: input not a valid byte value“ (message category ERROR) is generated.

The individual output bits b_i may be set individually while the simulation is running in the component control window. To set a bit, open the component control window. Then select the bits that you wish to set manually by pressing the button to the left of those bits. You can then set and reset each bit by pressing the button on the right, i.e. toggle between zero and one. In the control window shown in Figure 4-24, bits b_0, b_4 and b_6 are selected for setting, while bit b_4 has already been set.

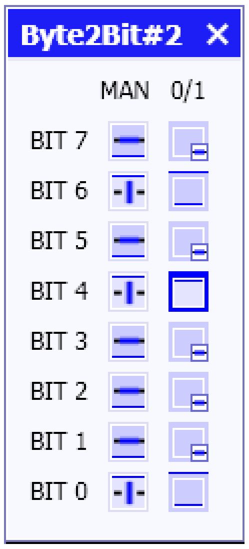
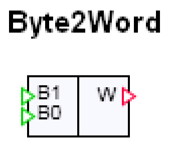


Figure 4-24: Control window for the "Byte2Bit"component type

4.5.3 Byte2Word – converting bytes into words

Symbol



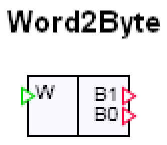
Function

The "Byte2Word" component type combines two byte values B_1, B_0 at the analog inputs to form a word at the analog output W. B_1 is the most significant byte and B_0 is the least significant byte.

The analog values at the input are limited to the range of values of one byte, i.e. to the range from zero to 255. When the simulation is run, if an input value is not within this range, the message "Byte2Word: input not a valid byte value“ (message category ERROR) is generated.

4.5.4 Word2Byte – converting words into bytes

Symbol



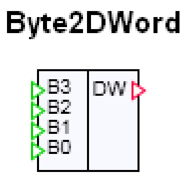
Function

The "Word2Byte" component type breaks down a word at the analog input W into two byte values B_1, B_0 at the analog outputs. B_1 is the most significant byte and B_0 is the least significant byte.

The analog value at the input is limited to the range of values of one word, i.e. to the range from zero to 2^{16}-1. When the simulation is run, if the input value is not within this range, the message "Word2Byte: input not a valid word value“ (ERROR message category) is generated.

4.5.5 Byte2DWord – converting bytes into double words

Symbol



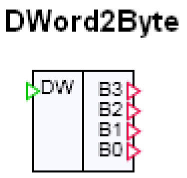
Function

The "Byte2DWord" component type combines four byte values B_i, $i = 0, \dots, 3$, at the analog inputs to form a double word in the order $B_3 - B_2 - B_1 - B_0$ at the analog output DW. $B3$ is thus the most significant byte and B_0 is the least significant byte.

The analog values at the input are limited to the range of values of one byte, i.e. to the range from zero to 255. When the simulation is run, if an input value is not within this range, the message "Byte2DWord: input not a valid byte value" (message category ERROR) is generated.

4.5.6 DWord2Byte – converting double words into bytes

Symbol



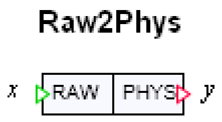
Function

The "DWord2Byte" component type converts a double word at the analog input DW into binary values B_i, $i = 0, \dots, 3$, at the analog outputs. $B3$ is the most significant byte and B_0 is the least significant byte.

The analog value at the input is limited to the range of values of one double word, i.e. to the range from zero to 2^{32}-1 . When the simulation is run, if the input value is not within this range, the message "DWord2Byte: input not a valid double word value" (message category ERROR) is generated.

4.5.7 Raw2Phys – converting from raw to physical

Symbol



Function

The "Raw2Phys" component type converts an analog value x at the input to an analog output value y using the simple linear transformation

$$\frac{y - y_L}{x - x_L} = \frac{y_U - y_L}{x_U - x_L}.$$

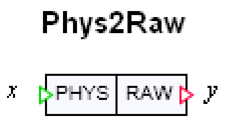
The input value x may be a "raw value" from an automation system, for example. The output value y is then the value transformed within a defined physical range of values.

The transformation intervals are set using the parameters x_L (Lower_Raw_Value), x_U (Upper_Raw_Value), y_L (Lower_Physical_Value) and y_U (Upper_Physical_Value). They may be modified at the simulation run time and have the following default settings:

• Upper_Raw_Value:	27648
• Lower_Raw_Value:	-27648
• Upper_Physical_Value:	100
• Lower_Physical_Value:	0

4.5.8 Phys2Raw – converting from physical to raw

Symbol



Function

The "Phys2Raw" component type converts an analog value x at the input to an analog output value y using the simple linear transformation

$$\frac{x - x_L}{y - y_L} = \frac{x_U - x_L}{y_U - y_L} .$$

The input value x is thus transformed as a measurement of a physical value into the raw value y for an automation system, for example.

