SIEMENS

SIMIT SCE

Basisbibliothek

Referenzhandbuch

SIEMENS

Ausgabestand

Juli 2009

Warenzeichen

SIMIT ist eine Marke der Siemens AG in Deutschland und in anderen Ländern.

Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Siemens AG
Industry
Industry Solutions
Industrial Technologies

SIMIT SCE-HB-V7BL-2009-07

Haftungsausschluß

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungs-vorschläge sind wir dankbar.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 VORWORT

1.1 Zielgruppe

Das vorliegende Handbuch wendet sich an Sie als Anwender des Simulationssystems SIMIT. Es beschreibt den Aufbau der Standardbibliothek des SIMIT-Basissystems sowie die in der Standardbibliothek enthaltenen Komponententypen. Die Komponententypen sind als zentrales Element von SIMIT die Basis der SIMIT-Modelle. Für die Erstellung von Modellen und das Durchführen von Simulationen ist die Kenntnis der Funktion der Komponententypen unabdingbar. Dieses Handbuch liefert die dafür erforderlichen Informationen.

Neben fundierten Kenntnissen im Umgang mit Personal Computern und ihrer Windows-Oberfläche werden Kenntnisse über das SIMIT-Basissystem ebenso wie grundlegende Kenntnisse der den Komponententypen zugrunde liegenden mathematischen Zusammenhänge vorausgesetzt.

1.2 Inhalt

In diesem Handbuch werden in der Hauptsache die in der Basisbibliothek enthaltenen Komponententypen beschrieben. In einem einleitenden Kapitel 2 wird zunächst die Struktur der Basisbibliothek beschrieben und es werden die allgemeinen Eigenschaften der Komponententypen erläutert. Dieses Kapitel ist Voraussetzung für das Verständnis der in den folgenden Kapiteln 3 bis 5 enthaltenen Beschreibungen der einzelnen Komponententypen. Es wird daher empfohlen in jedem Fall das Kapitel 2 zu lesen.

Die Kapitel 3 bis 5 können unabhängig voneinander gelesen werden. In Kapitel 3 werden die mit dem SIMIT-Basissystem in der Modellierung verwendbaren Konnektoren beschrieben. Kapitel 4 enthält Beschreibungen der Standardkomponententypen von SIMIT und die Kapitel 5 liefern Informationen über die in der Basisbibliothek enthaltenen Antriebskomponenten-typen.

Die Beschreibungen der einzelnen Komponententypen sind so gefasst, dass ihre prinzipielle Funktion klar zutage tritt. Besonderheiten der Implementierung werden nur dort erläutert, wo es zum Verständnis der Funktion einer Komponente notwendig ist.

1.3 Symbolik

Besonders wichtige Informationen werden im Text wie folgt hervorgehoben:

HINWEIS

Hinweise enthalten wichtige vertiefende Informationen zu den jeweiligen Dokumentationsinhalten. Sie stellen außerdem solche Eigenschaften von System oder Bedienung heraus, auf die wir Sie besonders aufmerksam machen wollen.

VORSICHT

bedeutet, dass sich das System nicht wie beschrieben verhält, wenn die angegebenen Vorsichtsmaßnahmen nicht beachtet werden.

WARNUNG

bedeutet, dass das System irreparablen Schaden nehmen könnte oder Datenverlust droht, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

2 EINFÜHRUNG

Die Basisbibliothek ist Bestandteil des Basissystems „SIMIT Basic“. Sie beinhaltet elementare Funktionen zur Modellierung von Anlagen- und Maschinenverhalten mit SIMIT. Wie in SIMIT üblich werden diese Funktionen in Form von Komponententypen zur Verfügung gestellt. In diesem Handbuch werden die einzelnen in der Basisbibliothek enthaltenen Komponententypen ausführlich erläutert.

Die Basisbibliothek ist in vier Abschnitte gegliedert:

- Konnektoren,
- Standardkomponenten, und
- Antriebskomponenten

Entsprechend dieser Gliederung sind die Komponententypen in drei verschiedenen Ordnern abgelegt:

- „CONNECTORS“,
- „STANDARD“ und
- „DRIVES“

2.1 Sprache

Die Namen aller Bibliothekskomponenten sowie zugehörige interne Namen für Anschlüsse, Parameter, Zustände und Meldungen sind einheitlich in englischer Sprache gehalten. Dies ist notwendig, um eine eindeutige und austauschbare Bibliotheksbasis für sowohl die deutsche als auch alle fremdsprachigen Versionen von SIMIT zu haben.

2.2 Allgemeines

Jeder Beschreibung eines Komponententyps ist sein Symbol vorangestellt. Da das Symbol eines Komponententyps bei der Erstellung der Simulation auf Diagrammen das zentrale Element ist, bildet es in diesem Handbuch den Bezugspunkt für die funktionale Beschreibung. Es enthält selbst verschiedene Elemente wie eine Beschriftung oder Grafik, sowie Anschlüsse mit Namen. Diese Elemente sind für jeden Komponententyp so gestaltet, dass sich die Funktion der Komponente wie auch der Anschlüsse bei der Erstellung der Simulation auf Diagrammen intuitiv erfassen lässt.

In der Beschreibung in diesem Handbuch werden weitere Anschlussbezeichnungen verwendet. Diese werden dann verwendet, wenn die funktionale Beschreibung es erfordert. Die einzelnen Elemente der Beschreibung eines Komponententyps sind am Symbol beispielsweise wie in Abbildung 2-1 dargestellt angeordnet.

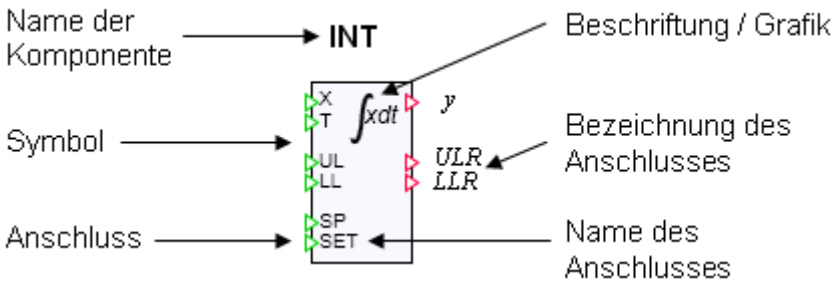


Abbildung 2-1: Elemente der Komponententypbeschreibung

Ein Komponententyp ist unter seinem Namen in der Bibliothek zu finden. Bei der Erstellung einer Simulation auf einem Diagramm werden Komponententypen mit ihrem Symbol als Komponenten instanziiert. Jedes Symbol besitzt deshalb eine Beschriftung oder Grafik, die die Funktion der Komponente auf Plänen leicht erkennen lässt.

Anschlüsse existieren als Eingänge oder als Ausgänge. Eingänge sind prinzipiell links, Ausgänge rechts am Komponentensymbol angeordnet. Alle Ein- und Ausgänge sind entweder binäre (logische), analoge oder ganzzahlige Ein- bzw. Ausgänge. Komplexe Anschlusstypen sind in den Komponenten der Basisbibliothek nicht verwendet. In den Erläuterungen werden die Werte der binären Ein- bzw. Ausgänge mit null und eins bezeichnet, wobei null für FALSE und eins für TRUE steht.

Funktional zusammengehörende Ein- und Ausgänge sind im Symbol gegenüberliegend angeordnet. Im obigen Beispiel sind dies beispielsweise Eingang x und Ausgang y. Des Weiteren sind funktional zusammengehörende Ein- und Ausgänge gruppiert und durch Leerplätze von anderen Gruppen abgesetzt. Dadurch sind die über die Verschaltung von Komponenten hergestellten funktionalen Zusammenhänge auf Plänen leichter zu erfassen. Im der obigen Beispielkomponente sind folgenden drei Gruppen gebildet:

- die Eingänge x und T zur Berechnung des Integralwerts am Ausgang y,
- die Begrenzungen „UL“ und „LL“ mit den binären Rückmeldungen b_{UL}, b_{LL} und
- Setzwert „SP“ und Setzbefehl „SET“ zum Setzen des Integralausgangs y.

Anschlüsse sind in der Beschreibung mit Bezeichnungen versehen, wenn es für die Erläuterung der Funktion erforderlich ist. Im obigen Beispiel ist damit die durch das Integral

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

beschriebene Funktion den Anschlüssen zuordenbar. Im Symbol sind Anschlüsse nur dann mit einem Namen versehen, wenn die Funktion des Anschlusses nicht klar erkennbar ist.

Im obigen Beispiel sind auch Größen, wie beispielsweise die Integrationszeit T, die für eine Simulation in der Regel als Parameter fest vorgegeben sind, als Eingangsgrößen definiert. Sie können damit auch verschaltet und so durch beliebige funktionale Zusammenhänge in der Simulation verändert werden. Beispielsweise kann mit den Komponenten der Standardbibliothek die Integrationszeit auf einen anderen Wert gesetzt werden, wenn die Eingangswerte x in einem bestimmten Bereich liegen (s. Abbildung 2-2).

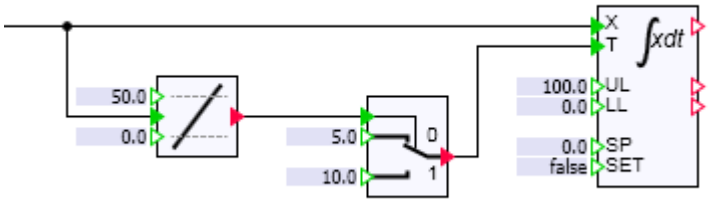


Abbildung 2-2: Intervallabhängige Umschaltung der Integrationszeit

Darüber hinaus können für Komponententypen auch Parameter definiert sein, die nur über den Parameter im Eigenschaftsfenster zugänglich sind.

2.2.1 Komponententypen mit variabler Anzahl an Eingängen

Bei den Komponententypen „ADD“, „MUL“, „AND“, „OR“ etc. ist die Anzahl der Eingänge einstellbar. Die Einstellung erfolgt grafisch am Symbol der Komponente bei der Erstellung einer Simulation. Als Indikator für eine einstellbare Eingangsanzahl ändert sich bei diesen Komponenten der Mauszeiger, wenn er über die untere oder obere Begrenzungslinie des Komponentensymbols auf einem Diagramm bewegt wird.

Die Anzahl der Eingänge können Sie einstellen, indem Sie die untere oder obere Begrenzungslinie des Symbols mit der linken Maustaste anklicken und mit niedergehaltener linker Maustaste nach unten oder oben ziehen (s. Abbildung 2-3).

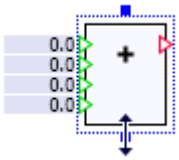


Abbildung 2-3: Einstellen der Anzahl der Eingänge am Komponentensymbol

3 KONNEKTOREN

Im Ordner „CONNECTORS“ der Basisbibliothek werden Konnektoren für die Modellierung zur Verfügung gestellt:

- Globale Konnektoren und
- Eingangs- und Ausgangskonnektoren.

Weitere Konnektoren werden mit spezifischen SIMIT-Modulen oder -Bibliotheken bereitgestellt. Diese Konnektoren werden dann ebenfalls in den Ordner „CONNECTORS“ eingefügt.

Globale Konnektoren und Eingangs- und Ausgangskonnektoren weisen die folgenden Gemeinsamkeiten auf:

Die Anschlüsse der Konnektoren sind untypisiert. Das bedeutet, dass die Konnektoren jeweils den Verbindungstyp des Anschlusses der angeschlossenen Komponente annehmen.

HINWEIS

Eine Typprüfung, d.h. die Prüfung, ob die über Konnektoren verbundenen Anschlüsse vom gleichen Typ sind, erfolgt automatisch vor dem Starten der Simulation. Falls Konnektoren mit Anschlüssen eines falschen Typs verbunden sind, wird von der Konsistenzprüfung ein entsprechender Hinweis ausgegeben und das Starten der Simulation wird abgebrochen.

Die Breite der Konnektor-Symbole auf einem Plan ist einstellbar und kann damit der Länge des Konnektornamens angepasst werden. Zum Einstellen der Breite bewegen Sie den Zeiger beim Ausgangs- bzw. Eingangskonnektor über den linken oder rechten Rand des Konnektors. Der Mauszeiger ändert seine Darstellung. Drücken Sie dann die linke Maustaste und bewegen Sie die Berandung mit niedergehaltener Maustaste nach links oder rechts auf die gewünschte Breite.

Der Konnektorname und gegebenenfalls der Planname werden im Konnektorsymbol auf dem Plan angezeigt.

3.1 Globale Konnektoren

Globale Konnektoren stellen Verbindungen von Komponenten über Plangrenzen hinweg zur Verfügung.

Der globale Konnektor kann sowohl als Ausgangs- als auch als Eingangskonnektor eingesetzt werden. Sein Symbol ist in Abbildung 3-1 dargestellt.

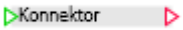


Abbildung 3-1: Globale Konnektoren

Eine Verbindung wird stets zwischen einem Ausgangskonnektor und einem oder mehreren zugehörigen Eingangskonnektoren hergestellt. Verbindungen mit globalen Konnektoren

erfordern lediglich die Angabe eines Namens für die Verbindung: den Konnektornamen.

HINWEIS

Beachten Sie dass die globalen Konnektornamen im gesamten Simulationsprojekt eindeutig sein müssen!

Der Konnektorname kann direkt in das Symbol eingetragen werden. Das Eingabefeld öffnet sich mit einem Doppelklick auf den Konnektor. Die Eingabe ist mit der „Return“-Taste abzuschließen.

3.2 Eingangs- und Ausgangskonnektoren

Die Komponenten „Input“ und „Output“ stellen die Verbindung zu Signalen in den SIMIT-Kopplungen her. Diese Komponenten können eine Verbindung zu Signalen einer jeden beliebigen SIMIT-Kopplung herstellen.

