

Funktionsblockbibliothek

Die mit Robotino® View erstellten Steuerungsprogramme bestehen aus miteinander vernetzten Funktionsblöcken. Diese Funktionsblöcke befinden sich in der [Funktionsblockbibliothek](#) und können per Drag&Drop in ein Programm eingefügt werden.

Funktionsblöcke sind verschiedenen Klassen zugeordnet. Durch Anklicken mit der linken Maustaste des Klassennamens werden die zu dieser Klasse gehörenden Funktionsblöcke angezeigt. Folgende Klassen sind verfügbar:

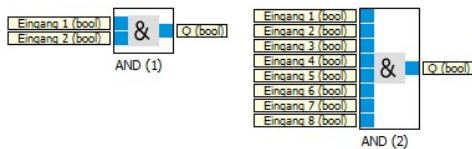
Name	Beschreibung
Logik	Von Elektronik-Logikbausteinen bekannte Komponenten
Mathematik	Mathematische Operationen und Funktionen
Vektorrechnung	Rechnen mit 2-dimensionalen Vektoren
Anzeige	Funktionsblöcke zur Visualisierung
Bildverarbeitung	Funktionsblöcke zum Arbeiten mit (Kamera-)Bildern
Generatoren	Erzeugung von Signalen
Filter	Glättung von Signalen
Navigation	Funktionsblöcke zum Bearbeiten und Befahren von Wegen
Eingabegeräte	Funktionsblöcke zur Interaktion des Benutzers mit dem Steuerungsprogramm
Datenaustausch	Funktionsblöcke zum Austausch von Daten mit externen Programmen
Eigene Funktionsblöcke	Tutorials zur Entwicklung eigener Funktionsblöcke

Logik

Die Klasse Logik enthält von Elektronik-Logikbausteinen bekannte Komponenten.

Logische Operationen

AND



Der Ausgang des AND nimmt nur dann den Zustand wahr (true) an, wenn alle Eingänge den Zustand wahr haben. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	true	
...			
Eingang 8	bool	true	
Ausgänge			
Q	bool		siehe Wahrheitstabelle

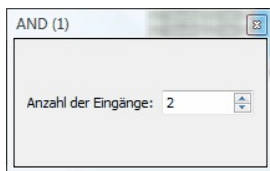
Eingänge								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	0
							1	0
						1		0
						1	1	0
					1			0
					1		1	0
					1	1		0
					1	1	1	0

				1				0
				1			1	0
				1		1		0
				1		1	1	0
				1	1			0
				1	1		1	0
				1	1	1		0
				1	1	1	1	0
			1					0
1	1	1	1	1	1	1	1	1

FESTO



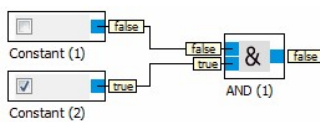
Dialog



FESTO



Beispiel



OR



or

Der Ausgang des OR nimmt dann den Zustand wahr (true) an, wenn mindestens ein Eingang den Zustand wahr hat. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	false	
...			
Eingang 8	bool	false	
Ausgänge			
Q	bool		siehe Wahrheitstabelle

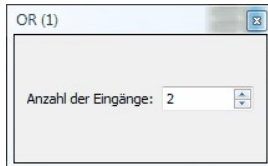
Eingänge								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	0
							1	1
						1		1
						1	1	1
					1			1
					1		1	1
					1	1		1
					1	1	1	1
				1				1
				1			1	1

				1		1		1
				1		1	1	1
				1	1			1
				1	1		1	1
				1	1	1		1
				1	1	1	1	1
			1					1
1	1	1	1	1	1	1	1	1

FESTO



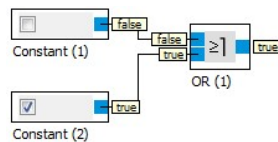
Dialog



FESTO



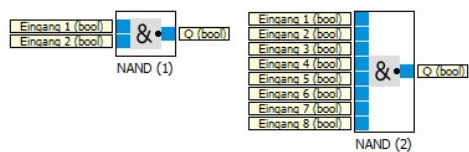
Beispiel



FESTO



NAND



Der Ausgang des NAND nimmt nur dann den Zustand unwahr (false) an, wenn alle Eingänge den Zustand wahr (true) haben. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	true	
...			
Eingang 8	bool	true	
Ausgänge			
Q	bool		siehe Wahrheitstabelle

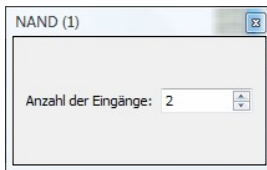
Eingänge								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	1
							1	1
						1		1
						1	1	1
					1			1
					1		1	1
					1	1		1
					1	1	1	1
			1					1
			1				1	1
			1			1		1
			1			1	1	1

				1	1			1
				1	1		1	1
				1	1	1		1
				1	1	1	1	1
			1					1
1	1	1	1	1	1	1	1	0

FESTO



Dialog



FESTO



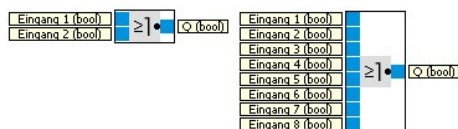
Beispiel

siehe Beispiele ► Logik ► [FlipFlop](#)

FESTO



NOR



Der Ausgang des NOR nimmt nur dann den Zustand wahr (true) an, wenn alle Eingänge den Zustand unwahr (false) haben. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	false	
...			
Eingang 8	bool	false	
Ausgänge			
Q	bool		siehe Wahrheitstabelle

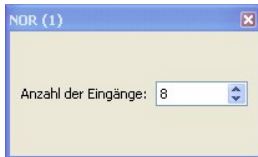
Eingänge								
1	2	3	4	5	6	7	8	Q
0	0	0	0	0	0	0	0	1
							1	0
						1		0
						1	1	0
					1			0
					1		1	0
					1	1		0
					1	1	1	0
				1				0
				1			1	0
				1		1		0
				1		1	1	0
				1	1			0
				1	1		1	0
				1	1	1		0
				1	1	1	1	0
			1					0

1	1	1	1	1	1	1	1	1	0

FESTO



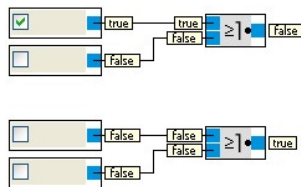
Dialog



FESTO



Beispiel



FESTO



XOR



Der Ausgang des XOR nimmt den Zustand wahr (true) an, wenn die Eingänge unterschiedliche Zustände besitzen. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

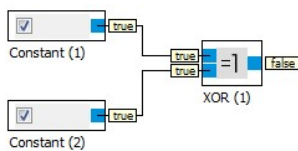
Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	false	
Eingang 2	bool	false	
Ausgänge			
Q	bool		siehe Wahrheitstabelle

Eingänge		
1	2	Q
0	0	0
0	1	1
1	0	1
1	1	0

FESTO



Beispiel



FESTO



NOT



Der Ausgang des NOT nimmt den Zustand wahr (true) an, wenn der Eingang den Zustand unwahr (false) hat. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

--	--	--	--

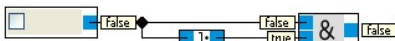
Eingänge	Typ	Standard	Beschreibung
Eingang	bool	false	
Ausgänge			
Q	bool		siehe Wahrheitstabelle

Eingänge	
1	Q
0	1
1	0

FESTO



Beispiel



Das Beispiel zeigt eine Besonderheit des NOT Funktionsblocks. Daten werden nicht wie bei anderen Funktionsblöcken neben den Anschlüssen angezeigt. Dadurch kann das NOT auf sehr engem Raum neben anderen Funktionsblöcken verwendet werden, ohne dass sich die Datenanzeigen überlagern und verdecken.

FESTO

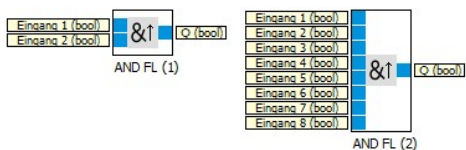


Getriggerte Operationen

FESTO



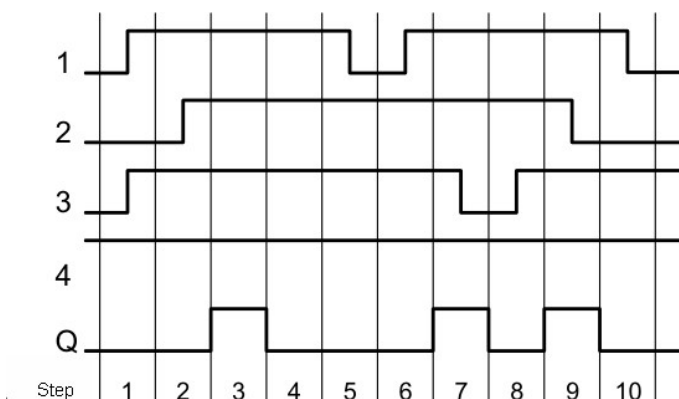
AND FL



Der Ausgang des AND mit Flankenbewertung nimmt nur dann den Zustand wahr (true) an, wenn alle Eingänge den Zustand wahr haben und im vorherigen Zyklus mindestens ein Eingang den Zustand unwahr (false) hatte. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	true	
...			
Eingang 8	bool	true	
Ausgänge			
Q	bool		Timingdiagramm

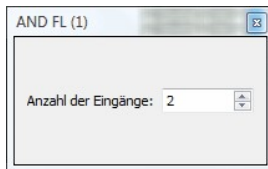
Timingdiagramm für das AND mit Flankenbewertung mit 4 Eingängen.



FESTO



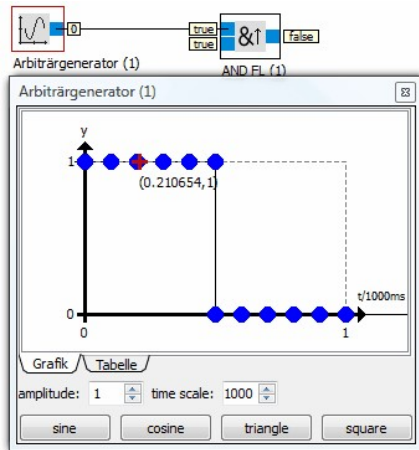
Dialog



FESTO



Beispiel

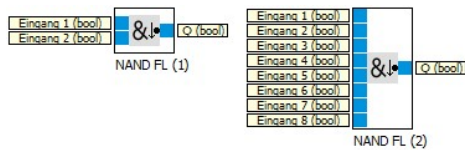


Bei jedem Sprung des Generators von 0 nach 1 ist der Ausgang des AND mit Flankenbewertung für einen Takt wahr (true).