The transformation intervals are set using the parameters x_L (Lower_Raw_Value), x_U (Upper_Raw_Value), y_L (Lower_Physical_Value) and y_U (Upper_Physical_Value). They may be modified at the simulation run time and have the following default settings:

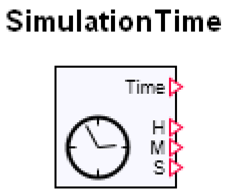
• Upper_Raw_Value:	27648
• Lower_Raw_Value:	-27648
• Upper_Physical_Value:	100
• Lower_Physical_Value:	0

4.6 General components in the “Misc” folder

The Misc subfolder of the standard library contains various components for inputting and outputting analog and binary values.

4.6.1 SimulationTime

Symbol



Function

The "SimulationTime" component type outputs the current simulation time at its four outputs. It is output at the "Time” output as a value in milliseconds. The simulation time is available at the "H", "M", "S" outputs, broken down into hours (H), minutes (M) and seconds (S).

The simulation time is the time that is counted at the simulation cycles while the simulation is running. The simulation time is set to zero when a simulation is initialised or started. The simulation time and real time run synchronously if the simulation is running in real-time mode. In slow-motion simulation mode, the simulation time is slower than real time, while in quick-motion simulation mode, the simulation time is faster than real time. If the simulation is paused, then the simulation time is stopped as well. When the simulation is resumed, then the simulation time starts running as well.

The simulation time is also displayed in the component control window (Figure 4-25).

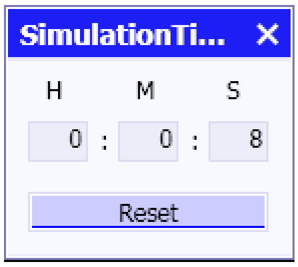


Figure 4-25: Control window for the “SimulationTime” component type

Press the "Reset” button in the component control window to reset the simulation time to zero.

5 DRIVE COMPONENTS

This library contains component types for basic drives. It can thus be used to simulate general drives for valves and pumps. The

- “DriveP1” and “DriveP2” types are designed for simulating pump drives, while
- “DriveV1” to “DriveV4” types are intended for simulating valve drives.

5.1 Valve drives

There are four component types available ("DriveV1", "DriveV2", "DriveV3" and "DriveV4") that may be used to simulate valve drives. The outputs and some of the inputs common to all four "DriveV1" to "DriveV4" component types are illustrated in Figure 5-1.

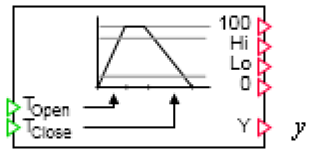


Figure 5-1: Common connections of the component types for valve drives

The two analog inputs T_{Open} and T_{Close} indicate the opening and closing times of the drive, i.e. the time that the drive takes to open or close fully. When the simulation is run, if one of the two input values is negative, the message "DriveVx: closing or opening time invalid value“ (message category ERROR) is generated.

The current position value of the drive is output at analog output y as a percentage, i.e. in the range from zero to one hundred:

$$0 \le y \le 100 .$$

The value zero corresponds to the fully closed valve, while the value one hundred represents the fully open valve.

The four binary outputs “100”, “Hi”, “Lo” and “0” may be interpreted as limit switches for the drive:

- “0” and “100” as limit switches for the valve fully closed or opened,
- “Lo” and “Hi” as pre-limit switches for the valve partially closed or opened.

The limit switches for the fully closed or opened valve are permanently set to the position values zero (for the closed valve) and one hundred (for the open valve). Binary outputs “0” and “100” are therefore set to one when the valve is fully closed or open, i.e. when the position y of the component has the value zero or one hundred.

The pre-limit switches can be set using the “HI_Limit” or “LO_Limit” parameter. The value should be between zero and one hundred. Position values of five (LO_Limit) and 95 (HI_Limit) are set by default.

When the simulation is run, if one of the two parameter values is negative, the message "DriveVx: high and low parameters do not match“ (message category ERROR) is generated.

The "Initial_Value" parameter is used to set the valve drive to the starting position of "Closed" or "Open". The default setting for "Initial_Value" is "Closed".

The current position value of the drive is visualised as a bar display in the component control window (Figure 5-2). The button on the left labelled "MAN" is used to change the position value to the desired setting by moving the slider in the control window.