HINWEIS

Um konsistent mit den Bezeichnungen der Eingangs- und Ausgangssignale in den Kopplungen zu sein, werden die Eingangs- und Ausgangskonnektoren ebenfalls aus Sicht der angeschlossenen Steuerung benannt.

Es wird unterschieden zwischen Eingangs- und Ausgangskonnektoren („Input“ und „Output“) wie in Abbildung 3-2 dargestellt.

Abbildung 3-2: Eingangs- und Ausgangskonnektoren

Die Verbindung wird dadurch hergestellt, dass in der Eigenschaftsmaske des Eingangs- oder Ausgangskonnektors der Name der Kopplung und der Name des Kopplungsignals eingetragen werden (vgl. Abbildung 3-3).

Abbildung 3-3: Parametrierung der Ein- oder Ausgangssignale

Falls Sie den Kopplungsnamen auf dem Diagramm nicht dargestellt haben wollen, können Sie die Option „Kopplungsnamen anzeigen“ deaktivieren.

Beim Ausgangskonnektor wird der Wert des Signals durch die Kopplung gesetzt. Ein Ausgangskonnektor kann daher mehrfach im Simulationsprojekt – und auch auf verschiedenen Diagrammen – verwendet werden. Beim Eingangskonnektor wird der Wert des Signals im Simulationsprojekt gesetzt. Ein Eingangskonnektor darf daher nur einmal definiert sein.

3.3 Verbinder

Im Ordner „CONNECTORS“ der Basisbibliothek werden zwei sogenannte Verbinderkomponenten – kurz Verbinder genannt – zur Verfügung gestellt (Abbildung 3-4):

- Die analoge Verbinderkomponente „A“ und
- die binäre Verbinderkomponente „B“.

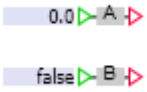


Abbildung 3-4: Verbinder-Komponenten

Eine Verbinderkomponente reicht den Eingangswert unverändert und ohne Verzögerung an ihren Ausgang weiter. Sie wird beispielsweise in den im Folgenden beschriebenen Fällen benötigt:

- Konnektoren (Eingangs- und Ausgangskonnektoren oder globale Konnektoren) können nicht direkt miteinander verbunden werden. Wie in Abbildung 3-5 beispielhaft dargestellt, können Konnektoren aber mit Hilfe von Verbinderkomponenten verbunden werden.

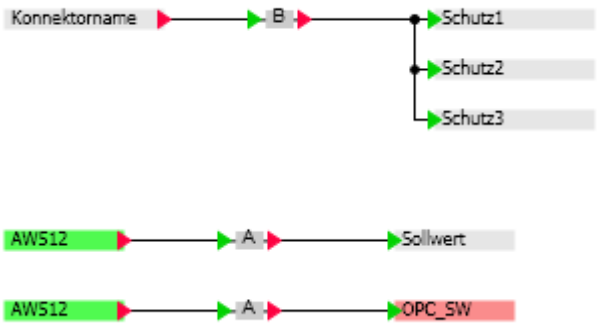


Abbildung 3-5: Verbindungen von Konnektoren

- Eingabeelemente wie beispielsweise ein Umschalter werden direkt mit dem zu setzenden Signal verknüpft. Mit Hilfe eines Verbinders kann ein Eingabeelement direkt auf mehrere Komponenten wirken. Wie in Abbildung 3-6 beispielhaft dargestellt, wird dazu das Eingangssignal „IN“ des Verbinders „VB“ mit dem Eingabeelement verknüpft.

Abbildung 3-6: Mehrfache Verbindungen mit einem Eingabeelement

- Komponenteneingänge können direkt mit einem Wert vorbelegt werden. Mit Hilfe eines Verbinders können mehrere Eingänge mit einem einzigen Wert vorbelegt werden. Wie in Abbildung 3-7 beispielhaft dargestellt, wird dazu ein Verbinder mit den zu setzenden Eingängen der Komponenten verbunden. Der zu setzende Wert wird dann am Eingang des Verbinders gesetzt.

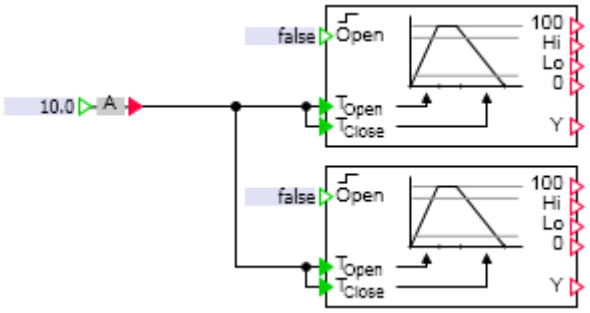


Abbildung 3-7: Vorbelegung von mehreren Eingängen mit einem Wert

4 STANDARDKOMPONENTEN

Die Standardkomponententypen der Basisbibliothek finden Sie im Bibliotheksordner „STANDARD“. Nach ihrer Funktion sind die Komponententypen unterteilt in Komponententypen mit

- analogen Funktionen,
- binären Funktionen,
- ganzzahligen Funktionen,
- Konvertierungsfunktionen,
- mathematischen Funktionen und
- verschiedene Hilfsfunktionen.

4.1 Analoge Funktionen

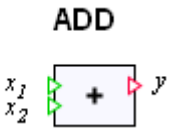
Alle Komponententypen mit analogen Funktionen sind in der Unterordnern „AnalogBasic“ und „AnalogExtended“ der Standardbibliothek enthalten. Unter „AnalogBasic“ sind dabei analoge Basisfunktionen eingeordnet, unter „AnalogExtended“ erweiterte Analogfunktionen.

4.1.1 Analoge Basisfunktionen

Die vier grundlegenden analogen Funktionen Addition, Subtraktion, Multiplikation und Division, also die vier arithmetischen Grundoperationen, sind als Komponententypen im Unterordner „AnalogBasic“ der Standardbibliothek abgelegt.

4.1.1.1 ADD – Addition

Symbol



Funktion

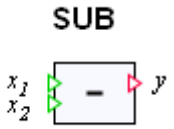
Der Komponententyp „ADD“ bildet die Summe der analogen Werte an den n Eingängen x_1 bis x_n auf den Ausgang y ab:

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

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbelegt.

4.1.1.2 SUB – Subtraktion

Symbol



Funktion

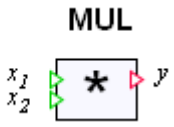
Der Komponententyp „SUB“ bildet die Differenz der analogen Werte an den beiden Eingängen x_1 und x_2 auf den Ausgang y ab gemäß

$$y = x_1 - x_2 .$$

Alle Eingänge sind mit null vorbesetzt.

4.1.1.3 MUL – Multiplikation

Symbol



Funktion

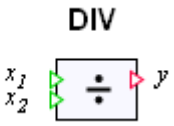
Der Komponententyp „MUL“ bildet das Produkt der analogen Werte an den n Eingängen x_1 bis x_n auf den Ausgang y ab:

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

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit eins vorbesetzt.

4.1.1.4 DIV – Division

Symbol



Funktion

Der Komponententyp „DIV“ bildet den Quotienten der analogen Werte an den Eingängen x_1 und x_2 auf den Ausgang y ab gemäß

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

Der Eingang x_1 ist mit null, der Divisioreingang x_2 ist mit eins vorbesetzt.

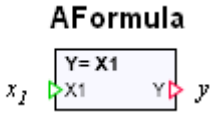
Der Wert des Divisors x_2 darf nicht null werden. Falls beim Durchführen der Simulation der Divisor null wird, wird die Fehlermeldung „DIV: division by zero“ (Meldekategorie ERROR) erzeugt und der Ausgang y wird auf null gesetzt.

4.1.2 Erweiterte analoge Funktionen

Im Unterordner „AnalogExtended“ der Standardbibliothek sind weitere analoge Funktionen als Komponententypen abgelegt.

4.1.2.1 AFormula – analoge Formelkomponente

Symbol



Funktion

Die Komponente „AFormula“ ermöglicht die Verwendung von expliziten algebraischen Funktionen. Diese Funktion f berechnet einen Wert in Abhängigkeit von den n Eingangswerten x_i. Der Funktionswert wird dem Ausgang *y* zugewiesen:

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

Zur Definition der Funktion tragen Sie im Parameter „Formula“ den gewünschten Formelausdruck ein, der den Ausgang y in Abhängigkeit von den Eingängen x_i berechnet. Setzen Sie dazu zunächst die gewünschte Anzahl an Eingängen und öffnen Sie dann zur Eingabe des Formelausdrucks den Eigenschaftsbereich der Komponente (s. Abbildung 4-1).

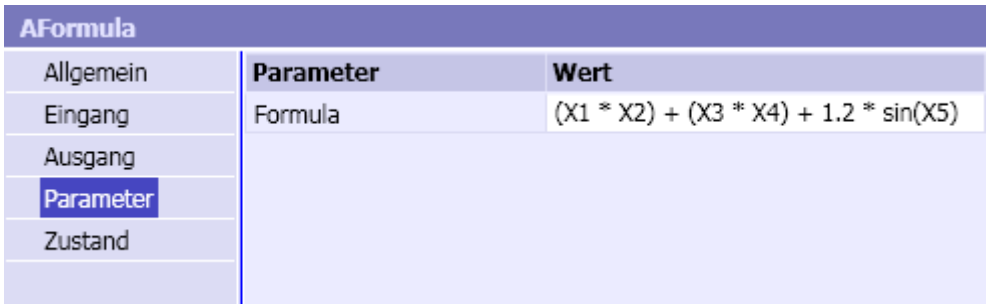


Abbildung 4-1: Parameter des Komponententyps „AFormula“

Die Anzahl n der Eingänge kann frei von 1 bis 32 variiert werden. Im Formelausdruck können nur die Eingänge benutzt werden, die gemäß der aktuell eingestellten Anzahl zur Verfügung stehen. Der Formelausdruck wird oben im Symbol der Komponente angezeigt. In Abbildung 4-2 ist beispielhaft eine Komponente mit drei Eingängen dargestellt.

Um längere Formelausdrücke vollständig sichtbar zu machen, können Sie die Komponente verbreitern. Bewegen Sie dazu den Mauszeiger über den rechten Rand der Komponente und ziehen Sie die Berandung bei gedrückter linker Maustaste auf die gewünschte Breite.

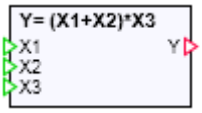


Abbildung 4-2: Komponente „AFormula“ mit drei Eingängen

Es sind die in Tabelle 4-1 aufgeführten Operatoren in Formelausdrücken verwendbar.

Operator	Funktion
+	Addition
-	Subtraktion
/	Division
*	Multiplikation
(	Öffnende Klammer
)	Schließende Klammer
Zahlenkonstante	Gleitkommazahlen, auch in Exponentialschreibweise
Funktionsaufrufe	Mathematische Standardfunktionen (siehe unten)

Tabelle 4-1: Operatoren in Formelausdrücken des Komponententyps „AFormula“

HINWEIS

Verwenden Sie bei der Eingabe von Gleitkommazahlen den Punkt als Dezimaltrennzeichen!

Als mathematische Standardfunktionen stehen die in der Programmiersprache C/C++ gebräuchlichen mathematischen Funktionen, wie in Tabelle 4-2 gelistet, zur Verfügung.

Eine detaillierte Beschreibung der Funktionen entnehmen Sie bitte der Dokumentation zu dem für SIMIT eingesetzten C/C++-Compiler. Der zur SIMIT-Software ausgelieferte Compiler unterstützt den Standardumfang nach ANSI-C. Eine Beschreibung dieses Standard-C-Umfangs können Sie beispielsweise auf den englischen Web-Seiten des „Microsoft Developer Network“ (MSDN) unter „http://msdn.microsoft.com/default.aspx“. Navigieren Sie dazu auf die entsprechenden Seiten der „Visual C++ Libraries Reference“.

HINWEIS

In der Formelkomponente wird nicht überprüft, ob im angegebenen Formelausdruck alle eingestellten Eingänge verwendet werden.

Formelausdruck	Funktion
sqrt(x)	$y = \sqrt{x}\ ;\quad x \geq 0$
fabs(x)	$y = \lvert x \rvert$
exp(x)	$y = \mathrm{e}^{x}$
pow(x, y)	$y = \mathrm{x}^{y}\ ;$
log(x)	Natürlicher Logarithmus: $y = \ln(x)\ ;\quad x > 0$
log10(x)	Dekadischer Logarithmus: $y = \lg(x)\ ;\quad x > 0$
ceil(x)	Kleinste ganze Zahl größer oder gleich x
floor(x)	Größte ganze Zahl kleiner oder gleich x
rand()	Zufallswert
sin(x)	$y = \sin(x)\ ;$ Winkel x im Bogenmaß
cos(x)	$y = \cos(x)\ ;$ Winkel x im Bogenmaß
tan(x)	$y = \tan(x)\ ;$ Winkel x im Bogenmaß; $x \neq \pm(2n+1)\pi/2$
asin(x)	$y = \arcsin(x)\ ;\quad -\pi/2 \leq y \leq \pi/2$
acos(x)	$y = \arccos(x)\ ;\quad 0 \leq y \leq \pi$
atan(x)	$y = \arctan(x)\ ;\quad -\pi/2 \leq y \leq \pi/2$
atan2(z, x)	$y = \arctan(x/z)\ ;\quad -\pi \leq z \leq \pi$
sinh(x)	$y = \sinh(x)\ ;$ Winkel x im Bogenmaß
cosh(x)	$y = \cosh(x)\ ;$ Winkel x im Bogenmaß
tanh(x)	$y = \tanh(x)\ ;$ Winkel x im Bogenmaß

Tabelle 4-2: Mathematische Funktionen in Formelausdrücken des Komponententyps „AFormula“

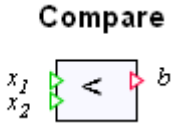
ACHTUNG

In der Formelkomponente wird nicht überprüft, ob die Argumente des Formelausdrucks gültige Werte aufweisen. Tritt während der Simulation in der Berechnung des Formelausdrucks eine Division durch null auf, dann hat der Ausgang y den Wert „**Inf“**. Ist ein Argument eines Formelausdrucks während der Simulation unbestimmt, wie beispielsweise die Division null durch null, dann hat der Ausgang y den Wert „**NaN“** (not a number).