FESTO



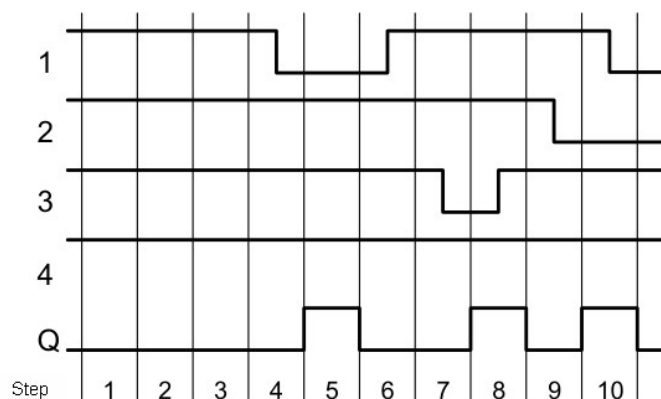
NAND FL



Der Ausgang des NAND mit Flankenbewertung nimmt nur dann den Zustand wahr (true) an, wenn alle Eingänge den Zustand unwahr (false) haben und im vorherigen Zyklus alle Eingänge den Zustand wahr (true) hatten. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

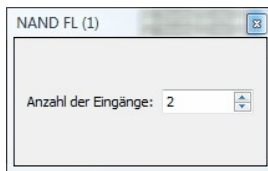
Eingänge	Typ	Standard	Beschreibung
Eingang 1	bool	true	
...			
Eingang 8	bool	true	
Ausgänge			
Q	bool		siehe Timingdiagramm

Timingdiagramm für das NAND mit Flankenbewertung mit 4 Eingängen.



FESTO

Dialog



FESTO



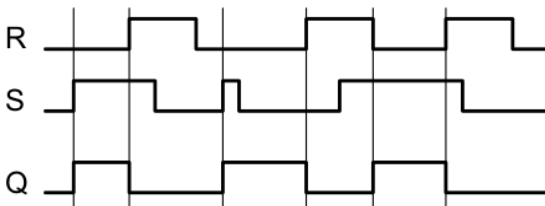
Selbsthalterelais



Über den Eingang S wird der Ausgang Q gesetzt. Über den Eingang R wird der Ausgang wieder zurückgesetzt. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
S	bool	false	Über den Eingang S setzen Sie den Ausgang Q auf wahr (true).
R	bool	false	Über den Eingang R setzen Sie den Ausgang Q auf 0 zurück. Wenn S und R gleichzeitig 1 sind, dann wird zurückgesetzt.
Par	bool	false	Remanenz: unwahr (false): keine Remanenz wahr (true): Der aktuelle Zustand wird remanent (unabhängig von S oder R) gespeichert
Ausgänge			
Q	bool		Q schaltet mit S auf wahr (true) und bleibt wahr bis Eingang R auf wahr (true) gesetzt wird.

Timingdiagramm



FESTO



Abtast-Halte-Glied



Wird Abtasten auf false gesetzt, kann das Signal am Eingang beliebig lang auf dem momentanen Wert gehalten werden. Für die Konvertierung von Zahlwerten nach bool siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Eingang	float	0	Eingangssignal
Abtasten	bool	false	Wenn true, wird das Signal am Eingang über den Ausgang ausgegeben. Wenn false, wird der aktuelle Wert am Ausgang "eingefroren".
Ausgänge			
Ausgang	float	0	Wert des Eingangssignal, als Abtasten von true auf false gewechselt hat.

FESTO



Bitweise Operationen

FESTO



Bitweises Und

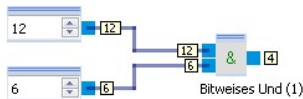


Dieser Funktionsblock führt eine bitweise Und-Verknüpfung zweier Ganzzahlen aus. Im Ergebnis sind nur die Bits gleich 1, die in beiden Operanden gleich 1 sind.

Eingänge	Typ	Standard	Beschreibung
Operand 1	int	0	
Operand 2	int	0	
Ausgänge			
Ergebnis	int		Ergebnis der bitweisen Und-Verknüpfung von Operand 1 und Operand 2



Beispiel



	Wert	Binärdarstellung
Operand 1	12	1100
Operand 2	6	0110
Ergebnis	4	0100



Bitweises Exklusiv-Oder

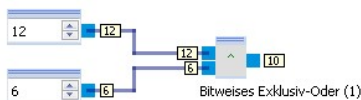


Dieser Funktionsblock führt eine bitweise Exklusiv-Oder-Verknüpfung zweier Ganzzahlen aus. Im Ergebnis sind nur die Bits gleich 1, in denen sich die Operanden unterscheiden.

Eingänge	Typ	Standard	Beschreibung
Operand 1	int	0	
Operand 2	int	0	
Ausgänge			
Ergebnis	int		Ergebnis der bitweisen Exklusiv-Oder-Verknüpfung von Operand 1 und Operand 2



Beispiel



	Wert	Binärdarstellung
Operand 1	12	1100
Operand 2	6	0110
Ergebnis	10	1010



Bitweises NOT



Dieser Funktionsblock führt eine bitweise Negation einer Ganzzahl aus.

--	--	--	--

Eingänge	Typ	Standard	Beschreibung
Operand	int	0	
Ausgänge			
Ergebnis	int		Ergebnis der bitweisen Negation (Einerkomplement) des Operanden

FESTO



Beispiel



	Wert	Binärdarstellung
Operand	6	00110
Ergebnis	-7	11001

Da mit vorzeichenbehafteten Zahlen gerechnet wird, wird das höchstwertige Bit als Vorzeichenbit interpretiert.

FESTO



Bitweises Oder



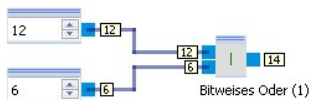
Dieser Funktionsblock führt eine bitweise Oder-Verknüpfung zweier Ganzzahlen aus. Im Ergebnis sind die Bits gleich 1, die in einem der beiden Operanden oder in beiden gleich 1 sind.

Eingänge	Typ	Standard	Beschreibung
Operand 1	int	0	
Operand 2	int	0	
Ausgänge			
Ergebnis	int		Ergebnis der bitweisen Oder-Verknüpfung von Operand 1 und Operand 2

FESTO



Beispiel



	Wert	Binärdarstellung
Operand 1	12	1100
Operand 2	6	0110
Ergebnis	14	1110

FESTO



Bitweises Linksschieben

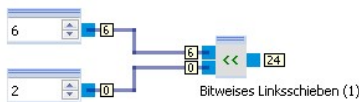


Dieser Funktionsblock führt eine bitweise Verschiebung der Eingabe nach links durch.

Eingänge	Typ	Standard	Beschreibung
Eingabe	int	0	
Verschiebung	int	0	Anzahl der Stellen, um die die Eingabe verschoben werden soll
Ausgänge			
Ergebnis	int		Ergebnis der Schiebeoperation

FESTO

Beispiel



Die Eingabe wird bitweise um 2 Stellen nach links geschoben.

	Wert	Binärdarstellung
Eingabe	6	00110
Ergebnis	24	11000

FESTO

Bitweises Rechtsschieben

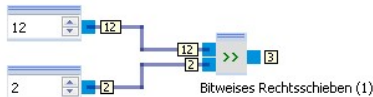


Dieser Funktionsblock führt eine bitweise Verschiebung der Eingabe nach rechts durch.

Eingänge	Typ	Standard	Beschreibung
Eingabe	int	0	
Verschiebung	int	0	Anzahl der Stellen, um die die Eingabe verschoben werden soll
Ausgänge			
Ergebnis	int		Ergebnis der Schiebeoperation

FESTO

Beispiel



Die Eingabe wird bitweise um 2 Stellen nach rechts geschoben.

	Wert	Binärdarstellung
Eingabe	12	1100
Ergebnis	3	0011

FESTO

Bitweiser Test



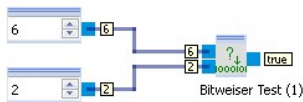
Dieser Funktionsblock testet, ob ein bestimmtes Bit der Eingabe gesetzt ist.

Eingänge	Typ	Standard	Beschreibung
Eingabe	int	0	
Bit	int	0	Zu testendes Bit der Eingabe
Ausgänge			
Ergebnis	bool		true, wenn das Bit gesetzt ist, ansonsten false

FESTO

Beispiel





FESTO



Zähler aufwärts



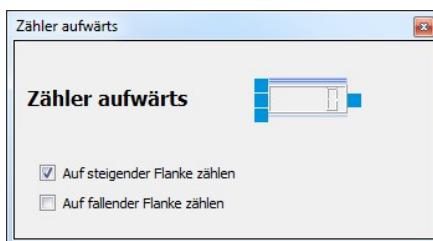
Der Zähler zählt die Anzahl der Änderungen seines Eingangs.

Eingänge	Typ	Standard	Beschreibung
Eingang	bool	false	Zählereingang. Gezählt wird beim Übergang von unwahr (false) nach wahr (true) und/oder beim Übergang von wahr nach unwahr.
Anfangswert	int	0	Die Zählung beginnt bei diesem Wert bei Programmstart und wenn Zurücksetzen wahr (true) ist.
Zurücksetzen	bool	false	Wenn wahr (true), wird der Zähler auf den Anfangswert gesetzt.
Ausgänge			
Ausgang	int		Zählerwert

FESTO



Dialog

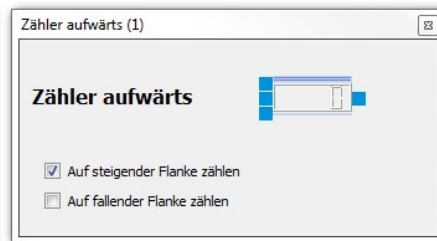
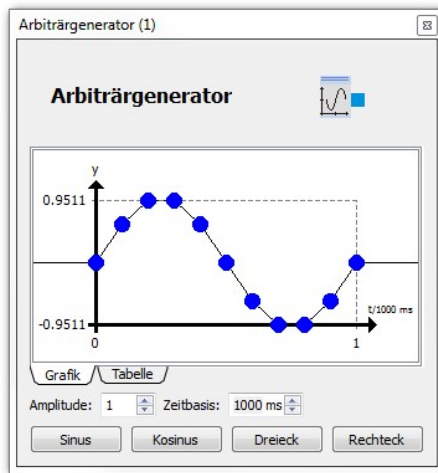


Zählen auf steigender Flanke	Erhöhung des Zählerwertes um 1 wenn der Eingang im Zeitschritt t unwahr (false) ist und im Zeitschritt t+1 wahr (true) ist.
Zählen auf fallender Flanke	Erhöhung des Zählerwertes um 1 wenn der Eingang im Zeitschritt t wahr (true) ist und im Zeitschritt t+1 unwahr (false) ist.