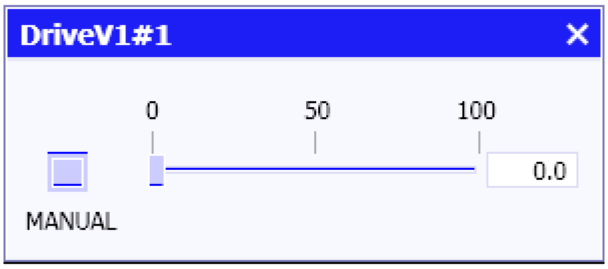
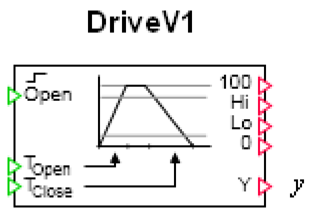


Figure 5-2: Control window for the component types for valve drives

The position value is then no longer derived from the component inputs and tracks the value set using the slider. The set opening and closing times remain active. The four binary outputs “0”, “Lo”, “Hi” and “100” all remain effective. They are also set as described above if the position value is to be tracked.

5.1.1 DriveV1 – type 1 valve drive

Symbol

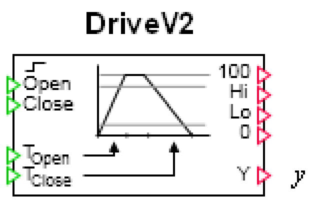


Function

The "DriveV1" component type simulates a drive unit that sets the position value y at the output in relation to a binary value at the input “Open”. If the binary input is equal to zero, then the position value y is continuously tracked to zero with the closing time T_{Close}. If the binary input is equal to one, then the position value is continuously tracked to one hundred with the opening time T_{Open}. The position value thus has only the two steady states of zero and one hundred, i.e. the states "Valve open” and “Valve closed” in relation to the valve function. The drive or the valve is thus "opened" or “closed” whenever the binary value at the input changes.

5.1.2 DriveV2 – type 2 valve drive

Symbol



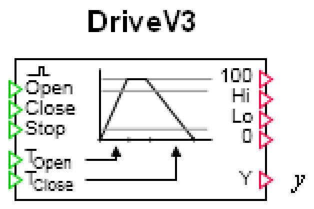
Function

The "DriveV2" component type simulates a drive unit that sets the position value y at the output in relation to two binary values and at the “Open” and “Close” inputs. If the binary value at input “Open” is equal to one, then the position value y is continuously tracked to one hundred with the opening time T_{Open}. If the binary value at the input “Close” is equal to one, then the position value is continuously tracked to zero with the closing time T_{Close}.

If both input values “Open” and “Close” are set to one or zero at the same time, then the position value remains unchanged. The position value thus only changes if the binary value at either the “Open” input or the “Close” input is set to one.

5.1.3 DriveV3 – type 3 valve drive

Symbol

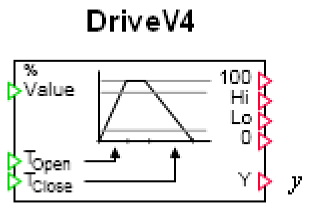


Function

The "DriveV3" component type simulates a drive unit that sets the position value y at the output in relation to three binary values at the “Open”, “Close” and “Stop” inputs. If the binary value at the “Open” input changes from zero to one, then the position value y is continuously tracked to one hundred with the opening time T_{Open}. The position value is tracked continuously to zero with the closing time T_{Close} if the binary value at the “Close” input changes from zero to one. If the binary value at the “Stop” input changes from zero to one, then the position value remains unchanged. “Opening”, “closing” and “stopping” of the drive is thus initiated via the edge of the corresponding binary input signal (signal change from zero to one).

5.1.4 DriveV4 – type 4 valve drive

Symbol



Function

The "DriveV4" component type simulates a drive unit that sets the position value y at the output that continuously tracks the analog value at the "Value" input. The tracking follows rising position values with the opening time T_{Open} and falling position values with the closing time T_{Close}. The position value is always limited to values from zero to one hundred, even if the input value is outside this interval.

5.2 Pump and fan drives

The two component types "DriveP1" and "DriveP2" may be used in the simulation as drives for pumps, fans or similar units. The outputs and some of the inputs common to the two component types are illustrated in Figure 5-3.

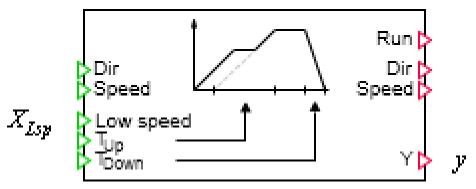


Figure 5-3: Common connections of the component types for pump drives

The run-up and run-down speeds of the drive are set at the two analog inputs T_{Up} and T_{Down}. T_{Up} is the time it takes the drive to run up from stopped to the nominal speed in seconds, while T_{Down} is the time in seconds taken to run the drive down from nominal speed to stopped. The two times are set to one second by default. When the simulation is run, if one of the two input values is negative, the message "DriveVx: run-up or run-down time invalid value“ (message category ERROR) is generated.