Diese irregulären Ausgangswerte setzen sich in alle von diesem Ausgang abhängige Werte fort. Ihre Simulation gerät damit in einen nicht definierten Zustand. Sorgen Sie daher zur Vermeidung dieser Situation dafür, dass die Eingänge der Formelkomponente nur Werte annehmen, die die Gültigkeit der Argumente im Formelausdruck sichern.

4.1.2.2 Compare – Vergleichsfunktionen

Symbol



Funktion

Die Komponente „Compare“ vergleicht die analogen Eingänge x_1 und x_2. Der binäre Ausgang b wird auf eins gesetzt, wenn der Vergleichsaudruck wahr ist – andernfalls wird b auf null gesetzt.

Die Art des Vergleichs wird über den Parameter „Comparison“ bestimmt.

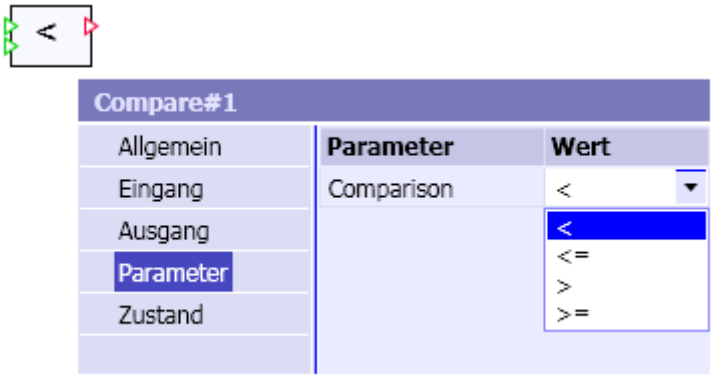


Abbildung 4-3: Parameter des Komponententyps „Compare“

Es sind die folgenden Vergleiche einstellbar:

- Vergleich „kleiner“, also $b = \begin{cases} 1, & \text{falls } x_1 < x_2 \\ 0, & \text{sonst} \end{cases}$,
- Vergleich „kleiner gleich“, also $b = \begin{cases} 1, & \text{falls } x_1 \le x_2 \\ 0, & \text{sonst} \end{cases}$,
- Vergleich „größer“, also $b = \begin{cases} 1, & \text{falls } x_1 > x_2 \\ 0, & \text{sonst} \end{cases}$ und
- Vergleich „größer gleich“, also $b = \begin{cases} 1, & \text{falls } x_1 \ge x_2 \\ 0, & \text{sonst} \end{cases}$.

Der gewählte Vergleichsoperator wird im Symbol der Komponente dargestellt (s. Abbildung 4-4).

Um die Vergleichsoperatoren „kleiner gleich“ und „größer gleich“ etwas von der rechten Berandung des Symbols absetzen zu können, kann die Breite des Symbols verändert werden. Zur Anpassung der Breite bewegen Sie den Mauszeiger über den rechten Rand des Symbols. Der Mauszeiger ändert sein Aussehen. Drücken Sie dann die linke Maustaste und verschieben Sie den Rand wie gewünscht mit der Maus.

Abbildung 4-4: Darstellung des Vergleichsoperators im Symbol

Da analoge Größen in SIMIT auf Variablen vom Typ „double“ abgebildet werden, ist ein unmittelbarer Vergleich auf Gleichheit oder Ungleichheit nicht sinnvoll. Gleichheit von double-Größen ist nur innerhalb der durch den Rechner vorgegebenen Genauigkeit (Maschinengenauigkeit) gegeben. Gleichheit lässt sich beispielsweise mit dem in Abbildung 4-5 dargestellten Modell prüfen.

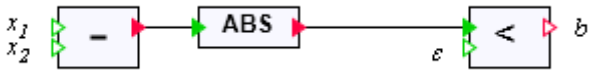


Abbildung 4-5: Prüfung auf Gleichheit mit der Vergleichskomponente

Alternativ kann ein funktional identisches Modell beispielsweise unter Verwendung des Formelbausteins „AFormula“ angesetzt werden (s. Abbildung 4-6).

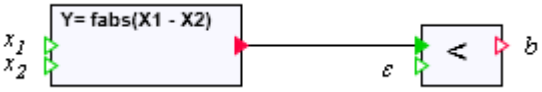


Abbildung 4-6: Prüfung auf Gleichheit mit der Formelkomponente "AFormula"

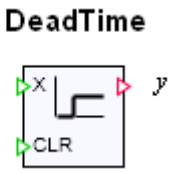
Von den beiden Größen x_1 und x_2 wird die Differenz gebildet. Der Betrag dieser Differenz wird dann mit einer vorgebbaren positiven Schranke ε verglichen:

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

Für die Prüfung auf Ungleichheit ist in den beiden skizzierten Modellen lediglich in der Vergleichs-Komponente der Vergleich „größer“ einzustellen.

4.1.2.3 DeadTime – Totzeitglied

Symbol



Funktion

Mit dem Komponententyp „DeadTime“ wird ein Totzeitglied zur Verfügung gestellt. Der Analogwert am Eingang x wird mit einer einstellbaren Verzögerung an den Ausgang y weitergegeben.

Eine sprungförmige Änderung des Eingangswertes x von null auf eins ergibt somit den Verlauf des Ausgangswerts y wie in Abbildung 4-11 dargestellt.

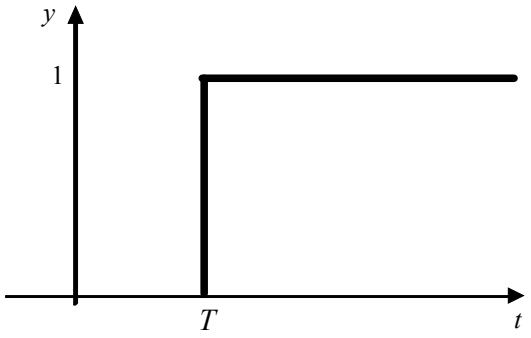


Abbildung 4-7: Sprungantwort des Totzeitgliedes

Die Verzögerungszeit T ist als Parameter „Delay_Cycles“ des Komponententyps in ganzzahligen Vielfachen n der Zykluszeit Δt einstellbar: $T = n\Delta t$. Sie ist mit zehn voreingestellt und kann maximal gleich 128 gesetzt werden. In der Voreinstellung mit einer Zykluszeit von 100 ms ergibt sich somit eine Verzögerung von einer Sekunde.

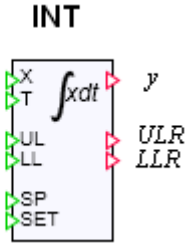
ACHTUNG

Wenn Sie die Zykluszeit Δt in den Projekteigenschaften ändern, ändern sich entsprechend auch die eingestellten Totzeiten.

In der Komponente werden die Eingangswerte gespeichert und verzögert, entsprechend der eingestellten Totzeit, am Ausgang ausgegeben. Der Speicher wird entsprechend der parametrierten Anzahl an Verzögerungszyklen für n Werte angelegt. Alle Speicherstellen sind mit null vorbelegt. Über den binären Eingang „CLR“ können der Ausgangswert y und alle Speicherstellen n auf null gesetzt werden.

4.1.2.4 INT – Integration

Symbol



Funktion

Der Komponententyp „INT“ bildet das Integral über das zeitabhängige analoge Eingangssignal x gemäß

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

Der Integralwert wird dem analogen Ausgang y zugewiesen.

Bei einem sprungförmigen Eingangssignal der Amplitude eins ergibt sich daher ein linear ansteigendes Ausgangssignal wie in Abbildung 4-8 dargestellt.

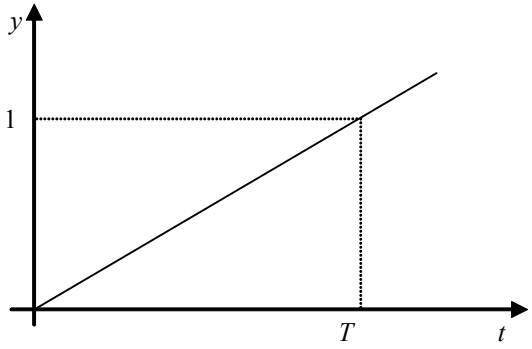


Abbildung 4-8: Sprungantwort der Integrationsfunktion

Der Integralwert y ist auf ein Intervall beschränkt das durch die beiden Grenzwerte „UL“ (obere Grenze) und „LL“ (untere Grenze) definiert ist:

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

Die binären Ausgänge ULR und LLR zeigen an, dass der Integralwert die untere bzw. obere Grenze erreicht hat:

$$ULR = \begin{cases} 1, & \text{falls} \quad y \ge UL \\ 0, & \text{sonst} \end{cases} \qquad \text{und} \qquad LLR = \begin{cases} 1, & \text{falls} \quad y \le LL \\ 0, & \text{sonst} \end{cases} .$$

Der untere Grenzwert muss kleiner als der obere Grenzwert sein. Wird diese Bedingung verletzt, dann wird die Fehlermeldung „INT: limits do not match“ (Meldekategorie ERROR) erzeugt, und der Ausgang y wird auf null gesetzt.

Die Zeitkonstante *T* der Integration muss einen positiven Wert haben. Falls *T* nicht positiv ist, wird die Fehlermeldung „INT: zero or negative time constant“ (Meldekategorie ERROR) erzeugt und der Ausgang y wird auf null gesetzt.

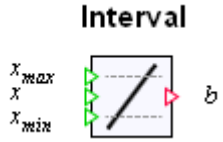
Der untere Grenzwert ist mit null, der obere Grenzwert mit hundert vorbelegt. Die Zeitkonstante *T* ist auf eine Sekunde voreingestellt. Alle anderen Eingänge sind mit null vorbelegt.

Der Komponententyp hat nur den Parameter „Initial_Value“. Auf diesen Wert wird der Intergrationswert y beim Initialisieren (Starten) der Simulation gesetzt. Der Parameter ist mit null vorbelegt.

Über den binären Eingang „SET“ kann der Integrationswert y auf den Wert am Eingang „SP“ gesetzt werden: falls „SET“ auf eins gesetzt ist, wird der Integrationswert y gleich dem Wert am Eingang „SP“ gesetzt. Auch in diesem Fall wird der Integrationswert durch „LL“ und „UL“ begrenzt.

4.1.2.5 Interval – Intervallabfrage

Symbol



Funktion

Der Komponententyp „Interval“ prüft, ob ein Eingangswert x im abgeschlossenen Intervall $[x_{min}, x_{max}]$ liegt. Liegt der Eingangswert im vorgegebenen Intervall, dann wird der Binärausgang auf eins gesetzt, andernfalls auf null:

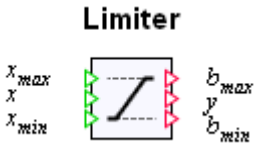
$$b = \begin{cases} 1 & \text{für} \quad x_{min} \le x \le x_{max} \\ 0 & \text{sonst} \end{cases},$$

Voreingestellt ist ein Intervall von null bis hundert, d.h. x_{min} hat die Vorbelegung null und x_{max} die Vorbelegung hundert.

Die obere Intervallgrenze darf nicht kleiner als die untere Intervallgrenze sein. Falls beim Durchführen der Simulation die obere Intervallgrenze kleiner als die untere wird, wird die Fehlermeldung „Limiter: limits do not match“ (Meldekategorie ERROR) erzeugt und der Ausgang b wird auf null (FALSE) gesetzt.

4.1.2.6 Limiter – Begrenzung

Symbol



Funktion

Der Komponententyp „Limiter“ bildet einen auf den Bereich von x_{min} bis x_{max} begrenzten Eingangswert auf den Ausgang y ab:

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

Die binären Ausgänge b_{min} und b_{max} werden auf eins gesetzt, wenn die Begrenzung wirksam wird, also

$$b_{max} = \begin{cases} 1, & \text{falls} \quad x \ge x_{max} \\ 0, & \text{sonst} \end{cases} \qquad \text{und} \qquad b_{min} = \begin{cases} 1, & \text{falls} \quad x \le x_{min} \\ 0, & \text{sonst} \end{cases}.$$

Voreingestellt sind die Begrenzungen x_{min} gleich null und x_{max} gleich hundert.

Die obere Begrenzung darf nicht kleiner als die untere Begrenzung sein. Falls beim Durchführen der Simulation die untere Begrenzung kleiner als die obere wird, wird die Fehlermeldung „Limiter: limits do not match“ (Meldekategorie ERROR) erzeugt und der Ausgang y wird auf null gesetzt.

4.1.2.7 MinMax – Minimal- und Maximalwertauswahl

Symbol

Funktion

Der Komponententyp „MinMax“ bildet das Minumum oder Maximum der n Eingänge x_1 bis x_n auf den Ausgang y ab. Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbesetzt.

Die Art der Abbildung wird über den Parameter „MinMax“ im Eigenschaftsfenster der Komponente bestimmt (s. Abbildung 4-9).

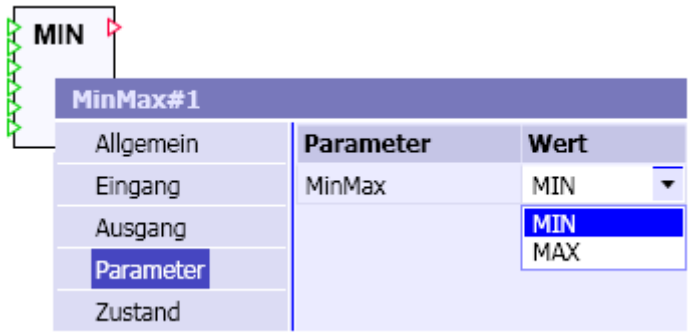


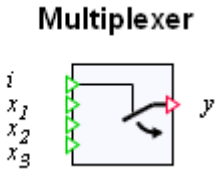
Abbildung 4-9: Parameter des Komponententyps „MinMax“

Die für eine Komponente eingestellte Abbildung „MIN“ oder „MAX wird im Komponentensymbol angezeigt wie in Abbildung 4-10 gezeigt.