FESTO



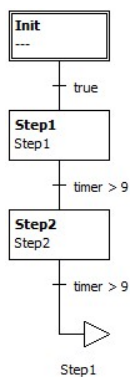
Beispiel



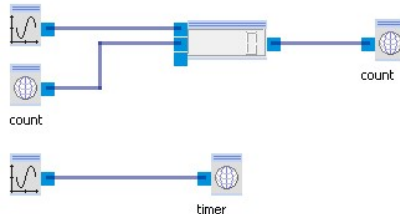
Der Arbiträrgenerator erzeugt einen Sinus mit Amplitude 2 und Frequenz 1 Hz. Der Ausgang des Generators ist vom Typ float. Zahlwerte kleiner gleich 0 werden zu unwahr (false). Zahlwerte größer 0 werden zu wahr (true) konvertiert (siehe [Typkonvertierung](#)). Der Zähler zählt auf steigender Flanke, d.h. beim Übergang von false nach true. Dieses Ereignis tritt genau einmal pro Sekunde zu Beginn des Sinus auf. Der Zählerwert entspricht also der Zeit in Sekunden seit Programmstart.

Im folgenden Beispiel wird der Eingang für den Anfangswert genutzt, um ein über Unterprogrammgrenzen fortlaufende Zählung vorzunehmen. Das Hauptprogramm führt sequentiell die Unterprogramme Step1 und Step2 aus. Nach dem Beenden von Step2 wird wieder mit Step1 begonnen.

Hauptprogramm

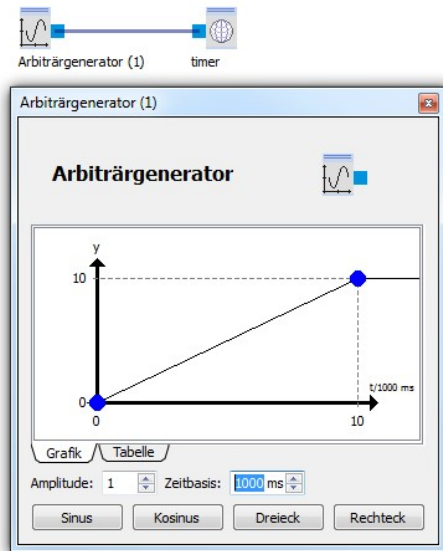


Step1



Der Zähler schreibt sein Ergebnis in die globale Variable "count". Bei Neustart des Unterprogramms Step1 wird der Wert von count als Anfangswert genutzt. Step1 ist solange aktiv, bis der Arbiträrgenerator einen Wert größer 9 erzeugt. Dies geschieht nach 10s.

Step2



Step2 ist 10s aktiv und hat ansonsten keine Funktion.

FESTO

Zähler abwärts



Der Zähler abwärts entspricht dem [Zähler aufwärts](#) mit dem Unterschied, dass bei einem Ereignis der Zählerwert nicht um 1 erhöht, sondern um 1 erniedrigt wird.

FESTO

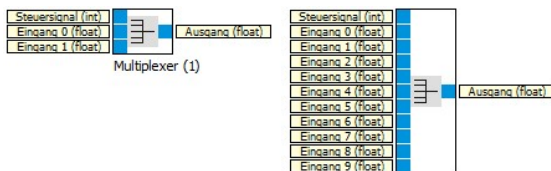
Dialog



Siehe Dialog des [Zähler aufwärts](#), mit dem Unterschied, dass der Wert nicht erhöht, sondern erniedrigt wird.

FESTO

Multiplexer

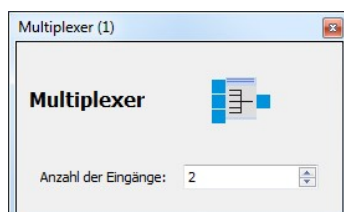


Der Multiplexer verknüpft einen Ausgang mit einem wählbaren Eingang.

Eingänge	Typ	Standard	Beschreibung
Steuersignal	int	0	Bestimmt, welcher Eingang auf den Ausgang durchgeschaltet wird. Ist das Steuersignal kleiner 0 oder größer gleich der Anzahl der Eingänge, so ist der Ausgang 0.
Eingang 0	float	0	Der Wert von Eingang 0 liegt am Ausgang an, wenn das Steuersignal 0 ist.
...			
Eingang 9	float	0	Der Wert von Eingang 9 liegt am Ausgang an, wenn das Steuersignal 9 ist.
Ausgänge			
Ausgang	float		Der Wert eines Eingangs. 0, wenn das Steuersignal kleiner 0 oder größer gleich der Anzahl der Eingänge ist.

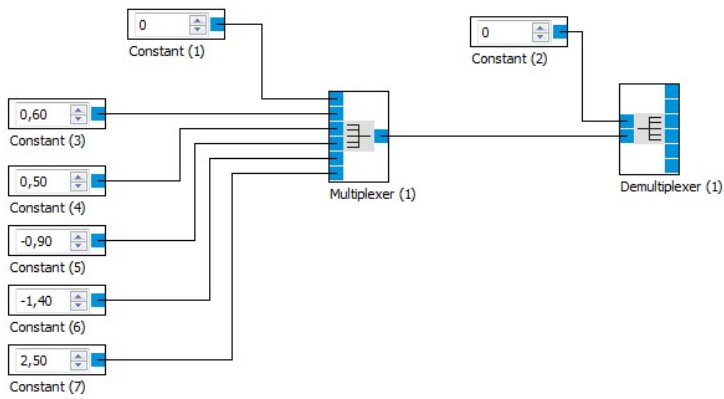
FESTO

Dialog



FESTO

Beispiel

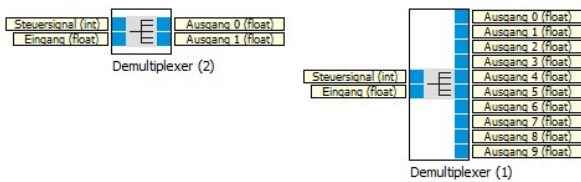


siehe auch siehe Beispiele Logik [Multiplexer](#)

FESTO



Demultiplexer



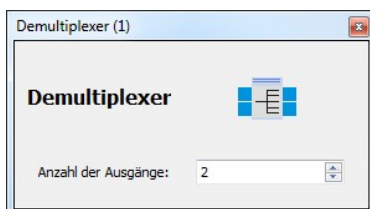
Der Demultiplexer verteilt ein Eingangssignal auf bis zu 10 umschaltbare Ausgänge.

Eingänge	Typ	Standard	Beschreibung
Steuersignal	int	0	Bestimmt, an welchen Ausgang der Eingang durchgeschaltet wird. Ist das Steuersignal kleiner 0 oder größer gleich der Anzahl der Ausgänge, werden alle Ausgänge auf 0 zurückgesetzt.
Eingang	float	0	Der Wert des Eingangs liegt am "Ausgang + Steuersignal" an.
Ausgänge			
Ausgang 0	float		Wert des Eingangs, wenn das Steuersignal 0 ist.
...			
Ausgang 9	float		Wert des Eingangs, wenn das Steuersignal 9 ist.

FESTO



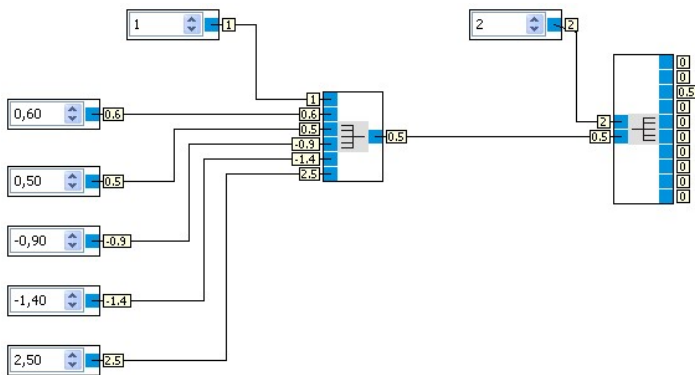
Dialog



FESTO



Beispiel



FESTO



Mathematik

Diese Klasse enthält einfache mathematische Operationen.

FESTO



Arithmetische Operationen

FESTO



Modulo



Modulo (lat. Modulus, Kasus Ablativ: "durch Maß" oder auch "mit Maß", somit Mehrzahl Moduli), mathematisches Formelzeichen mod, in vielen Programmiersprachen durch % wiedergegeben, ist eine mathematische Funktion, die den Rest aus der Division zweier ganzer Zahlen angibt. (Quelle: <http://de.wikipedia.org/wiki/Modulo>)

Eingänge	Typ	Standard	Beschreibung
Dividend	int	0	
Divisor	int	1	
Ausgänge			
Rest	int		Dividend mod Divisor

FESTO



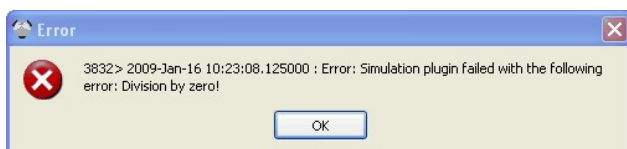
Division



Berechnet den Quotienten aus Dividend und Divisor. Siehe auch [http://de.wikipedia.org/wiki/Division_\(Mathematik\)](http://de.wikipedia.org/wiki/Division_(Mathematik)).

Eingänge	Typ	Standard	Beschreibung
Dividend	float	0	
Divisor	float	1	
Ausgänge			
Quotient	float		Dividend geteilt durch Divisor

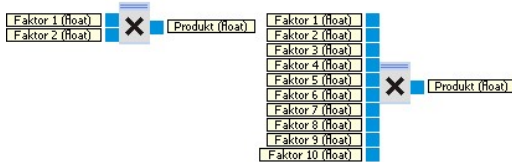
Ist der Dividend ungleich 0 und der Divisor gleich 0, wird die Simulation mit folgender Fehlermeldung abgebrochen:



FESTO



Multiplikation



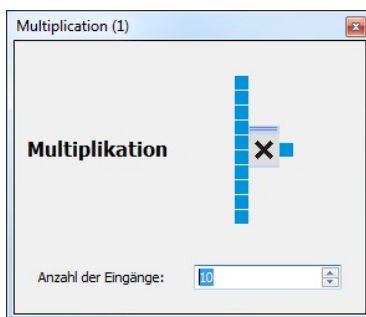
Das Multiplikationsmodul multipliziert bis zu 10 Eingangssignal. Siehe auch <http://de.wikipedia.org/wiki/Multiplikation>.