The current speed value of the drive is output at analog output y as a percentage, i.e. in the range from zero to one hundred:

$$0 \le y \le 100 \,.$$

The value zero corresponds to the stopped drive, while the value one hundred represents the drive at nominal speed.

The direction of rotation of the drive can be set using binary input “Dir”. This input is set to zero for the positive direction of rotation and is set to one for the negative direction of rotation. This input can therefore be used to defined whether the drive turns clockwise or anti-clockwise, for example. If it turns in the negative direction, the speed is output at output y as a negative value:

$$-100 \le y \le 0\,.$$

Positive values for y thus identify speeds in the positive direction, while negative values indicate speeds in the negative direction. The change in direction of rotation only takes effect if the drive is stopped.

The binary input “Speed” is used to toggle the speed between nominal speed (full speed) and a partial speed (low speed). If ”Speed” is one, the nominal speed was selected, otherwise the partial speed was selected. The default setting for the “Speed” binary input is one. The partial speed is specified as a numerical value (percentage) at the analog input x_{LSp}:

$$0 \le x_{LSp} \le 100\,.$$

The default setting for the partial speed is 50, i.e. half the nominal speed. If the partial speed is not within the defined range, the message "DrivePx: low speed invalid value“ (message category ERROR) is generated.

The binary output “Run” is then only set to one if the drive has reached the preset speed value in the positive or negative direction of rotation, i.e. only if the absolute value at the analog output y is equal to one hundred (nominal speed) or is equal to the partial speed set at the input x_{LSp}.

The feedback signal for the selected speed is available at the binary output “Speed”. The binary output is only set to one if the drive has reached its nominal speed in the positive or negative direction of rotation.

The value at the binary output “Dir” takes the form of a feedback signal for the current direction of rotation of the drive. The binary output is zero if the drive is turning in the positive direction and the value is one if the drive is turning in the negative direction.

The current speed value of the drive is visualised as a bar display in the component control window (Figure 5-4). The button on the left labelled "MAN” is used to change the speed value to the desired setting by moving the slider in the control window.

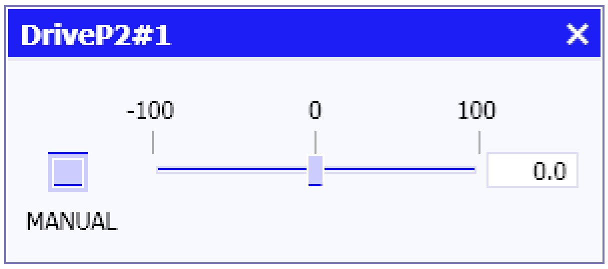
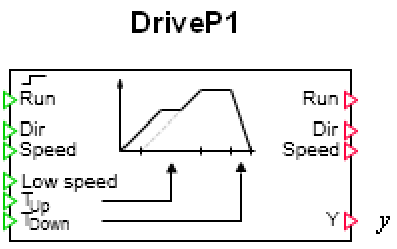


Figure 5-4: Control window for the component types for pump drives

The speed value is then no longer derived from the component inputs and tracks the value set using the slider. The set run-up and run-down times remain active. The binary outputs “Run”, “Dir” and “Speed” also remain effective. They are also set as described above if the speed value is to be tracked.

5.2.1 DriveP1 – type 1 pump drive

Symbol

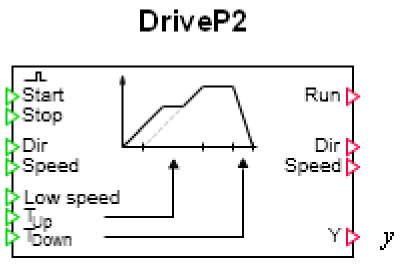


Function

The "DriveP1" component type simulates a drive unit that is switched on and off via the binary value at the "Run" input. The drive is switched on while the "Run" input value is set to one. If the input value is set to zero, then the drive is switched off.

5.2.2 DriveP2 – type 2 pump drive

Symbol



Function

The "DriveP2" component type simulates a drive unit that is switched on and off via the two binary inputs "Start" and "Stop". If the binary value "Start" changes from zero to one, then the drive is switched on. If the binary value "Stop" changes from zero to one, then the drive is switched off. Switching the drive on and off is thus initiated via the edge of the corresponding binary input signal (signal change from zero to one).