Abbildung 4-10: Auswahl im Symbol des Komponententyps „MinMax“

4.1.2.8 Multiplexer

Symbol



Funktion

Der Komponententyp „Multiplexer“ schaltet einen der n verschiedenen Eingänge x_i in Abhängigkeit vom Wert am Selektionseingang i auf den Ausgang y durch:

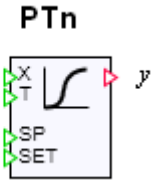
$$y = x_i \quad \text{für } 1 \le i \le n \,.$$

Die Anzahl n der Eingänge x_i ist variabel und kann auf einen Wert zwischen 3 und 32 eingestellt werden. Alle Eingänge sind mit null vorbelegt.

Der Wert am Selektionseingang i wird intern auf 1 bis n begrenzt, d.h. für Werte kleiner als eins wird i auf Eins gesetzt, für Werte größer n wird i auf den Wert n gesetzt. Damit ist in der Voreinstellung der erste Eingang x_i auf den Ausgang y durchgeschaltet.

4.1.2.9 PTn – Verzögerung n-ter Ordnung

Symbol



Funktion

Mit dem Komponententyp „PTn“ wird eine Verzögerung n-ter Ordnung zur Verfügung gestellt.

Bei einer Verzögerung erster Ordnung folgt der Funktionswert y am Ausgang dem Wert x am Eingang verzögert entsprechend der Differentialgleichung

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

Eine sprungförmige Änderung des Eingangswertes x von null auf eins ergibt somit einen exponentiellen Verlauf des Ausgangswerts y (Sprungantwort) gemäß

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

wie in Abbildung 4-11 dargestellt.

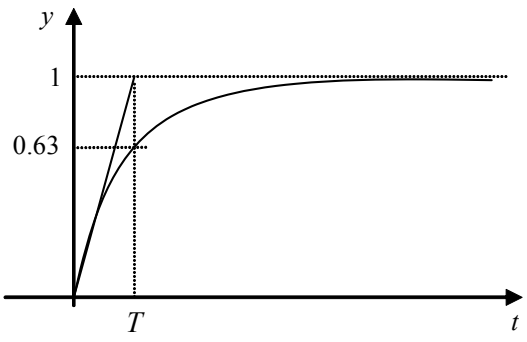


Abbildung 4-11: Sprungantwort der Verzögerungsfunktion 1. Ordnung

Bei Verzögerungen höherer Ordnung wird der Ausgangswert y aus der Verkettung von Verzögerungen erster Ordnung gebildet:

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

$$y = z_n .$$

Dabei sind $z_i, i = 1,...,n,$ die n Zustände der Verzögerung n-ter Ordnung. Bildlich gesehen entspricht eine Verzögerung n-ter Ordnung der Hintereinanderschaltung von n Verzögerungen erster Ordnung.

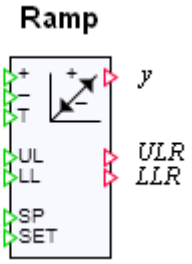
Die Ordnung n der Verzögerung ist als Parameter des Komponententyps einstellbar. Sie ist mit eins voreingestellt und kann maximal gleich 32 gesetzt werden. Der Parameter „Initial_Value“ mit der Vorbelegung null dient zum Initialisieren der Zustände der Verzögerungsfunktion.

Um zu vermeiden, dass die Komponente für zu kleine Verzögerungszeiten instabiles Verhalten annimmt, wird die Verzögerungszeitkonstante T auf Werte begrenzt, die größer oder gleich der Zykluszeit der Komponente sind. Sind die Werte am Eingang T kleiner als die Zykluszeit, dann wird die Fehlermeldung „PTn: delay time below cycle time“ (Meldekategorie ERROR) erzeugt. Die Verzögerungszeit ist mit 1 sec vorbelegt. Alle anderen Eingänge der Komponente sind mit null vorbelegt.

Über den binären Eingang „SET“ können der Ausgangswert y und die Zustandswerte $z_i, i = 1,...,n,$ auf den Wert am Eingang „SP“ gesetzt werden: falls „SET“ auf eins gesetzt ist, werden diese Werte gleich dem Wert am Eingang „SP“ gesetzt.

4.1.2.10 Ramp – Rampenfunktion

Symbol



Funktion

Der Komponententyp „Ramp“ inkrementiert oder dekrementiert seinen Funktionswert y in jedem Zeitschritt der Simulation um den Wert

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

wobei Δt die Schrittweite der Simulation ist. Der Rampenwert y wird um Δy inkrementiert, wenn der Eingang „+“ (UP) mit eins belegt ist. Ist der Eingang „-“ (DOWN) mit eins belegt, wird der Rampenwert y um Δy dekrementiert. Sind beide Eingänge „+“ und „-“ mit eins belegt, dann wird der Rampenwert y nicht verändert.

Der Rampenwert ist auf ein Intervall beschränkt das durch die beiden Grenzwerte „UL“ (obere Grenze) und „LL“ (untere Grenze) definiert ist:

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

Die binären Ausgänge ULR und LLR zeigen an, dass der Rampenwert die untere bzw. obere Grenze erreicht hat:

$$ULR = \begin{cases} 1, & \text{falls } y \geq UL \\ 0, & \text{sonst} \end{cases} \quad \text{und} \quad LLR = \begin{cases} 1, & \text{falls } y \leq LL \\ 0, & \text{sonst} \end{cases}.$$

Der untere Grenzwert muss kleiner als der obere Grenzwert sein. Wird diese Bedingung verletzt, dann wird die Fehlermeldung „RAMP: limits do not match“ (Meldekategorie ERROR) erzeugt, und der Ausgang y wird auf null gesetzt .

Die Zeitkonstante *T* muss einen positiven Wert haben. Falls *T* nicht positiv ist, wird die Fehlermeldung „INT: zero or negativ time constant“ (Meldekategorie ERROR) erzeugt und der Ausgang y wird auf null gesetzt.

Die untere Grenze ist mit null, die obere Grenze ist mit hundert vorbelegt. Die Zeitkonstante ist mit 10 Sekunden voreingestellt. Alle anderen Eingänge sind mit null vorbelegt.

Der Komponententyp hat nur den Parameter „Initial_Value“. Auf diesen Wert wird der Rampenfunktionswert y beim Initialisieren (Starten) der Simulation gesetzt. „Initial_Value“ ist mit null vorbelegt.

Über den binären Eingang „SET“ kann der Rampenwert y auf den Wert am Eingang „SP“ gesetzt werden: falls „SET“ auf eins gesetzt ist, wird der Rampenwert y gleich dem Wert am Eingang „SP“ gesetzt. Auch hier wird der Rampenwert durch „LL“ und „UL“ begrenzt.

Über das Bedienfenster der Komponente ist es möglich, den Rampenfunktionswert während einer Simulation manuell zu setzen. Öffnen Sie dazu das Bedienfenster (s. Abbildung 4-12).

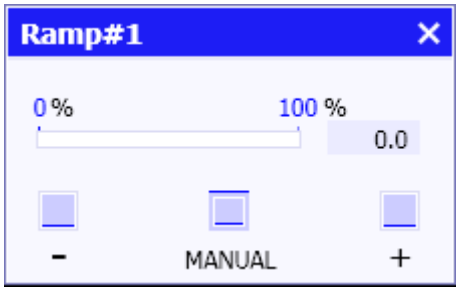


Abbildung 4-12: Bedienfenster des Komponententyps „Ramp“

Schalten Sie die Rampenfunktion mit dem Schalter „MANUAL“ in den manuellen Modus. Der Rahmen des Schalters verändert seine Farbe nach dunkelblau. Durch Drücken der Taster „-“ oder „+“ können Sie nun den Rampenfunktionswert vermindern oder vergrößern. Der aktuelle Funktionswert wird als Dezimalwert und als Prozentualwert angezeigt (s. Abbildung 4-13).

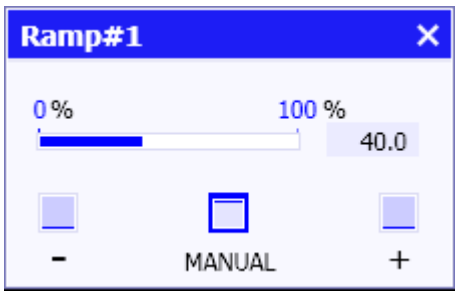
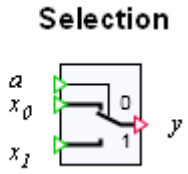


Abbildung 4-13: Bedienfenster des Komponententyps „Ramp“ im manuellen Modus

4.1.2.11 Selection – Analogschalter

Symbol



Funktion

Der Komponententyp „Selection“ schaltet einen der beiden Eingänge x_0 oder x_1 in Abhängigkeit von der Belegung des Binäreingangs a auf den Ausgang y durch.

Ist der Auswahleingang $a = 0$, so wird der Eingang x_0 durchgeschaltet; ist der Auswahleingang $a = 1$, so wird der Eingang x_1 durchgeschaltet:

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

Alle Eingänge sind mit null vorbelegt.

4.2 Ganzzahlige Funktionen

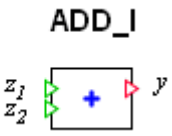
Alle Komponententypen mit ganzzahligen Funktionen sind in den Unterordnern „IntegerBasic“ und „IntegerExtended“ der Standardbibliothek enthalten. Unter „IntegerBasic“ sind dabei ganzzahlige Basisfunktionen eingeordnet, unter „IntegerExtended“ erweiterte ganzzahlige Funktionen.

4.2.1 Ganzzahlige Basisfunktionen

Die vier grundlegenden ganzzahligen Funktionen Addition, Subtraktion, Multiplikation und Division, also die vier arithmetischen Grundoperationen, sind als Komponententypen im Unterordner „IntegerBasic“ der Standardbibliothek abgelegt.

4.2.1.1 ADD_I – Addition

Symbol



Funktion

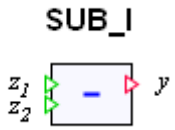
Der Komponententyp „ADD_I“ bildet die Summe der ganzzahligen Werte an den n Eingängen z_1 bis z_n auf den Ausgang y ab:

$$y = \sum_{i=1}^{n} z_i = z_1 + z_2 + \ ... \ + z_n .$$

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbelegt.

4.2.1.2 SUB_I – Subtraktion

Symbol



Funktion

Der Komponententyp „SUB_I“ bildet die Differenz der ganzzahligen Werte an den beiden Eingängen z_1 und z_2 auf den Ausgang y ab gemäß

$$y = z_1 - z_2 .$$

Alle Eingänge sind mit null vorbesetzt.

4.2.1.3 MUL_I – Multiplikation

Symbol

Funktion

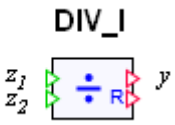
Der Komponententyp „MUL_I“ bildet das Produkt der ganzzahligen Werte an den n Eingängen z_1 bis z_n auf den Ausgang y ab:

$$y = \prod_{i=1}^{n} z_i = z_1 z_2 \ \ldots \ z_n .$$

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit eins vorbesetzt.

4.2.1.4 DIV_I – Ganzzahlige Division

Symbol



Funktion

Der Komponententyp „DIV_I“ bildet den Quotienten der ganzzahligen Werte an den Eingängen z_1 und z_2 gemäß

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

als ganzzahlige Division mit Rest. Am Ausgang y steht das ganzzahlige Ergebnis der Division, am Ausgang R der Rest der ganzzahligen Division zur Verfügung.

Der Divident am Eingang z_1 ist mit null, der Divisioreingang z_2 ist mit eins vorbesetzt.

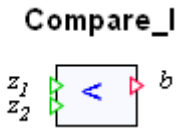
Der Wert des Divisors z_2 darf nicht null werden. Falls beim Durchführen der Simulation der Divisor null wird, wird die Fehlermeldung „DIV: division by zero“ (Meldekategorie ERROR) erzeugt und der Ausgang y und Rest R werden auf null gesetzt.

4.2.2 Erweiterte ganzzahlige Funktionen

Im Unterordner „IntegerExtended“ der Standardbibliothek sind weitere ganzzahlige Funktionen als Komponententypen abgelegt.

4.2.2.1 Compare_I – Vergleichsfunktionen

Symbol



Funktion

Die Komponente „Compare_I“ vergleicht die ganzzahligen Eingänge z_1 und z_2. Der binäre Ausgang b wird auf eins gesetzt, wenn der Vergleichsaudruck wahr ist – andernfalls wird b auf null gesetzt.

Die Art des Vergleichs wird über den Parameter „Comparison“ bestimmt.

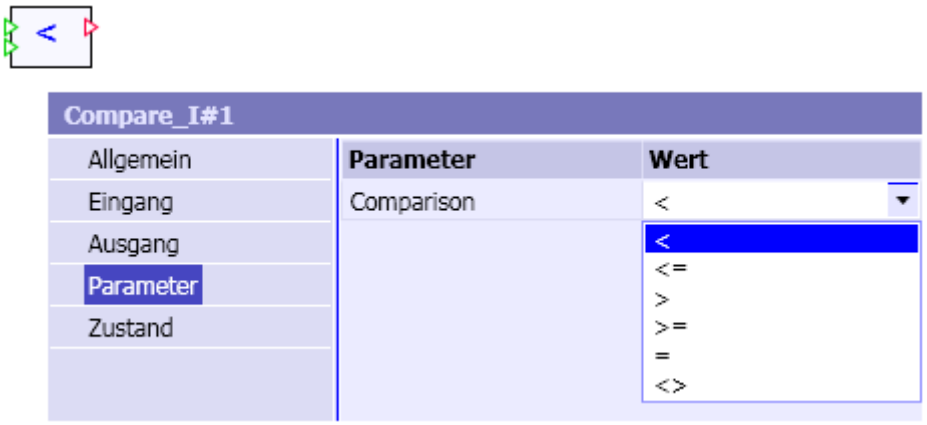


Abbildung 4-14: Parameter des Komponententyps „Compare_I“

Es sind die folgenden Vergleiche einstellbar:

- Vergleich „kleiner“, also $b = \begin{cases} 1, & \text{falls } z_1 < z_2 \\ 0, & \text{sonst} \end{cases}$,
- Vergleich „kleiner gleich“, also $b = \begin{cases} 1, & \text{falls } z_1 \le z_2 \\ 0, & \text{sonst} \end{cases}$,
- Vergleich „größer“, also $b = \begin{cases} 1, & \text{falls } z_1 > z_2 \\ 0, & \text{sonst} \end{cases}$ und
- Vergleich „größer gleich“, also $b = \begin{cases} 1, & \text{falls } z_1 \ge z_2 \\ 0, & \text{sonst} \end{cases}$.