Eingänge	Typ	Standard	Beschreibung
Faktor 1	float	1	
...			
Faktor 10	float	1	
Ausgänge			
Produkt	float		"Faktor 1" * "Faktor 2" * ... * "Faktor 10"

FESTO



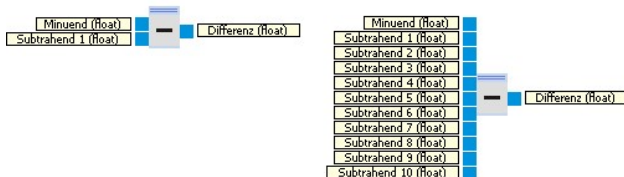
Dialog



FESTO



Subtraktion



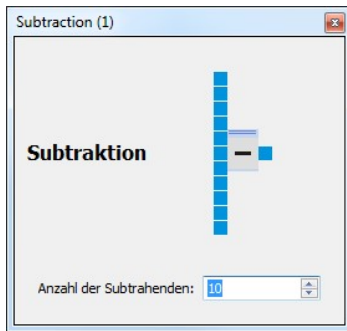
Das Subtraktionsmodul subtrahiert von dem Minuend bis zu 10 Subtrahenden. Siehe auch [http://de.wikipedia.org/wiki/Division_\(Mathematik\)](http://de.wikipedia.org/wiki/Division_(Mathematik)).

Eingänge	Typ	Standard	Beschreibung
Minuend	float	0	
Subtrahend 1	float	0	
...			
Subtrahend 10	float	0	
Ausgänge			
Differenz	float		Minuend - "Subtrahend 1" - "Subtrahend 2" - ... - "Subtrahend 10"

FESTO



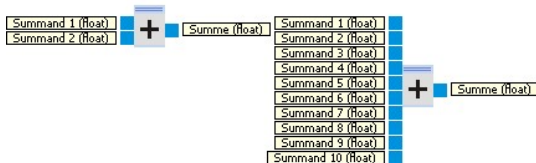
Dialog



FESTO



Addition



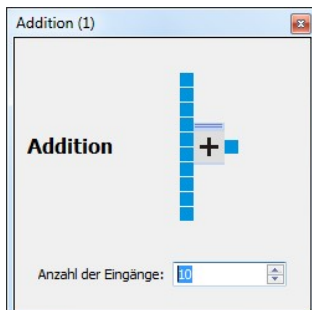
Das Additionsmodul addiert bis zu 10 Eingangswerte. Siehe auch <http://de.wikipedia.org/wiki/Addition>.

Eingänge	Typ	Standard	Beschreibung
Summand 1	float	0	
...			
Summand 10	float	0	
Ausgänge			
Summe	float		"Summand 1" + "Summand 2" + ... + "Summand 10"

FESTO



Dialog



FESTO



Vergleichende Operationen

FESTO



Ungleich



Der Ausgang wird wahr (true), wenn der absolute Betrag von Eingang 1 - Eingang 2 größer oder gleich epsilon ist, mit epsilon = 0.0000002384185792.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	0	
Eingang 2	float	0	
Ausgänge			

Ausgang	bool		fabs(Eingang 1 - Eingang 2) >= epsilon
---------	------	--	--

FESTO



Gleich

Eingang 1 (float)		Ausgang (bool)
Eingang 2 (float)		

Der Ausgang wird wahr (true), wenn der absolute Betrag von Eingang 1 - Eingang 2 kleiner epsilon ist, mit epsilon = 0.0000002384185792.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	0	
Eingang 2	float	0	
Ausgänge			
Ausgang	bool		fabs(Eingang 1 - Eingang 2) < epsilon

FESTO



Kleiner gleich

Eingang 1 (float)		Ausgang (bool)
Eingang 2 (float)		

Der Ausgang wird wahr (true), wenn Eingang 1 kleiner oder gleich Eingang 2 ist.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	0	
Eingang 2	float	0	
Ausgänge			
Ausgang	bool		"Eingang 1" kleiner oder gleich "Eingang 2"

FESTO



Kleiner

Eingang 1 (float)		Ausgang (bool)
Eingang 2 (float)		

Der Ausgang wird wahr (true), wenn Eingang 1 kleiner Eingang 2 ist.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	0	
Eingang 2	float	0	
Ausgänge			
Ausgang	bool		"Eingang 1" kleiner "Eingang 2"

FESTO



Größer gleich

Eingang 1 (float)		Ausgang (bool)
Eingang 2 (float)		

Der Ausgang wird wahr (true), wenn Eingang 1 größer oder gleich Eingang 2 ist.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	0	
Eingang 2	float	0	
Ausgänge			
Ausgang	bool		"Eingang 1" größer oder gleich "Eingang 2"

FESTO

Größer



Der Ausgang wird wahr (true), wenn Eingang 1 größer Eingang 2 ist.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	0	
Eingang 2	float	0	
Ausgänge			
Ausgang	bool		"Eingang 1" größer "Eingang 2"

FESTO



Funktionen

FESTO



Betrag



Bildet den Absolutbetrag.

Eingänge	Typ	Standard	Beschreibung
Eingang	float	0	
Ausgänge			
Ausgang	float		abs(Eingang)

FESTO



Transferfunktion



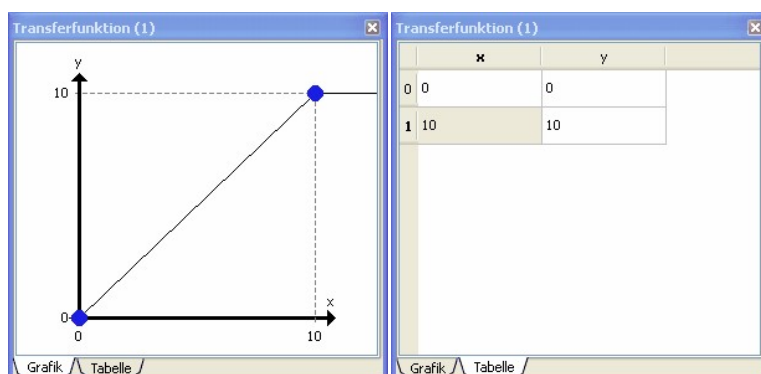
Mit diesem Funktionsblock ist eine beliebige Abbildung des Eingangs x auf den Ausgang y möglich.

Eingänge	Typ	Standard	Beschreibung
x	float	0	
Ausgänge			
x	float		siehe Dialog

FESTO



Dialog



Über den Dialog werden Stützpunkte der Abbildungsfunktion $y(x)$ definiert. Standardmäßig sind die Stützpunkte

$p_0 = (x_0, y_0) = (0, 0)$
 $p_1 = (x_1, y_1) = (10, 10)$

vorhanden. Damit ergibt sich folgende Abbildung:

$y = y_0$ für den Bereich $x \leq x_0$
 $y = x$ für den Bereich $x > x_0$ und $x \leq x_1$
 $y = y_1$ für den Bereich $x > x_1$

Randbereiche

$p_0 = (x_0, y_0)$ ist der erste Stützpunkt.
 $p_n = (x_n, y_n)$ ist der letzte Stützpunkt.

Für x kleiner als x_0 gilt: $y = y_0$

Für x größer als x_n gilt: $y = y_n$

Abbildungsfunktion

Bei einer Liste von Stützpunkten p_0, p_1, \dots, p_n ergibt sich die Abbildung $y(x)$ zu:

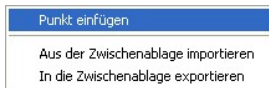
$y = y_0$ für $x \leq x_0$
 $y = (y_1 - y_0) / (x_1 - x_0) * (x - x_0) + y_0$ für $x > x_0$ und $x \leq x_1$
 $y = (y_2 - y_1) / (x_2 - x_1) * (x - x_1) + y_1$ für $x > x_1$ und $x \leq x_2$
 ...
 $y = y_n$ für $x > x_n$

Punkte verschieben

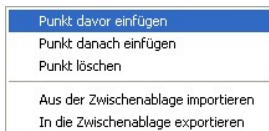
Stützpunkte können verschoben, hinzugefügt und gelöscht werden. Um einen Stützpunkt zu verschieben, können in der Grafik-Ansicht Punkte mit der Maus verschoben werden. In der Tabellen Ansicht können die x und y Werte der Stützpunkte editiert werden. Der x Wert eines Stützpunktes kann dabei niemals kleiner als der x Wert des vorherigen bzw. niemals größer als der x Wert des nachfolgenden Stützpunktes werden.

Punkte hinzufügen

In der Grafik-Ansicht kann über das mit der rechten Maustaste zugängliche Kontextmenü ein Punkt an beliebiger Stelle eingefügt werden.



In der Tabellen-Ansicht ist das Kontextmenü zugänglich, wenn man mit der rechten Maustaste auf eine Zelle klickt.



Hier kann gewählt werden, ob der Punkt vor oder nach dem aktuellen Punkt eingefügt werden soll.

Punkte löschen

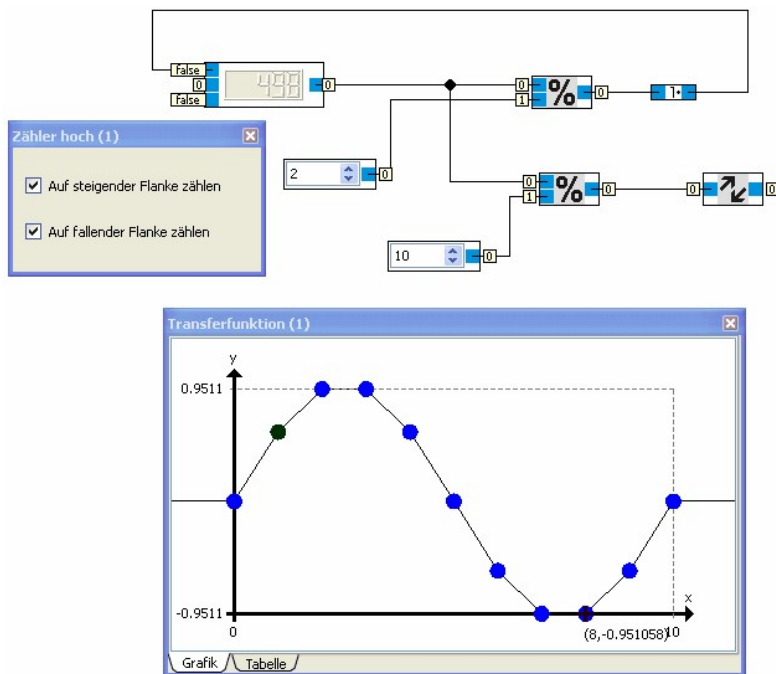
In der Grafik-Ansicht und Tabellen-Ansicht mit der rechten Maustaste auf einen Punkt klicken und über das Kontextmenü den Punkt löschen. Ist nur noch ein Stützpunkt übrig, ist die Funktion zum löschen des Punktes deaktiviert.

Import/Export der Stützpunkte

Über die Zwischenablage kann die Funktion als Tabulator separiert Liste exportiert und importiert werden. Dadurch ist der Austausch mit Programmen wie Excel oder Matlab möglich. Die Funktionen sind über das Kontextmenü sowohl in der Grafik- als auch in der Tabellen-Ansicht verfügbar.



Beispiel

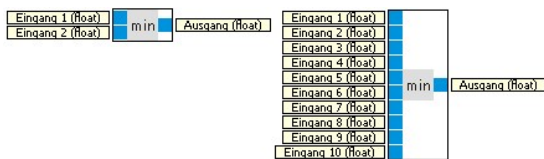


Der Zähler wird so beschaltet, dass mit jedem Simulationstakt der Zählerwert um 1 erhöht wird. Der Zählerwert wird mittels Modulo auf den Bereich [0, 10] beschränkt. Dies ist der Eingang für die Transferfunktion. Über 10 Stützpunkte wird ein Sinus abgebildet.

FESTO



Minimum



Gibt den kleinsten Eingangswert aus.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	1e+037	
...			
Eingang 10	float	1e+037	
Ausgänge			
Ausgang	float		min("Eingang 1", "Eingang 2", ... , "Eingang 10")

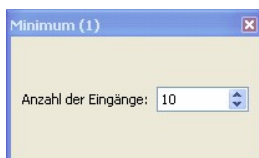
1e+037 = 10 hoch 37

größte mögliche Fließkommazahl

FESTO



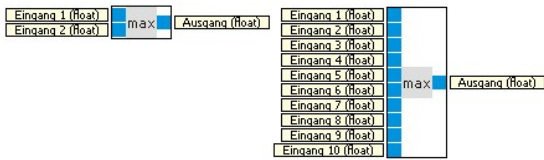
Dialog



FESTO



Maximum



Gibt den größten Eingangswert aus.

Eingänge	Typ	Standard	Beschreibung
Eingang 1	float	-1e+037	
...			
Eingang 10	float	-1e+037	
Ausgänge			
Ausgang	float		max("Eingang 1", "Eingang 2", ... , "Eingang 10")

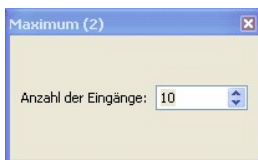
-1e+037 = - (10 hoch 37)

kleinste mögliche Fließkommazahl

FESTO



Dialog



FESTO



Skalierung



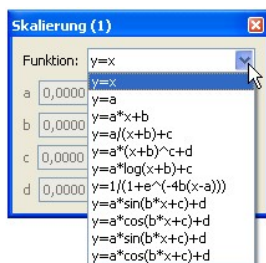
Erlaubt die einfache Skalierung von Werten.

Eingänge	Typ	Standard	Beschreibung
x	float	0	
Ausgänge			
y	float		siehe Dialog

FESTO

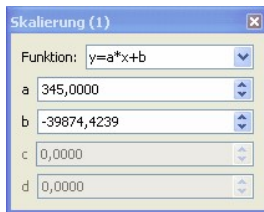


Dialog



Zunächst kann eine Funktion gewählt werden. Standardmäßig ist die Identität ausgewählt, d.h. der Eingangswert x wird unverändert auf den Ausgang y geschrieben.

Je nach Funktion, sind die Parameter a, b, c, und d editierbar. Wählt man die Funktion $y=a*x+b$ aus, so sind die Felder für a und b aktiv.



Die Abbildungsfunktion heißt in diesem Fall $y = 345 * x - 39874,4239$

FESTO

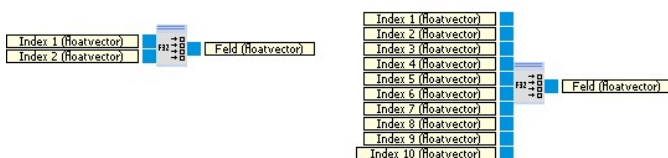


Felder

FESTO



Fließkommafeld-Zusammensetzer



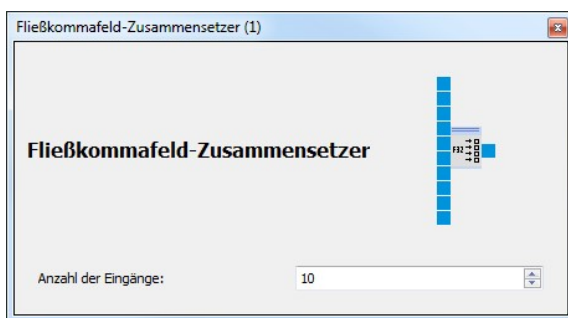
Der Fließkommafeld-Zusammensetzer erzeugt aus bis zu 10 Fließkommazahlen oder -feldern ein Fließkommafeld. Für die Konvertierung von Fließkommazahlen nach Fließkommafeldern siehe [Typkonvertierung](#).

Eingänge	Typ	Standard	Beschreibung
Index 1	float array	leeres Feld	
...			
Index 10	float array	leeres Feld	
Ausgänge			
Feld	float array	leeres Feld	(Index 1, ..., Index 10)

FESTO



Dialog



Hier kann die Anzahl der Eingänge des Funktionsblocks eingestellt werden.

FESTO



Fließkommafeld-Zerleger



Der Fließkommafeld-Zerleger extrahiert aus einem Fließkommafeld ein Teilfeld.

--	--	--	--

Eingänge	Typ	Standard	Beschreibung
Feld	float array	leeres Feld	Das zu zerlegende Feld
Start index	int	1	Der Wert an der Position Start index des zu zerlegenden Feldes wird der erste Wert des zerlegten Feldes.
Länge	int	1	Das zerlegte Feld besteht aus Länge Werten beginnend mit dem Wert an der Position Start index des zu zerlegenden Feldes.
Ausgänge			
Teilfeld	float array	leeres Feld	(Feld[Start index] , ..., Feld[Start index + Länge - 1])

FESTO



Fließkommafeld Indexzugriff



Das Indexzugriff-Modul ermöglicht den Zugriff auf einzelne Werte eines Fließkommafelds.

Eingänge	Typ	Standard	Beschreibung
Feld	float array	leeres Feld	Fließkommafeld, auf das zugegriffen werden soll.
Index	int	1	Der Index des gewünschten Werts.
Ausgänge			
Wert	float	0	Der Wert an der Position Index .

FESTO



Fließkommafeld-Zerhacker



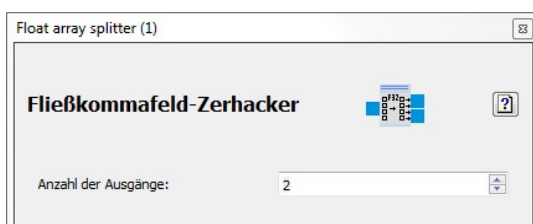
Aufteilung eines Feldes auf einzelne Ausgänge.

Eingänge	Typ	Standard	Beschreibung
Feld	float array	leeres Feld	Fließkommafeld
Ausgänge			
Index 1	float	0	Wert an Feldposition 1. Wenn die Größe des Feldes kleiner 1 ist, wird 0 ausgegeben.
...			
Index 10	float	0	Wert an Feldposition 10. Wenn die Größe des Feldes kleiner 10 ist, wird 0 ausgegeben.

FESTO



Dialog



Hier kann die Anzahl der Ausgänge des Funktionsblocks eingestellt werden.

FESTO



Beispiel

siehe [Beispiele/Felder](#)

FESTO



Vektorrechnung

Diese Klasse enthält die grundlegenden Vektorrechnungsverfahren für 2-dimensionale Vektoren.

FESTO



Vektoroperationen

FESTO



Skalarprodukt



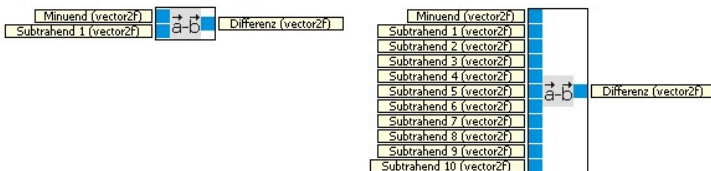
Das Skalarprodukt von zwei Vektoren. Siehe auch <http://de.wikipedia.org/wiki/Skalarprodukt>.

Eingänge	Typ	Standard	Beschreibung
Vektor 1	vector2f	(0, 0)	
Vektor 2	vector2f	(0, 0)	
Ausgänge			
Produkt	float		

FESTO



Subtraktion



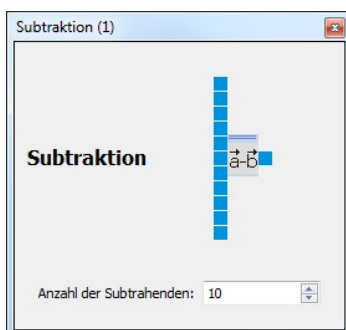
Subtraktion von bis zu 10 Vektoren. Siehe auch http://de.wikipedia.org/wiki/Vektoraddition#Addition_und_Subtraktion.

Eingänge	Typ	Standard	Beschreibung
Minuend	vector2f	(0, 0)	
Subtrahend 1	vector2f	(0, 0)	
...			
Subtrahend 10	vector2f	(0, 0)	
Ausgänge			
Differenz	vector2f		Minuend - "Subtrahend 1" - "Subtrahend 2" - ... - "Subtrahend 10"

FESTO



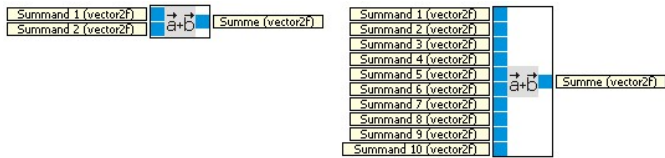
Dialog



FESTO



Addition



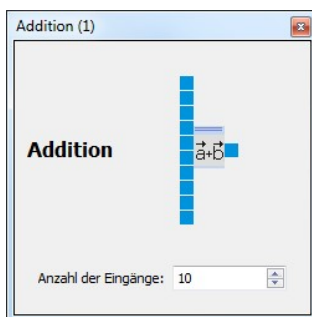
Addition von bis zu 10 Vektoren. Siehe auch http://de.wikipedia.org/wiki/Vektoraddition#Addition_und_Subtraktion.

Eingänge	Typ	Standard	Beschreibung
Summand 1	vector2f	(0, 0)	
...			
Summand 10	vector2f	(0, 0)	
Ausgänge			
Summe	vector2f		"Summand 1" + "Summand 2" + ... + "Summand 10"

FESTO



Dialog



FESTO



Norm



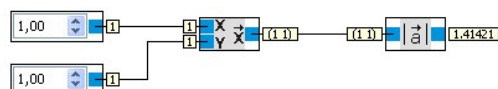
Die Norm (oder Länge) des Vektors. Siehe auch <http://de.wikipedia.org/wiki/Vektornorm>.