- Vergleich „gleich“, also $b = \begin{cases} 1, & \text{falls } z_1 = z_2 \\ 0, & \text{sonst} \end{cases}$.
- Vergleich „ungleich“, also $b = \begin{cases} 0, & \text{falls } z_1 \neq z_2 \\ 1, & \text{sonst} \end{cases}$.

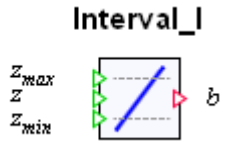
Der gewählte Vergleichsoperator wird im Symbol der Komponente dargestellt (s. Abbildung 4-4).

Um die Vergleichsoperatoren „kleiner gleich“, „größer gleich“ und „ungleich" etwas von der rechten Berandung des Symbols absetzen zu können, kann die Breite des Symbols verändert werden. Zur Anpassung der Breite bewegen Sie den Mauszeiger über den rechten Rand des Symbols. Der Mauszeiger ändert sein Aussehen. Drücken Sie dann die linke Maustaste und verschieben Sie den Rand wie gewünscht mit der Maus.

Abbildung 4-15: Darstellung des Vergleichsoperators im Symbol

4.2.2.2 Interval_I – Intervallabfrage

Symbol



Funktion

Der Komponententyp „Interval_I“ prüft, ob ein ganzzahliger Eingangswert z im abgeschlossenen Intervall $[z_{min}, z_{max}]$ liegt. Liegt der Eingangswert im vorgegebenen Intervall, dann wird der Binärausgang auf eins gesetzt, andernfalls auf null:

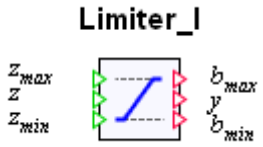
$$b = \begin{cases} 1 & \text{für } z_{min} \le z \le z_{max}, \\ 0 & \text{sonst} \end{cases}.$$

Voreingestellt ist ein Intervall von null bis hundert, d.h. z_{min} hat die Vorbelegung null und z_{max} die Vorbelegung hundert.

Die obere Intervallgrenze darf nicht kleiner als die untere Intervallgrenze sein. Falls beim Durchführen der Simulation die obere Intervallgrenze kleiner als die untere wird, wird die Fehlermeldung „Limiter: limits do not match“ (Meldekategorie ERROR) erzeugt und der Ausgang b wird auf null (FALSE) gesetzt.

4.2.2.3 Limiter_I – Begrenzung

Symbol



Funktion

Der Komponententyp „Limiter_I“ bildet einen auf den Bereich von z_{min} bis z_{max} begrenzten Eingangswert auf den Ausgang y ab:

$$y = \begin{cases} z_{max} & \text{für } z \geq z_{max} \\ z & \text{für } z_{min} < z < z_{max} \\ z_{min} & \text{für } z \leq z_{min} \end{cases}$$

Die binären Ausgänge b_{min} und b_{max} werden auf eins gesetzt, wenn die Begrenzung wirksam wird, also

$$b_{max} = \begin{cases} 1, & \text{falls } z \geq z_{max} \\ 0, & \text{sonst} \end{cases} \quad \text{und} \quad b_{min} = \begin{cases} 1, & \text{falls } z \leq z_{min} \\ 0, & \text{sonst} \end{cases}.$$

Voreingestellt sind die Begrenzungen z_{min} gleich null und z_{max} gleich hundert.

Die obere Begrenzung darf nicht kleiner als die untere Begrenzung sein. Falls beim Durchführen der Simulation die untere Begrenzung kleiner als die obere wird, wird die Fehlermeldung „Limiter: limits do not match“ (Meldekategorie ERROR) erzeugt und der Ausgang y wird auf null gesetzt.

4.2.2.4 MinMax_I – Minimal- und Maximalwertauswahl

Symbol

Funktion

Der Komponententyp „MinMax_I“ bildet das Minumum oder Maximum der n Eingänge z_1 bis z_n auf den Ausgang y ab. Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbesetzt.

Die Art der Abbildung wird über den Parameter „MinMax“ im Eigenschaftsfenster der Komponente bestimmt (s. Abbildung 4-9).

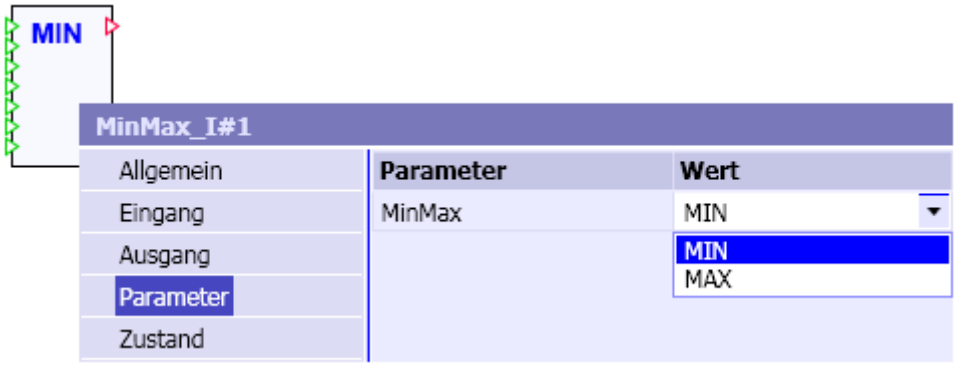


Abbildung 4-16: Parameter des Komponententyps „MinMax“

Die für eine Komponente eingestellte Abbildung „MIN“ oder „MAX wird im Komponentensymbol angezeigt wie in Abbildung 4-10 gezeigt.

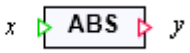
Abbildung 4-17: Auswahl im Symbol des Komponententyps „MinMax_I“

4.3 Mathematische Funktionen

Im Unterordner „Math“ der Standardbibliothek sind die gebräuchlichsten mathematischen Funktionen in Form von Komponententypen enthalten: die Absolutwertbildung (ABS), das Radizieren (SQRT), der Natürliche Logarithmus (LN) und die Exponentialfunktion (EXP), sowie die trigonometrischen Funktionen Sinus (SIN), Cosinus (COS) und Tangens (TAN).

4.3.1 ABS – Absolutwert

Symbol



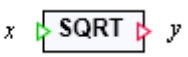
Funktion

Die Komponente „ABS“ bildet den Absolutwert (Betrag) des Eintritts x auf den Ausgang y ab:

$$y = |x| .$$

4.3.2 SQRT – Quadratwurzel

Symbol



Funktion

Die Komponente „SQRT“ bildet die Quadratwurzel des Eintritts x auf den Ausgang y ab:

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

Der Radikand x darf nicht negativ werden. Falls beim Durchführen der Simulation der Radikand negativ wird, wird die Fehlermeldung „SQRT: invalid argument“ (Meldekategorie ERROR) erzeugt, und der Ausgang y wird auf null gesetzt.

4.3.3 EXP – Exponentialfunktion

Symbol

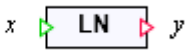
Funktion

Die Komponente „EXP“ bildet den Exponentialwert des Eintritts x auf den Ausgang y ab:

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

4.3.4 LN – Natürlicher Logarithmus

Symbol



Funktion

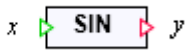
Die Komponente „LN“ bildet den natürlichen Logarithmus des Eintritts x auf den Ausgang y ab:

$$y = \ln(x) .$$

Das Argument x muss positiv sein. Falls beim Durchführen der Simulation das Argument kleiner oder gleich null wird, wird die Fehlermeldung „LN: invalid argument“ (Meldekategorie ERROR) erzeugt, und der Ausgang y wird auf null gesetzt. Das Argument x ist mit eins vorbesetzt.

4.3.5 SIN – Sinusfunktion

Symbol



Funktion

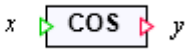
Die Komponente „SIN“ bildet den Sinuswert des Eintritts x auf den Ausgang y ab:

$$y = \sin(x) .$$

Die Maßeinheit für das Argument x (Winkel) kann über den Parameter „Unit“ im Bogenmaß (rad) oder im Gradmaß (deg) eingestellt werden. Voreingestellt ist das Gradmaß (deg) als Maßeinheit.

4.3.6 COS – Kosinusfunktion

Symbol



Funktion

Die Komponente „COS“ bildet den Cosinuswert des Eintritts x auf den Ausgang y ab:

$$y = \cos(x) .$$

Die Maßeinheit für das Argument x (Winkel) kann über den Parameter „Unit“ im Bogenmaß (rad) oder im Gradmaß (deg) eingestellt werden. Voreingestellt ist das Gradmaß (deg) als Maßeinheit.

4.3.7 TAN – Tangensfunktion

Symbol

Funktion

Die Komponente „TAN“ bildet den Tangenswert des Eintritts x auf den Ausgang y ab:

$$y = \tan(x) .$$

Die Maßeinheit für das Argument x (Winkel) kann über den Parameter „Unit“ im Bogenmaß (rad) oder im Gradmaß (deg) eingestellt werden. Voreingestellt ist das Gradmaß (deg) als Maßeinheit.

4.4 Binäre Funktionen

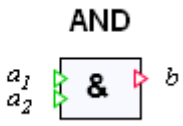
Alle Funktionen zur Verarbeitung von binären Signalen sind in der Ordnern „BinaryBasic“ und „BinaryExtended“ enthalten. Unter „BinaryBasic“ sind dabei Basisfunktionen eingeordnet, unter „BinaryExtended“ erweiterte Binärfunktionen.

4.4.1 Grundlegende binäre Funktionen

Die drei grundlegenden binären Operationen Konjunktion (AND), Disjunktion (OR) und Negation (NOT), sowie Äquivalenz (XNOR) und Antivalenz (XOR) sind als Komponententypen im Ordner „BinaryBasic“ abgelegt.

4.4.1.1 AND – Konjunktion

Symbol



Funktion

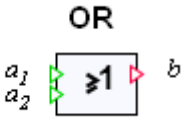
Der Komponententyp „AND“ bildet die n binären Werte an den Eingängen a_i als Konjunktion, also mit einer logischen (booleschen) „Und“-Funktion auf den Ausgang b ab:

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

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit eins vorbesetzt.

4.4.1.2 OR – Disjunktion

Symbol



Funktion

Der Komponententyp „OR“ bildet die n binären Werte an den Eingängen a_i als Disjunktion, also mit einer logischen (booleschen) „Oder“-Funktion auf den Ausgang b ab:

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

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbesetzt.

4.4.1.3 NOT, NOTc – Negation

Die Negation wird in zwei Komponenten „NOT“ und „NOTc“ zur Verfügung gestellt. Sie unterscheiden sich lediglich in der verwendeten Symbolik - ihre Funktion ist völlig identisch.

Symbol

Funktion

Der Ausgang b ist gleich dem negierten Eingang a, also

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

Wie in Abbildung 4-18 zu sehen ist, kann bei Verwendung der Komponente „NOTc“ die Negation übersichtlich und kompakt an den Ein- oder Ausgängen von Komponenten erfolgen.

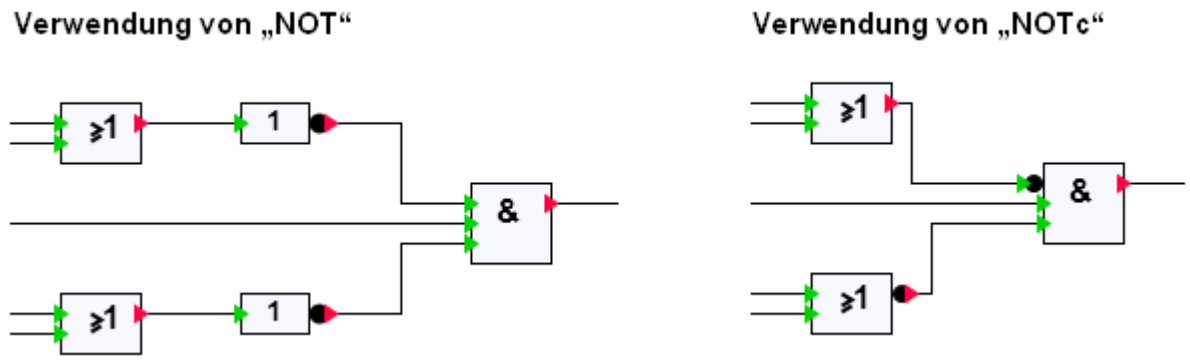
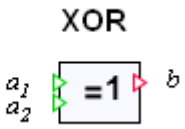


Abbildung 4-18: Verwendung der Komponententypen „NOT“ und „NOTc“

4.4.1.4 XOR – Antivalenz

Symbol



Funktion

Der Komponententyp „XOR“ bildet die n binären Werte an den Eingängen a_i als Antivalenz, also mit einer logischen (booleschen) „Exklusiv-Oder“-Funktion auf den Ausgang b ab. Für zwei Eingänge ist also

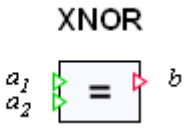
$$b = a_1 \otimes a_2 .$$

Der Ausgang b ist eins, wenn ausschließlich einer der Eingänge a_i gleich eins ist. In allen anderen Fällen nimmt der Ausgang den Wert null an.

Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbesetzt.

4.4.1.5 XNOR – Äquivalenz

Symbol



Funktion

Der Komponententyp „XNOR“ bildet die n binären Werte an den Eingängen a_i mit einer binären booleschen „Exklusiv-Nicht-Oder“-Funktion (Äquivalenz-Funktion) auf den Ausgang b ab. Für zwei Eingänge ist also

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

Als inverse „XOR“-Funktion ist der Ausgang b null, wenn ausschließlich einer der Eingänge a_i gleich eins ist. In allen anderen Fällen nimmt der Ausgang den Wert eins an.

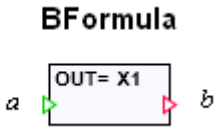
Die Anzahl der Eingänge n ist variabel und kann auf einen Wert zwischen 2 und 32 eingestellt werden. Alle Eingänge sind mit null vorbesetzt.

4.4.2 Erweiterte binäre Funktionen

Im Unterordner „BinaryExtended“ der Standardbibliothek sind weitere über die elementaren Binärfunktionen hinausgehende binäre (logische) Funktionen als Komponententypen abgelegt.

4.4.2.1 BFormula – binäre Formelkomponente

Symbol



Funktion

Der Komponententyp „BFormula“ ermöglicht die Verwendung von expliziten logischen Funktionen. Diese logische Funktion f berechnet einen Wert in Abhängigkeit von den n Werten an den Eingängen a_i. Der Funktionswert wird dem Ausgang b zugewiesen:

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

Zur Definition der Funktion tragen Sie im Parameter „Formula“ den gewünschten logischen Formelausdruck ein, der den Ausgangswert b in Abhängigkeit von den Eingangswerten a_i berechnet. Setzen Sie dazu zunächst die gewünschte Anzahl an Eingängen und öffnen Sie dann zur Eingabe des Formelausdrucks das Eigenschaftsfenster der Komponente (s. Abbildung 4-19).

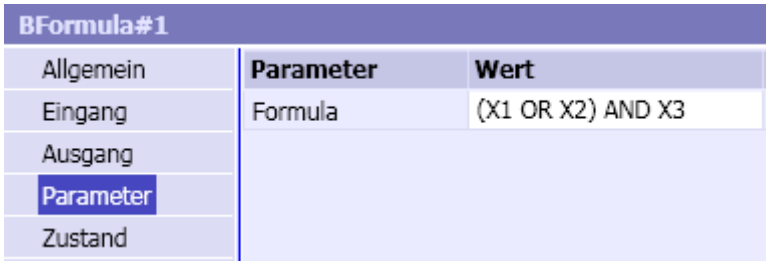


Abbildung 4-19: Eigenschaftsfenster des Komponententyps „BFormula“

Die Anzahl der Eingänge kann frei von 1 bis 32 variiert werden. Im Formelausdruck können nur die Eingänge benutzt werden, die gemäß der aktuell eingestellten Anzahl zur Verfügung stehen. Wie in Abbildung 4-20 gezeigt, wird der Formelausdruck im Symbol der Komponente auf dem Plan angezeigt.

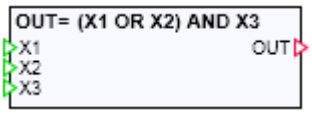


Abbildung 4-20: Komponente „BFormula“ mit drei Eingängen

Um längere Formelausdrücke vollständig sichtbar zu machen, können Sie die Komponente verbreitern. Bewegen Sie dazu den Mauszeiger über den rechten Rand der Komponente und ziehen Sie die Berandung bei gedrückter linker Maustaste auf die gewünschte Breite.

Die in Tabelle 4-3 gelisteten Operatoren sind in Formelausdrücken verwendbar.

Operator	Funktion
AND	Konjunktion (Und-Funktion)
OR	Disjunktion (Oder-Funktion)
NOT	Negation
(	Öffnende Klammer
)	Schließende Klammer

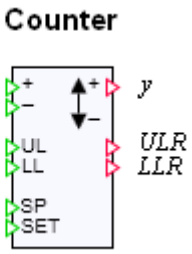
Tabelle 4-3: Zulässige Operatoren in Formelausdrücken des Komponententyps "BFormula"

HINWEIS

In der Formelkomponente wird nicht überprüft, ob im angegebenen Formelausdruck alle eingestellten Eingänge verwendet werden.

4.4.2.2 Counter – Auf- und Abwärtszähler

Symbol



Funktion

Der Komponententyp „Counter“ bietet die Möglichkeit, die Wechsel von Binärsignalen zu zählen. Mit einem Wechsel des Binärwerts am Eingang „+“ von null nach eins wird der Zählerwert am Ausgang y inkrementiert. Mit einem Wechsel des Binärwerts am Eingang „-“ von null nach eins wird der Zählwert am Ausgang y dekrementiert.

Die Werte, um die der Ausgang dekrementiert bzw. inkrementiert wird, können als Parameter („Decrement“ bzw. „Increment“) eingestellt werden. Beide Parameter sind on-line, d.h. bei laufender Simulation änderbar. Sie können jeden beliebigen Analogwert für das Dekrement und Inkrement einstellen. Beide Werte sind mit eins voreingestellt.

Der Zählerwert ist auf ein Intervall beschränkt das durch die beiden Grenzwerte „UL“ (obere Grenze) und „LL“ (untere Grenze) definiert ist:

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

Die binären Ausgänge ULR und LLR zeigen an, dass der Zählerwert die untere bzw. obere Grenze erreicht hat:

$$ULR = \begin{cases} 1, & \text{falls} \quad y \ge UL \\ 0, & \text{sonst} \end{cases} \qquad \text{und} \qquad LLR = \begin{cases} 1, & \text{falls} \quad y \le LL \\ 0, & \text{sonst} \end{cases}\ .$$

Der untere Grenzwert muss kleiner als der obere Grenzwert sein. Wird diese Bedingung verletzt, dann wird die Fehlermeldung „Counter: limits do not match“ (Meldekategorie ERROR) erzeugt und der Zählerwert am Ausgang y wird auf null gesetzt.

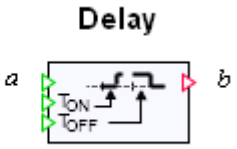
Über den binären Eingang „SET“ kann der Zählerwert y auf den Wert am Eingang „SP“ gesetzt werden: falls „SET“ auf eins gesetzt ist, wird der Zählerwert y gleich dem Wert am Eingang „SP“ gesetzt. Auch hier wird der Zählerwert durch „LL“ und „UL“ begrenzt.

Mit dem Parameter „Initial_Value“ wird der Zählerwert beim Initialisieren (Starten) der Simulation gesetzt. „Initial_Value“ ist mit null vorbelegt.

Die Grenzen sind mit null für den unteren Grenzwert und hundert für den oberen Grenzwert vorbelegt. Alle anderen Eingänge sind mit null vorbelegt.

4.4.2.3 Delay – Ein- und Ausschaltverzögerung

Symbol



Funktion

Beim „Delay“ wird das Binärsignal b am Ausgang dem Binärsignal a am Eingang verzögert nachgeführt.

Wenn am Eingang ein Signalwechsel von null nach eins stattfindet, wird der Ausgang nach Ablauf der Einschaltverzögerungszeit T_{ON} auf eins gesetzt. Wird das Eingangssignal vor Ablauf der Einschaltverzögerung wieder auf null gesetzt, dann bleibt das Ausgangssignal unverändert auf null. Erfolgt am Eingang ein Signalwechsel von eins nach null, dann wird der Ausgang nach Ablauf der Ausschaltverzögerungszeit T_{OFF} auf null gesetzt. Wird das Ausgangssignal vor Ablauf der Ausschaltverzögerung wieder auf eins gesetzt, dann bleibt das Ausgangssignal unverändert auf eins. Dieser Zusammenhang ist in Abbildung 4-21 veranschaulicht.

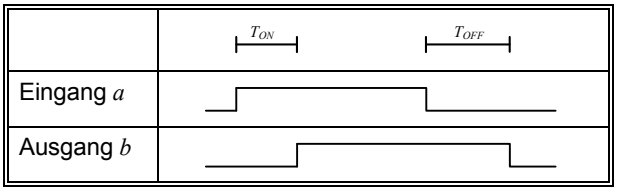
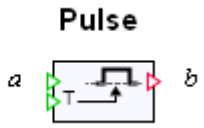


Abbildung 4-21: Signalverläufe am Ein- und Ausgang des Komponententyps „Delay“

Die Verzögerungszeiten dürfen keine negativen Werte annehmen. Nimmt eine Verzögerungszeit negative Werte an, dann wird die Fehlermeldung „Delay: on-delay time negativ value“ bzw. „Delay: off-delay time negativ value“ (Meldekategorie ERROR) erzeugt und die entsprechende Verzögerungszeit wird auf null gesetzt.

4.4.2.4 Pulse – Impuls

Symbol



Funktion

Der Komponententyp „Pulse“ setzt den Ausgang b, wenn am Eingang a ein Flankenwechsel von null nach eins stattfindet. Nach Ablauf der Zeit T wird der Ausgang b zurückgesetzt. Am Ausgang b wird somit ein Impuls mit der Pulsbreite T erzeugt (s. Abbildung 4-22). Die Pulsbreite ist über das Signal am Analogeingang T einstellbar.

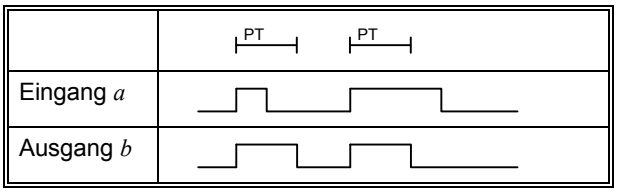


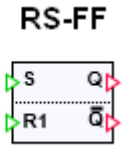
Abbildung 4-22: Signalverläufe am Ein- und Ausgang des Komponententyps „Pulse“

Die Pulsbreite T darf keine negativen Werte annehmen, andernfalls wird die Fehlermeldung „Puls: puls width negativ value“ (Meldekategorie ERROR) erzeugt und der Ausgang b wird auf null gesetzt. Für Pulsbreiten, die kleiner als die Abtastzeit der Simulation sind, wird der Ausgangsimpuls für die Dauer eines Simulationszyklus gesetzt.

Falls der on-line änderbare Parameter „Retriggerable“ eins gesetzt ist, wird der Ausgangsimpuls bei jedem Wechsel im Eingangssignal von null nach eins erneut gestartet. Der Parameter ist mit null vorbelegt.

4.4.2.5 RS_FF – Flipflop mit Vorzugslage „Rücksetzen“

Symbol



Funktion

Der Komponententyp „RS_FF“ stellt die einfachste Art eines Flipflops zur Verfügung: ein RS-Flipflop. Ist der Wert am Setzeingang S gleich eins, dann wird der Ausgangswert Q auf eins gesetzt. Ist der Eingangswert am Rücksetzeingang $R1$ auf eins gesetzt, dann wird der Ausgang auf null zurückgesetzt. Der Rücksetzeingang ist dominierend, d.h. wenn beide Eingänge auf eins gesetzt sind, wird der Ausgang Q auf null zurückgesetzt. Der Ausgang $\overline{Q}$ nimmt immer den zum Ausgang Q inversen Wert an.

In Tabelle 4-4 sind die möglichen Zustände des Komponententyps gelistet.

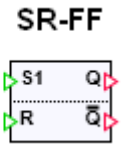
Eingang S	Eingang $R1$	Ausgang Q	Ausgang $\overline{Q}$
0	0	unverändert	unverändert
0	1	0	1
1	0	1	0
1	1	0	1

Tabelle 4-4: Zustandstabelle des Komponententyps „RS_FF“

Beim Start der Simulation wird der Ausgang Q auf den Wert des Parameters „Initial_State“ gesetzt. „Initial_State“ ist mit null vorbelegt.

4.4.2.6 SR_FF – Flipflop mit Vorzugslage „Setzen“

Symbol



Funktion

Die Komponente „SR_FF“ bildet ein SR-Flipflop nach. Ist der Wert am Setzeingang $S1$ gleich eins, dann wird der Ausgangswert Q auf eins gesetzt. Ist der Eingangswert am Rücksetzeingang R auf eins gesetzt, dann wird der Ausgang auf null zurückgesetzt. Der Setzeingang ist dominierend, d.h. wenn beide Eingänge auf eins gesetzt sind, wird der Ausgang Q auf eins gesetzt. Der Ausgang $\overline{Q}$ besitzt immer den zum Ausgang Q inversen Wert.

In Tabelle 4-5 sind die möglichen Zustände des Komponententyps gelistet.

Beim Start der Simulation wird der Ausgang Q auf den Wert des Parameters „Initial_State“ gesetzt. „Initial_State“ ist mit null vorbelegt.

Eingang S	Eingang $R1$	Ausgang Q	Ausgang $\overline{Q}$
0	0	unverändert	unverändert
0	1	0	1
1	0	1	0
1	1	1	0

Tabelle 4-5: Zustandstabelle des Komponententyps „SR_FF“

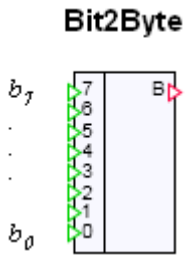
4.5 Konvertierung von Werten

Im Ordner „Conv“ der Standardbibliothek sind Komponententypen zur Konvertierung von Signalen eingeordnet:

- „Bit2Byte“ zur Konvertierung von Bits in einen Bytewert,
- „Byte2Bit“ zur Konvertierung von Bytes in Bit,
- „Byte2Word“ zur Konvertierung von Bytes in ein Wort,
- „Word2Byte“ zur Konvertierung eines Worts in Bytes,
- „Byte2DWord“ zur Konvertierung von Bytes in ein Doppelwort und
- „DWord2Byte“ zur Konvertierung eines Doppelworts in Bytes.