Eingänge	Typ	Standard	Beschreibung
Vector	vector2f	(0, 0)	
Ausgänge			
Norm	float		

FESTO



Beispiel



Die Länge des Vektors (1, 1) ist gleich Wurzel 2 (1.41421...).

FESTO



Elementoperationen

FESTO



Division



Division auf Elementebene.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Divisor	float	1	
Outputs			
Resultat	vector2f		Vector = (x0, x1) Result = (x0 / Divisor, x1 / Divisor)



Subtraktion



Subtraktion auf Elementebene.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Minuend	float	0	
Outputs			
Resultat	vector2f		Vector = (x0, x1) Result = (x0 - Minuend, x1 - Minuend)



Addition



Addition auf Elementebene.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Summand	float	0	
Outputs			
Resultat	vector2f		Vector = (x0, x1) Result = (Summand + x0, Summand + x1)



Multiplikation



Multiplikation auf Elementebene.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Factor	float	1	
Outputs			
Resultat	vector2f		Vektor = (x0, x1) Resultat = (Factor * x0, Factor * x1)



Transformationen



FESTO

Vektor nach Polar



Zerlegung eines Vektors in seine polaren Komponenten.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Ausgänge			
Länge	float		Länge (Norm) von Vektor .
Phi	float		Winkel zwischen Vektor und der x-Achse in Grad.

FESTO



Vektor nach Kartesisch



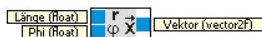
Zerlegung eines Vektors in seine kartesischen Komponenten.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Ausgänge			
x	float		x-Komponente von Vektor .
y	float		y-Komponente von Vektor .

FESTO



Polar nach Vektor



Erzeugung eines Vektors aus seiner Länge und Orientierung.

Eingänge	Typ	Standard	Beschreibung
Länge	float	0	Länge (Norm) des Vektors.
Phi	float	0	Winkel zwischen dem Vektor und der x-Achse.
Ausgänge			
Vektor	vector2f		Vektor mit Länge Länge und Orientierung Phi .

FESTO



Kartesisch nach Vektor



Erzeugung eines Vektors aus seinen kartesischen Komponenten.

Eingänge	Typ	Standard	Beschreibung
x	float	0	x-Komponente.
y	float	0	y-Komponente.
Ausgänge			
Vektor	vector2f		

Ausgänge			
Vektor	vector2f		Vektor (x, y).

FESTO



Rotieren



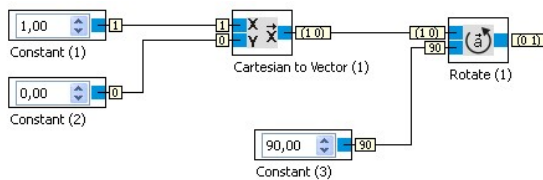
Rotiert den Vektor um den vorgegebenen Wert in Grad.

Eingänge	Typ	Standard	Beschreibung
Vektor	vector2f	(0, 0)	
Phi	float	0	Rotationswinkel
Ausgänge			
Resultat	vector2f		Vektor um Phi rotiert.

FESTO



Beispiel



FESTO



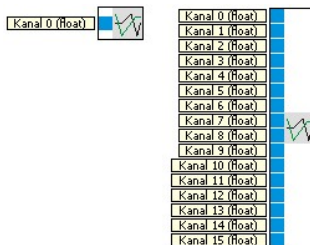
Anzeigen

Diese Klasse enthält Funktionsblöcke zur Visualisierung von Daten.

FESTO



Oszilloskop



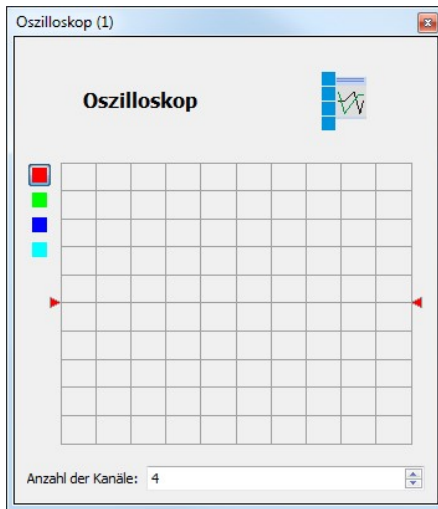
Mit dem Oszilloskop können bis zu 16 Kanäle dargestellt werden.

Eingänge	Typ	Standard	Beschreibung
Kanal 0	float	0	
Kanal 1	float	0	
...			
Kanal 16	float	0	

FESTO



Dialog



Der Dialog dient zur Visualisierung der an den Kanälen anliegenden Signale. Für jeden Kanal können Einstellungen wie z.B. Verstärkung vorgenommen werden. Einzelne Kanäle können auch deaktiviert werden.

Rechtsklick-Optionen

Nach Klick mit der rechten Maustaste in den Dialog öffnet sich ein Menü mit folgenden Optionen:

1. Aktivieren/Deaktivieren, Einstellen der Werte von Zeit, Verstärkung und Trigger für alle Kanäle.
2. Aktivieren/Deaktivieren, Einstellen der Werte von Zeit, Verstärkung und Trigger für bestimmte Kanäle.
3. Kopieren der grafischen Ausgabe als Bild zur weiteren Verwendung z.B. in Grafikprogrammen wie Paint.
4. Kopieren der Daten zur weiteren Verwendung z.B. in Microsoft Excel oder OpenOffice Calc.

FESTO



Laserscanner Datenanzeige



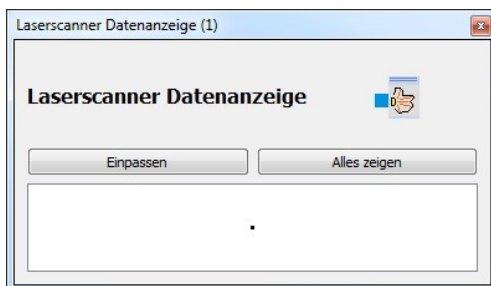
Die Laserscanner Datenanzeige ermöglicht die Visualisierung der Daten von einem Laserscanner.

Eingänge	Typ	Beschreibung
Daten	laser range data	

FESTO



Dialog



FESTO



Bildverarbeitung

Diese Klasse enthält Funktionsblöcke zur Bildverarbeitung.

FESTO



Linienerkennung



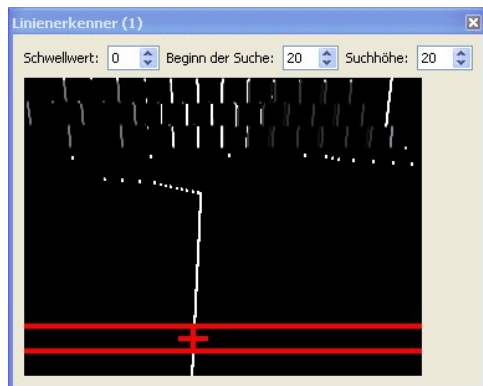
Findet Linien in einem Bild.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Schwellwert	int	0	Legt die Sensitivität des Algorithmus in Bezug auf Rauschen im Bild fest. Um Rauschen zu unterdrücken muss ein höherer Grenzwert gewählt werden. Wertebereich: [0,255]
Beginn der Suche	int	20	Die Suche nach einer Linie beginnt am unteren Bildrand den angegebenen Wert.
Suchhöhe	int	20	Das Bild wird von unten nach oben auf Linien untersucht. Die Suchhöhe legt die Anzahl der Bildlinien fest, die für eine Linienerkennung berücksichtigt werden.
Ausgänge			
x	int		x-Position der innerhalb des Suchfensters gefundenen Linie.
Linie gefunden	bool		Wahr (true) wenn eine Linie gefunden wurde, ansonsten unwahr (false)

FESTO



Dialog



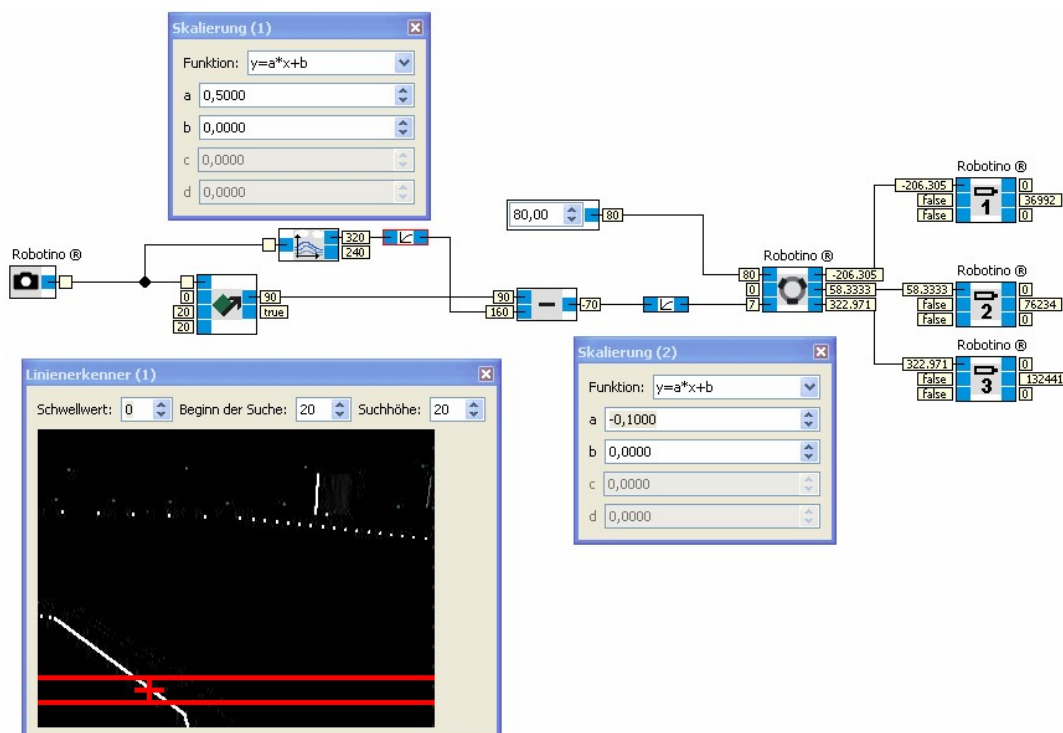
Der Bereich, in welchem nach einer Linie gesucht wird, ist durch die beiden horizontal verlaufenden roten Linien gekennzeichnet. Die untere Linie markiert den Beginn der Suche. Die obere Linie hat als y Koordinate den Wert Bildhöhe - "Beginn der Suche" - Suchhöhe. Der Bereich zwischen den Linien ist das Suchfenster.