4.5.1 Bit2Byte – Konvertierung von Bit in Byte

Symbol



Funktion

Der Komponententyp „Bit2Byte“ konvertiert die binären Eingangswerte b_i , $i = 0, ..., 7,$ in einen Bytewert B am analogen Ausgang gemäß

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

Bei der Umwandlung stellt b_0 somit das niedrigstwertige Bit, b_7 das höchstwertige Bit dar.

Im Bedienfenster der Komponente können die einzelnen Bits b_i bei laufender Simulation individuell gesetzt werden. Zum Setzen eines Bits öffnen Sie bitte das Bedienfenster der Komponente. Wählen Sie jetzt die Bits aus, die Sie manuell setzen möchten, indem Sie den jeweils links angeordneten Schalter betätigen. Anschließend können Sie das jeweilige Bit durch Betätigen des rechten Schalters setzen und rücksetzen, d.h. zwischen null und eins umschalten. In Abbildung 4-23 sind die Bits b_0, b_3 und b_7 zum Setzen ausgewählt, Bit b_3 ist gesetzt.

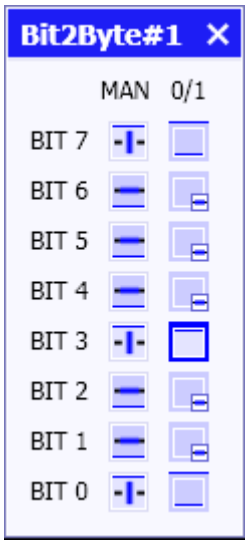
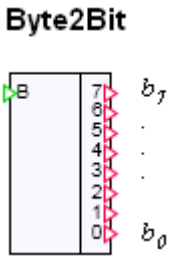


Abbildung 4-23: Bedienfenster des Komponententyps „Bit2Byte“

Nicht gesetzte Bits sind durch einen hellblauen Rahmen gekennzeichnet, gesetzte Bits durch einen dunkelblauen Rahmen.

4.5.2 Byte2Bit – Konvertierung von Byte in Bit

Symbol



Funktion

Der Komponententyp „Byte2Bit“ konvertiert einen Bytewert am analogen Eingang B in binäre Werte b_i , $i = 0, \ldots, 7,$ an den Ausgängen. Bei der Umwandlung wird b_0 das niedrigstwertige Bit, b_7 das höchstwertige Bit des Bytewerts zugewiesen.

Der Eingang wird auf den Wertebereich eines Bytes, also auf den Bereich null bis 255 begrenzt. Falls beim Durchführen der Simulation der Eingangswert nicht in diesem Bereich liegt, wird die Meldung „Byte2Bit: input not a valid byte value“ (Meldekategorie ERROR) erzeugt.

Im Bedienfenster der Komponente können die einzelnen Ausgangsbits b_i bei laufender Simulation individuell gesetzt werden. Zum Setzen eines Bits öffnen Sie bitte das Bedienfenster der Komponente. Wählen Sie jetzt die Bits aus, die Sie manuell setzen möchten, indem Sie den jeweils links angeordneten Schalter betätigen. Anschließend können Sie das jeweilige Bit durch Betätigen des rechten Schalters setzen und rücksetzen, d.h. zwischen null und eins umschalten. In Abbildung 4-24 sind die Bits b_0, b_4 und b_6 zum Setzen ausgewählt, Bit b_4 ist gesetzt:

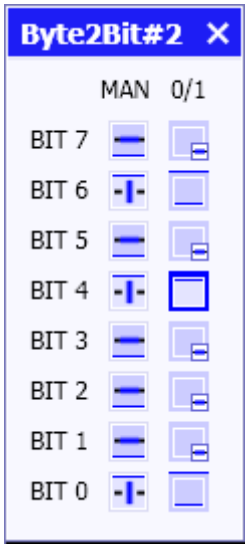
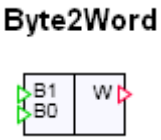


Abbildung 4-24: Bedienfenster des Komponententyps „Byte2Bit“

4.5.3 Byte2Word – Konvertierung von Byte in Wort

Symbol



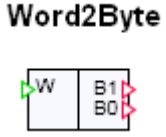
Funktion

Der Komponententyp „Byte2Word“ fasst zwei Bytewerte B_1, B_0 an den analogen Eingängen zu einem Wort am analogen Ausgang W zusammen. B_1 ist das höchstwertige Byte, B_0 das niedrigstwertige Byte.

Die Analogwerte am Eingang werden auf den Wertebereich eines Bytes, also auf den Bereich null bis 255 begrenzt. Falls beim Durchführen der Simulation ein Eingangswert nicht in diesem Bereich liegt, wird die Meldung „Byte2Word: input not a valid byte value“ (Meldekategorie ERROR) erzeugt.

4.5.4 Word2Byte – Konvertierung von Wort in Byte

Symbol



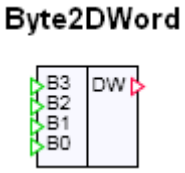
Funktion

Der Komponententyp „Word2Byte“ zerlegt ein Wort am analogen Eingang W in zwei Bytewerte B_1, B_0 an den analogen Ausgängen. B_1 ist das höchstwertige Byte, B_0 das niedrigstwertige Byte.

Der Analogwert am Eingang wird entsprechend dem Wertebereich eines Worts auf den Bereich null bis 2^{16}-1 begrenzt. Falls beim Durchführen der Simulation der Eingangswert nicht in diesem Bereich liegt, wird die Meldung „Word2Byte: input not a valid word value“ (Meldekategorie ERROR) erzeugt.

4.5.5 Byte2DWord – Konvertierung von Byte in Doppelwort

Symbol



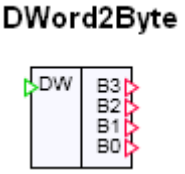
Funktion

Der Komponententyp „Byte2DWord“ fasst vier Bytewerte B_i, $i = 0, \ldots, 3$, an den analogen Eingängen zu einem Doppelwort in der Reihenfolge $B_3 - B_2 - B_1 - B_0$ am analogen Ausgang DW zusammen. B_3 ist somit das höchstwertige Byte, B_0 das niedrigstwertige Byte.

Die Analogwerte am Eingang werden auf den Wertebereich eines Bytes, also auf den Bereich null bis 255 begrenzt. Falls beim Durchführen der Simulation ein Eingangswert nicht in diesem Bereich liegt, wird die Meldung „Byte2DWord: input not a valid byte value“ (Meldekategorie ERROR) erzeugt.

4.5.6 DWord2Byte – Konvertierung von Doppelwort in Byte

Symbol



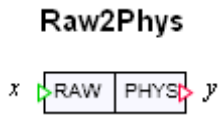
Funktion

Der Komponententyp „DWord2Byte“ zerlegt ein Doppelwort am analogen Eingang DW in vier Bytewerte B_i, $i = 0, \ldots, 3$, an den analogen Ausgängen. B_3 ist das höchstwertige Byte, B_0 das niedrigstwertige Byte.

Der Analogwert am Eingang wird entsprechend dem Wertebereich eines Doppelworts auf den Bereich null bis 2^{32}-1 begrenzt. Falls beim Durchführen der Simulation der Eingangswert nicht in diesem Bereich liegt, wird die Meldung „DWord2Byte: input not a valid double word value“ (Meldekategorie ERROR) erzeugt.

4.5.7 Raw2Phys – Konvertierung von Raw in Physical

Symbol



Funktion

Der Komponententyp „Raw2Phys“ überführt einen Analogwert x am Eingang in einen analogen Ausgangswert y gemäß der einfachen linearen Transformation

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

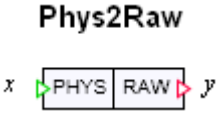
Der Eingangswert x kann beispielsweise ein sogenannter Rohwert eines Automatisierungssystems sein. Der Ausgangswert y ist dann der auf einen definierten physikalischen Wertebereich abgebildete Wert.

Die Transformationsintervalle werden durch die Parameter x_L (Raw_Lower_Limit), x_U (Raw_Upper_Limit), y_L (Phys_Lower_Limit) und y_U (Phys_Upper_Limit) eingestellt. Sie können zur Laufzeit einer Simulation geändert werden und sie sind wie folgt vorbesetzt:

- Raw_Upper_Limit: 27648
- Raw_Lower_Limit: -27648
- Phys_Upper_Limit: 100
- Phys_Lower_Limit: 0

4.5.8 Phys2Raw – Konvertierung von Physical in Raw

Symbol



Funktion

Der Komponententyp „Phys2Raw“ überführt einen Analogwert x am Eingang in einen analogen Ausgangswert y gemäß der einfachen linearen Transformation

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

Der Eingangswert x wird damit beispielsweise als Messwert einer physikalischen Größe in den sogenannten Rohwert y eines Automatisierungssystems transformiert.

Die Transformationsintervalle werden durch die Parameter x_L (Raw_Lower_Limit), x_U (Raw_Upper_Limit), y_L (Phys_Lower_Limit) und y_U (Phys_Upper_Limit) eingestellt. Sie können zur Laufzeit einer Simulation geändert werden und sie sind wie folgt vorbesetzt:

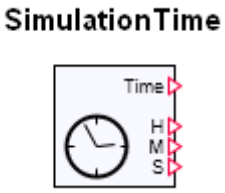
- Raw_Upper_Limit: 27648
- Raw_Lower_Limit: -27648
- Phys_Upper_Limit: 100

- Phys_Lower_Limit: 0

4.6 Allgemeine Komponenten im Ordner „Misc“

4.6.1 SimulationTime – Simulationszeit

Symbol



Funktion

Der Komponententyp „SimulationTime“ gibt an seinen vier Ausgängen die aktuelle Simulationszeit aus. Die Ausgabe erfolgt am Ausgang „Time“ als Wert in Millisekunden. An den Ausgängen „H“, „M“ und „S“ steht die Simulationszeit aufgelöst in Stunden (H), Minuten (M) und Sekunden (S) zur Verfügung.

Die Simulationszeit ist die Zeit, die beim Durchführen der Simulation im Zyklus der Taktung der Simulation gezählt wird. Beim Initialisieren bzw. Starten einer Simulation wird die Simulationszeit auf null gesetzt. Simulationszeit und Realzeit laufen synchron, wenn die Simulation im Echtzeitmodus läuft. Im verlangsamten Simulationsmodus (Zeitlupe) läuft die Simulationszeit langsamer als die Realzeit und im beschleunigten Simulationsmodus (Zeitraffer) läuft die Simulationszeit schneller als die Realzeit. Wird die Simulation angehalten, dann wird auch die Simulationszeit angehalten. Wird die Simulation dann wieder fortgesetzt, dann läuft auch die Simulationszeit weiter.

Die Simulationszeit wird auch im Bedienfenster (Abbildung 4-25) der Komponente angezeigt.

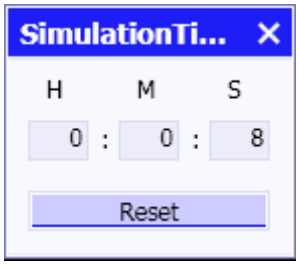


Abbildung 4-25: Bedienfenster des Komponententyps „SimulationTime“

Durch Betätigen der Schaltfläche „Reset“ im Bedienfenster der Komponente können Sie die Simulationszeit auf null zurücksetzen.

5 ANTRIEBSKOMPONENTEN

Im Ordner „DRIVES“ dieser Bibliothek sind Komponententypen von Antrieben enthalten. Es lassen sich damit allgemeine Antriebe für Ventile und Pumpen simulieren. Die Typen

- „DriveP1“ und „DriveP2“ sind für die Simulation von Pumpenantrieben, die Typen
- „DriveV1“ bis „DriveV4“ sind für die Simulation von Ventilantrieben konzipiert.

5.1 Ventilantriebe

Es stehen vier Komponententypen „DriveV1“, „DriveV2“, „DriveV3“ und „DriveV4“ zur Verfügung, die in der Simulation als Nachbildung von Ventilantrieben eingesetzt werden können. Den Komponententypen „DriveV1“ bis „DriveV4“ gemeinsam sind die Ausgänge und ein Teil der Eingänge wie in Abbildung 5-1 dargestellt.

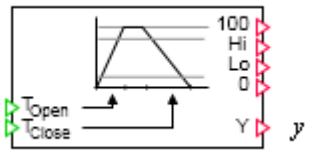


Abbildung 5-1: Gemeinsame Anschlüsse der Komponententypen für Ventilantriebe

An den beiden Analogeingängen T_{Open} und T_{Close} werden die Öffnungs- bzw. Schließzeiten des Ventils, d.h. die Laufzeit des Antriebs zum vollständigen Öffnen bzw. Schließen, vorgegeben. Falls beim Durchführen der Simulation einer der beiden Eingangswerte negativ ist, wird die Meldung „DriveVx: closing or opening time invalid value“ (Meldekategorie ERROR) erzeugt.

Am Analogausgang y wird der aktuelle Positionswert des Antriebs als Prozentwert, d.h. im Bereich null bis hundert ausgegeben:

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

Der Wert null entspricht dem vollständig geschlossenen Ventil, der Wert hundert dem vollständig geöffneten Ventil.

Die vier Binärausgänge „100“, „Hi“, „Lo“ und „0“ können als Endschalter des Antriebs interpretiert werden:

- „100“ und „0“ als Endschalter „Ventil vollständig auf“ bzw. „Ventil vollständig zu“,
- „Hi“ und „Lo“ als Vorendschalter „Ventil auf“ bzw. „Ventil zu“.

Die Endschalter sind dabei fest auf die Positionswerte null für das vollständig geschlossene Ventil und hundert für das vollständig geöffnete Ventil eingestellt. Die Binärausgänge „100“ bzw. „0“ werden demzufolge auf eins gesetzt, wenn das Ventil vollständig geschlossen bzw. geöffnet ist, also wenn die Position y der Komponente den Wert null bzw. hundert hat.

Die Vorendschalter „Hi“ und „Lo“ sind über die Parameter „HI_Limit“ bzw. „LO_Limit“ einstellbar. Ihr Wert sollte zwischen null und hundert liegen. Voreingestellt sind Positionswerte von fünf („LO_Limit“) und 95 („HI_Limit“).

Falls beim Durchführen der Simulation einer der beiden Parameterwerte negativ ist, wird die Meldung „DriveVx: high and low parameters do not match“ (Meldekategorie ERROR) erzeugt.

Über den Parameter „Initial_Value“ kann der Ventilantrieb auf die Anfangsposition „Closed“ oder „Open“ eingestellt werden. Die Vorbelegung für „Initial-Value“ ist „Closed“.

Im Bedienfenster (Abbildung 5-2) der Komponenten ist der aktuelle Positionswert des Antriebs in Form einer Balkenanzeige visualisiert. Über den linken mit „MAN“ gekennzeichneten Schalter kann der Positionswert auf Vorgabe durch den ebenfalls im Bedienfenster verfügbaren Schieber umgeschaltet werden.

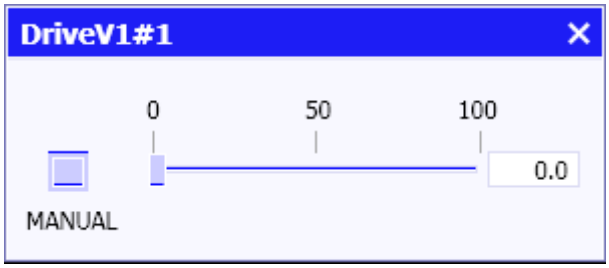
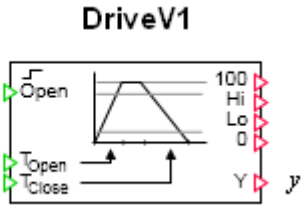


Abbildung 5-2: Bedienfenster der Komponententypen für Ventilantriebe

Die Bildung des Positionswerts wird dann nicht mehr aus den Eingängen der Komponente abgeleitet, sondern dem über den Schieber vorgegebenen Wert nachgeführt. Die vorgegebenen Öffnungs- und Schließzeiten bleiben dabei wirksam. Ebenfalls wirksam bleiben die vier Binärausgänge „100“, „Hi“, „Lo“ und „0“. Sie werden auch im Fall der Nachführung des Positionswertes wie oben beschrieben gesetzt.

5.1.1 DriveV1 – Ventilantrieb vom Typ 1

Symbol

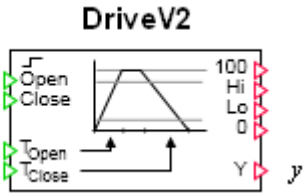


Funktion

Der Komponententyp „DriveV1“ bildet eine Antriebseinheit nach, die den Positionswert y am Ausgang in Abhängigkeit von einem Binärwert am Eingang „Open“ setzt. Ist der Binäreingang gleich null, dann wird der Positionswert y kontinuierlich mit der Schließzeit T_{Close} nach null verfahren. Ist der Binäreingang gleich eins, dann wird der Positionswert kontinuierlich mit der Öffnungszeit T_{Open} nach hundert verfahren. Der Positionswert hat damit nur die beiden stabilen Zustände null und hundert, d.h. bezogen auf die Ventilfunktion die Zustände „Ventil auf“ und „Ventil zu“. Mit jedem Wechsel des Binärwerts am Eingang wir der Antrieb bzw. das Ventil somit „aufgefahren“ oder „zugefahren“.

5.1.2 DriveV2 – Ventilantrieb vom Typ 2

Symbol



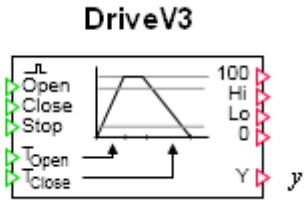
Funktion

Der Komponententyp „DriveV2“ bildet eine Antriebseinheit nach, die den Positionswert y am Ausgang in Abhängigkeit von zwei Binärwerten an den Eingängen „Open“ und „Close“ setzt. Ist der Binärwert am Eingang „Open“ gleich eins, dann wird der Positionswert y kontinuierlich mit der Öffnungszeit T_{Open} nach hundert verfahren. Ist der Binärwert am Eingang „Close“ gleich eins, dann wird der Positionswert kontinuierlich mit der Öffnungszeit T_{Close} nach null verfahren.

Sind die Werte an beiden Eingängen gleichzeitig auf eins oder null gesetzt, dann bleibt der Positionswert unverändert. Der Positionswert ändert sich damit nur, wenn entweder der Binärwert am Eingang „Open“ oder am Eingang „Close“ auf eins gesetzt ist.

5.1.3 DriveV3 – Ventilantrieb vom Typ 3

Symbol

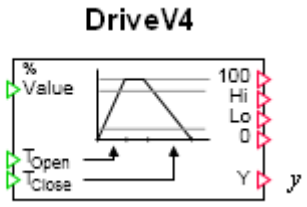


Funktion

Der Komponententyp „DriveV3“ bildet eine Antriebseinheit nach, die den Positionswert y am Ausgang in Abhängigkeit von Binärwerten an den drei Eingängen „Open“, „Close“ und „Stop“ setzt. Ändert der Binärwert am Eingan „Open“ seinen Wert von null nach eins, dann wird der Positionswert y kontinuierlich mit der Öffnungszeit T_{Open} nach hundert verfahren. Der Positionswert wird kontinuierlich mit der Schließzeit T_{Close} nach null verfahren wenn der Binärwert am Eingang „Close“ seinen Wert von null nach eins ändert. Ändert der Binärwert am Eingang „Stop“ seinen Wert von null nach eins, dann bleibt der Positionswert unverändert. Das „Öffnen“, „Schließen“ und „Stoppen“ des Antriebs wird damit über die Flanke des entsprechenden binären Eingangssignals (Signalwechsel von null nach eins) initiiert.

5.1.4 DriveV4 – Ventilantrieb vom Typ 4

Symbol



Funktion

Der Komponententyp „DriveV4“ bildet eine Antriebseinheit nach, die den Positionswert y am Ausgang dem Analogwert am Eingang „Value“ kontinuierlich nachführt. Die Nachführung erfolgt in Richtung steigender Positionswerte mit der Öffnungszeit T_{Open} und in Richtung fallender Positionswerte mit der Schließzeit T_{Close}. Der Positionswert bleibt in jedem Fall auf Werte von null bis hundert beschränkt, auch wenn der Eingangswert außerhalb dieses Intervalls liegt.

5.2 Pumpen-, Lüfterantriebe

Die beiden Komponententypen „DriveP1“ und „DriveP2“ können in der Simulation als Antriebe für Pumpen, Lüfter oder vergleichbare Aggregate eingesetzt werden. Den beiden Komponententypen gemeinsam sind die Ausgänge und ein Teil der Eingänge wie in Abbildung 5-3 dargestellt.

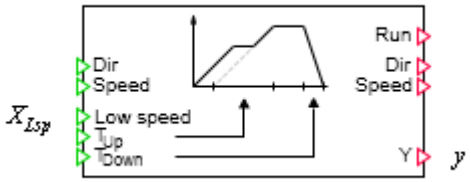


Abbildung 5-3: Gemeinsame Anschlüsse der Komponententypen für Pumpenantriebe

An den beiden Analogeingängen T_{Up} und T_{Down} werden die Hoch- bzw. Runterlaufzeiten des Antriebs vorgegeben. T_{Up} ist die Laufzeit des Antriebs zum Hochfahren vom Stillstand auf die Nenndrehzahl in Sekunden, T_{Down} ist die Zeit zum Abfahren des Antriebs von Nenndrehzahl bis zum Stillstand in Sekunden. Voreingestellt sind beide Zeiten mit einer Sekunde. Falls beim Durchführen der Simulation einer der beiden Eingangswerte negativ ist, wird die Meldung „DriveVx: run-up or run-down time invalid value“ (Meldekategorie ERROR) erzeugt.

Am Analogausgang y wird der aktuelle Drehzahlwert des Antriebs als Prozentwert, d.h. im Bereich null bis hundert ausgegeben:

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

Der Wert null entspricht dem stillstehenden Antrieb, der Wert hundert dem Antrieb auf Nenndrehzahl.

Über den Binäreingang „Dir“ ist die Drehrichtung des Antriebs vorgebbar. Ist dieser Eingang auf null gesetzt, dann ist die positive Drehrichtung vorgegeben. Ist der Eingang auf eins gesetzt, dann ist die Drehrichtung negativ vorgegeben. Es kann über diesen Eingang also

beispielsweise Rechts- und Linkslauf des Antriebs vorgegeben werden. Bei Drehung in negativer Richtung wird die Drehzahl am Ausgang y als negativer Wert ausgegeben:

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

Positive Werte von y kennzeichnen somit Drehzahlen in positive Richtung, negative Werte Drehzahlen in negative Richtung. Die Drehrichtungsumschaltung wird nur wirksam, wenn der Antrieb steht.

Über den Binäreingang „Speed“ kann die Drehzahl zwischen Nenndrehzahl (Volldrehzahl) und Teildrehzahl umgeschaltet werden. Falls „Speed“ auf eins gesetzt ist, ist Nenndrehzahl angewählt, andernfalls ist Teildrehzahl angewählt. Voreinstellung für den Binäreingang „Speed“ ist eins. Die Teildrehzahl wird als Zahlenwert (Prozentwert) am Analogeingang x_{LSp} vorgegeben:

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

Die Voreinstellung für die Teildrehzahl ist 50, also halbe Nenndrehzahl. Falls die Teildrehzahl nicht im definierten Bereich liegt, wird die Meldung „DrivePx: low speed invalid value“ (Meldekategorie ERROR) erzeugt.

Der Binärausgang „Run“ ist nur dann auf eins gesetzt, wenn der Antrieb den vorgewählten Drehzahlwert in positive oder negative Drehrichtung erreicht hat, also nur wenn der Absolutwert am Analogausgang y gleich hundert (Nenndrehzahl) oder gleich der am Eingang x_{LSp} vorgegeben Teildrehzahl ist.

Am Binärausgang „Speed“ steht das Rückmeldesignal für die Drehzahlanwahl zur Verfügung. Der Binärausgang wird nur auf eins gesetzt, wenn der Antrieb seine Nenndrehzahl in positive oder negative Drehrichtung erreicht hat.

Der Wert am Binärausgang „Dir“ wird als Rückmeldesignal der aktuellen Drehrichtung des Antriebs gebildet. Der Binärausgang ist null, wenn der Antrieb in positive Richtung dreht; der Wert ist eins, wenn der Antrieb in negative Richtung dreht.

Im Bedienfenster (Abbildung 5-4) der Komponenten ist der aktuelle Drehzahlwert des Antriebs in Form einer Balkenanzeige visualisiert. Über den linken mit „MAN“ gekennzeichneten Schalter kann der Drehzahlwert auf Vorgabe durch den ebenfalls im Bedienfenster verfügbaren Schieber umgeschaltet werden.

Die Bildung des Drehzahlwerts wird dann nicht mehr aus den Eingängen der Komponente abgeleitet, sondern dem über den Schieber vorgegebenen Wert nachgeführt. Die vorgegebenen Hoch- und Runterlaufzeiten bleiben dabei wirksam. Ebenfalls wirksam bleiben die Binärausgänge „Run“, „Speed“ und „Dir“. Sie werden auch im Fall der Nachführung des Drehzahlwertes wie oben beschrieben gesetzt.

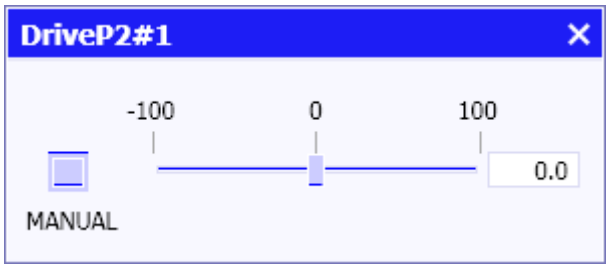
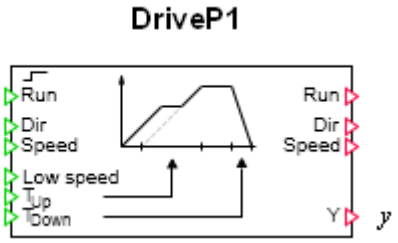


Abbildung 5-4: Bedienfenster der Komponententypen für Pumpenantriebe

5.2.1 DriveP1 – Pumpenantrieb vom Typ 1

Symbol

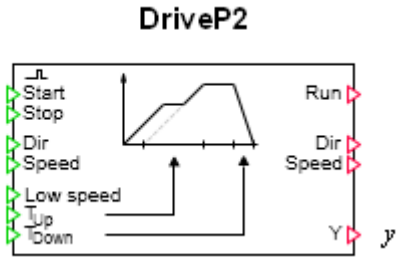


Funktion

Der Komponententyp „DriveP1“ bildet eine Antriebseinheit nach, die über den Binärwert am Eingang „Run“ ein- und ausgeschaltet wird. Solange der Eingangswert auf eins gesetzt ist, ist der Antrieb eingeschaltet. Wird der Eingangswert auf null gesetzt, dann wird der Antrieb ausgeschaltet.

5.2.2 DriveP2 – Pumpenantrieb vom Typ 2

Symbol



Funktion

Der Komponententyp „DriveP2“ bildet eine Antriebseinheit nach, die über die beiden Binäreingänge „Start“ und „Stop“ ein- und ausgeschaltet wird. Ändert sich der Binärwert am Eingang „Start“ von null nach eins, dann wird der Antrieb eingeschaltet. Bei einer Änderung des Binärwerts am Eingang „Stop“ von null nach eins wird der Antrieb ausgeschaltet. Das Ein- und Ausschalten des Antriebs wird damit über die Flanke des entsprechenden binären Eingangssignals (Signalwechsel von null nach eins) initiert.