Das rote + markiert die von links auftretende dunkel-hell Kante der Linie.

FESTO



Beispiel



Robotinos Kamera liefert ein Bild, welches als Eingang für den Linienerkenner dient. Mithilfe der Bildinformation wird die x-Position der Linie im Bild von dem Bereich

[0;Bildbreite] in den Bereich [-Bildbreite/2; Bildbreite/2] abgebildet. Nach anschließendem Wechsel des Vorzeichens und Skalierung erhält man einen Wert, der direkt als Eingangswert für die Winkelgeschwindigkeit des Antriebs von Robotino dienen kann. Auf diese Weise dreht Robotino nach links, wenn sich die Linie in der linken Bildhälfte befindet und nach rechts, wenn sich die Linie in der rechten Bildhälfte befindet. Mit einer Konstanten Vorwärtsgeschwindigkeit fährt Robotino der Linie nach.

FESTO



ROI



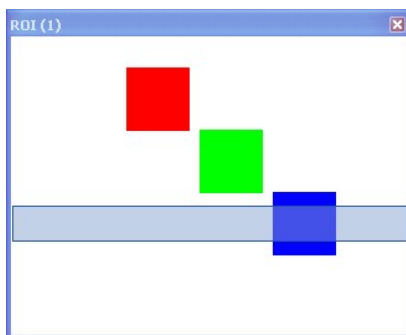
Wählt eine interessante Region in einem Bild (Region Of Interest, ROI).

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Ausgang	image		Das um die ROI erweiterte Eingangsbild. Nachfolgende Bildverarbeitungsoperationen arbeiten auf der gewählten Region.

FESTO



Dialog

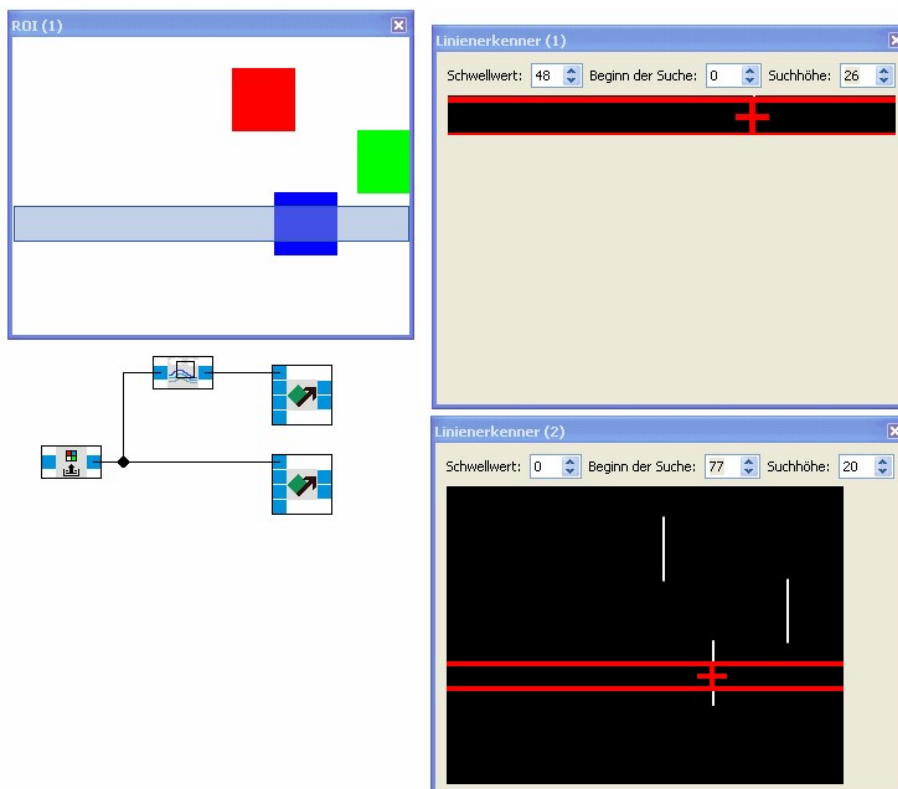


In dem angezeigten Eingangsbild kann eine interessante Region mit der Maus selektiert werden.

FESTO



Beispiel



Der Bildleser erzeugt eine Testsequenz von Bildern. Der untere Linienkennner arbeitet auf dem vollen Bild, welches von dem Bildleser erzeugt wird. Der obere Linienkennner arbeitet nur auf dem durch die vorgeschaltete ROI als interessant markierten Bereich.



Bildinformation

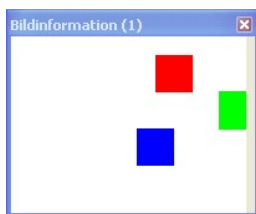


Liefert die Breite und Höhe eines Bildes.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Breite	int		Die Bildbreite in Pixel.
Höhe	int		Die Bildhöhe in Pixel.



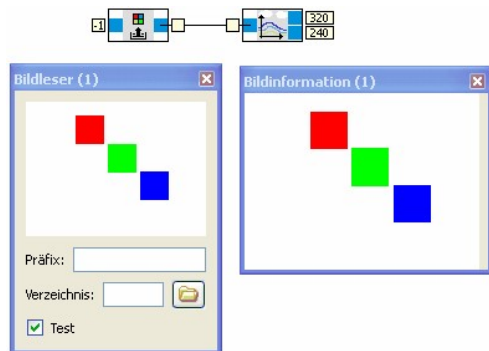
Dialog



Der Dialog zeigt das Eingangsbild.



Beispiel



Die Bilder der vom Bildleser erzeugten Testsequenz haben eine Auflösung von 320 x 240 Pixel.



Farbraumkonvertierung

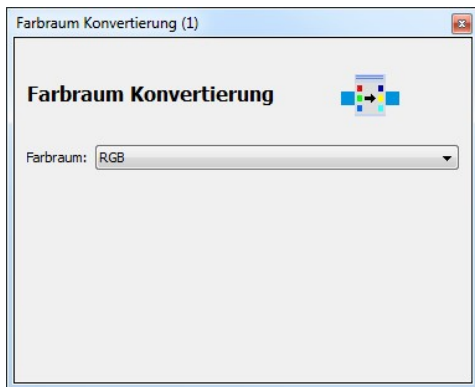


Robotino überträgt JPEG komprimierte Bilder, die in den Farbraum RGB dekodiert werden. In diesem Farbraum werden die Farben in der klassischen Form dargestellt. Dieser Farbraum hat jedoch den Nachteil, dass er sehr sensitiv auf Lichteinflüsse reagiert. Der Farbraum YCbCr ist in der Regel deutlich stabiler gegenüber Helligkeitsstörungen.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Ausgang	image		Konvertiertes Bild



Dialog



Im Dialog zur Farbraumkonvertierung kann der Ziel-Farbraum eingestellt werden.



Farbbereichssuche

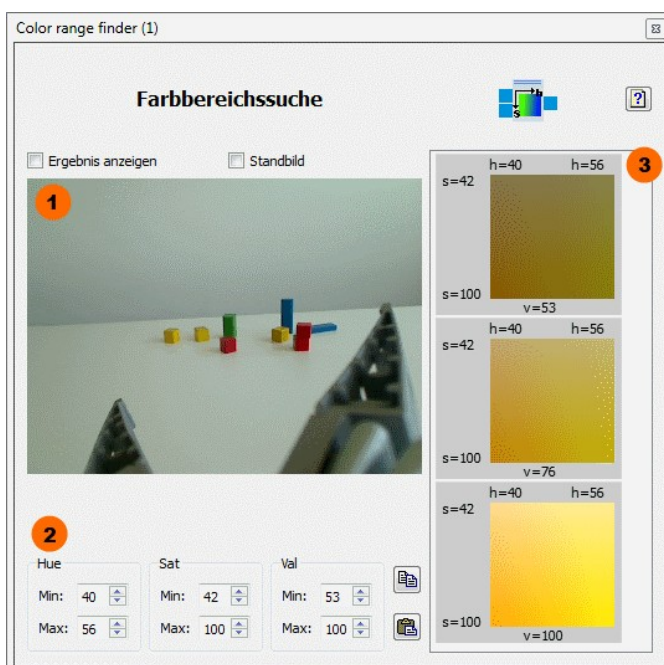


Die Pixel, deren Farbwert innerhalb eines ausgewählten Bereichs liegen, werden weiß eingefärbt. Pixel, deren Farbwert außerhalb des gewählten Bereichs liegen, werden schwarz eingefärbt.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild. Es können Bilder in den Formaten RGB, BGR und HSV verarbeitet werden.
Farbbereich	intvector	(0 0 0 0 0)	Setzt den Farbbereich, nachdem das Bild durchsucht wird. Einstellungen, die im Dialog gemacht wurden, werden überschrieben.
Ausgänge			
Ausgang	image		Graustufenbild





Dialog



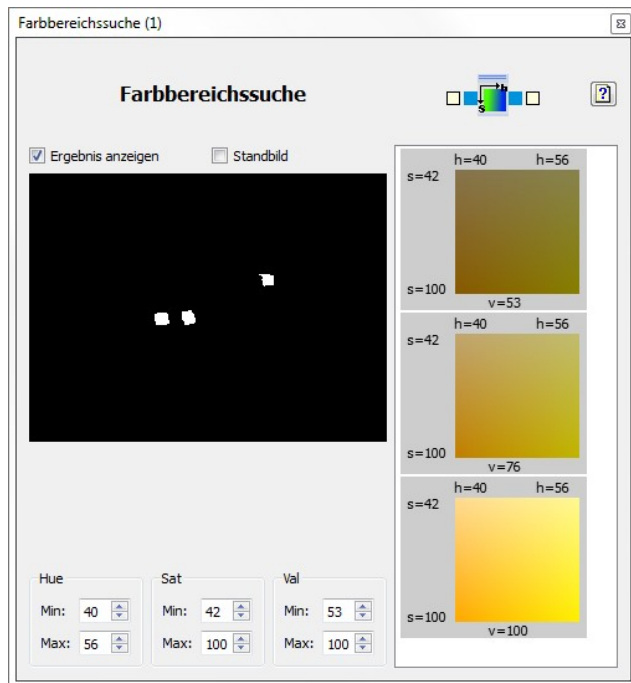
Wenn "Ergebnis anzeigen" nicht angekreuzt ist, wird das Eingangsbild angezeigt (1). In dem Eingangsbild kann man einen Bereich mit der Maus durch klicken und ziehen auswählen. Nach der Auswahl eines Bereichs wird der maximale und minimale Wert aller Pixel im ausgewählten Bereich pro Bildkanal ermittelt. Dabei wird auf einem in den [HSV-Farbraum](#) konvertierten Bild gearbeitet. Die minimalen und maximalen Werte werden unter (2) angezeigt. (3) ist eine visuelle Darstellung des ausgewählten Farbbereichs. Für den minimalen, den mittleren und den maximalen Hellwert (Value) wird jeweils ein Bild gezeigt, das die Farben in dem gewählten Bereich für Farbtone (Hue) und Sättigung (Sat) zeigt.

Durch Veränderung der Werte unter (2) ändert sich der Farbbereich und damit auch die Darstellung in (3).

 Kopiert die aktuellen Werte in die Zwischenablage. Von dort können die Werte in eine globale Variable vom Typ "floatvector" eingefügt werden.

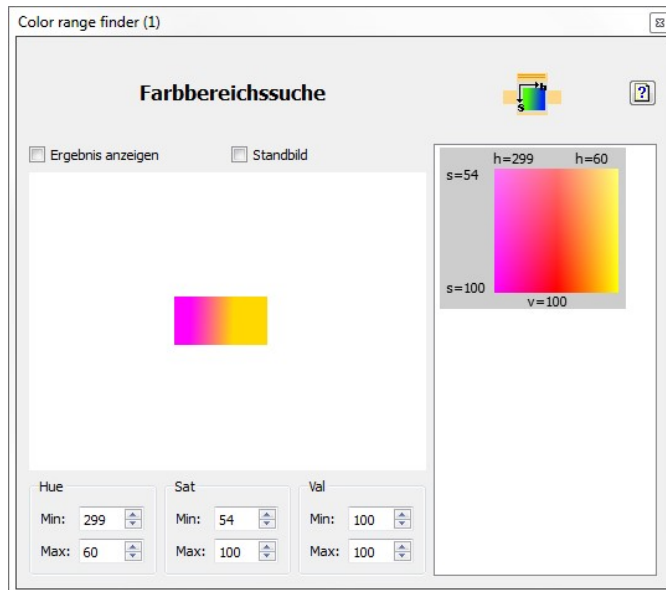
 Überschreibt die aktuellen Werte mit Werten aus der Zwischenablage.

Wählt man "Ergebnis anzeigen", wird in (1) das erzeugte Schwarz-Weiß-Bild gezeigt.



Je geschlossener und größer die weißen Bereiche sind, desto besser passt der gewählte Farbbereich zu den Farben im Live-Bild. Um die Erkennung der Farben robust gegen Helligkeitsunterschiede zu machen, erweitert man den Bereich des Hellwerts (Val). Bei glänzenden Oberflächen kann es zu einer Variation der Farbsättigung (Saturation) kommen, so dass man diesen Bereich erweitert, um die Farben bei unterschiedlichen Bedingungen zu erkennen. Eine komplette Änderung der Farbe kommt normalerweise nicht vor, so dass der Farbtonbereich (Hue) nur selten angepasst werden muss.

Im HSV-Farbraum wird der Farbton (Hue) als Farbwinkel zwischen 0° und 360° dargestellt. Ein der Farbton 361 ist gleich dem Farbton 1. Problematisch ist es, wenn ein Bildbereich ausgewählt wird, dessen Farbtöne sich über die 0° Grenze erstrecken.



Die oben dargestellte Abbildung zeigt einen Bildbereich, dessen Farbton von violett bis gelb variiert. Violett hat einen Farbton von 300 und gelb einen Farbton von 60. Würde man einfach den minimalen und maximalen Farbton betrachten, so ergäbe sich der Bereich der Farbtöne, nach denen im Bild gesucht wird zu [60;300]. In diesem Bereich liegen die Farben grün und blau.

Die Lösung für dieses Problem besteht darin, den Abstand der Farbtongrenzen zu betrachten. In diesem Beispiel ist der Abstand $300-60=240$. Sobald der Abstand größer 180 ist, wird angenommen, dass sich der Farbbereich für die Suche über die 0° Grenze erstreckt. Dies wird dadurch angezeigt, dass der Min und Max Wert vertauscht werden und damit $\text{Max} < \text{Min}$ ist. Wenn das der Fall ist, erstreckt sich der Farbtonbereich von Min-359 und von 0-Max.

FESTO



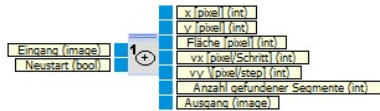
Beispiel

siehe [Farbverfolgung](#)

FESTO



Segmentverfolger



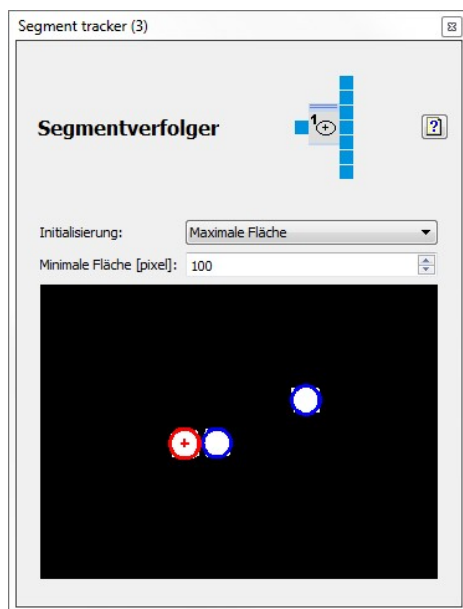
Dieser Funktionsblock findet in einem Schwarz-Weiß-Bild, wie es von der Farbbereichssuche erzeugt wird, zusammenhängende Segmente und verfolgt ein bestimmtes Segment. Um alle Segmente in einem Schwarz-Weiß-Bild zu verfolgen, schaltet man entsprechend viele Segmentverfolger-Funktionsblöcke hintereinander.

Eingänge	Typ	Einheit	Standard	Beschreibung
Eingang	image			Eingangsbild
Neustart	bool		false	Die Segmentverfolgung wird neu initialisiert, wenn der Eingang wahr ist. Dadurch ist es möglich, während des Programmablaufs zum größten bzw. zum der Bildmitte nächsten Segment zu wechseln. Wenn man überhaupt keine Verfolgung des Segments wünscht, sondern immer nur das größte, der Bildmitte am nächsten ... Segment finden möchte, setzt man diesen Eingang konstant auf wahr.
Ausgänge				
x	int	pixel		x-Koordinate in Pixel des Schwerpunkts des verfolgten Segments
y	int	pixel		y-Koordinate in Pixel des Schwerpunkts des verfolgten Segments
Fläche	int	pixel		Anzahl der Pixel aus denen das verfolgte Segment besteht
vx	int	pixel/Schritt		Anzahl der Pixel, um die sich das verfolgte Segment seit dem vorherigen Zeitschritt bewegt hat. Positiv: Bewegung nach rechts. Negativ: Bewegung nach links.
vy	int	pixel/Schritt		Anzahl der Pixel, um die sich das verfolgte Segment seit dem vorherigen Zeitschritt bewegt hat. Positiv: Bewegung nach unten. Negativ: Bewegung nach oben.
Anzahl gefundener Segmente	int			Die Anzahl der gefundenen Segmente
Ausgang	image			Das Schwarz-Weiß-Bild mit Informationen über die gefundenen Segmente. Ein nachgeschalteter Segmentverfolger kann ein weiteres Segmente verfolgen.

FESTO



Dialog



Initialisierung: Hiermit wird die Methode gewählt, nach welcher das zu verfolgende Segment ausgewählt wird. Der Initialisierungsschritt findet statt bei jedem Programmstart und wenn der Neustart Eingang wahr ist.

- Maximale Fläche: Es wird das Segment verfolgt, welches bei Programmstart die maximale Fläche unter allen gefundenen Segmenten hat.
- Bildmitte: Es wird das Segment verfolgt, welches bei Programmstart (oder bei Neustart=true) der Bildmitte am nächsten ist.
- Unten: Es wird das Segment verfolgt, welches bei Programmstart (oder bei Neustart=true) dem unteren Bildrand am nächsten ist.
- Oben: Es wird das Segment verfolgt, welches bei Programmstart (oder bei Neustart=true) dem oberen Bildrand am nächsten ist.
- Links: Es wird das Segment verfolgt, welches bei Programmstart (oder bei Neustart=true) dem linken Bildrand am nächsten ist.
- Rechts: Es wird das Segment verfolgt, welches bei Programmstart (oder bei Neustart=true) dem rechten Bildrand am nächsten ist.

Minimale Fläche: Segmente werden nur dann berücksichtigt, wenn ihre Fläche größer als die angegebene minimale Fläche ist.

Bildansicht: Segmente mit einer Fläche größer der minimalen Fläche werden mit einem Kreis markiert, dessen Fläche der Fläche des Segments entspricht. Der Kreis um das verfolgte Segment wird rot, der um die übrigen Segmente blau eingefärbt. Zusätzlich wird in das Zentrum des verfolgten Segments ein rotes Kreuz eingezeichnet. Die Position des Kreuzes entspricht den ausgegebenen Werte für x und y.

FESTO

Beispiel



siehe [Farbverfolgung](#)

FESTO

Filter

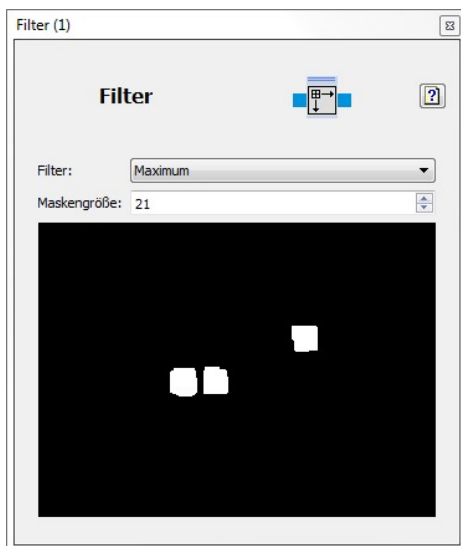


Dieser Funktionsblock wendet unterschiedliche Filter auf Farb-, Graustufen oder Schwarz-Weiß-Bilder an.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Ausgang	image		Gefiltertes Bild

FESTO

Dialog



Filter: Auswahl des Filters

- Maximum: Setzt den Farbwert des aktuellen Pixel auf das Maximum der Werte der Pixel innerhalb der Maske. Das Ausgangsbild verkleinert sich dabei sowohl in der Höhe als auch in der Breite um Maskengröße-1. Bei einem Farbbild wird der Filter pro Kanal angewendet. [siehe Beispiele](#)
- Minimum: Setzt den Farbwert des aktuellen Pixel auf das Minimum der Werte der Pixel innerhalb der Maske. Das Ausgangsbild verkleinert sich dabei sowohl in der Höhe als auch in der Breite um Maskengröße-1. Bei einem Farbbild wird der Filter pro Kanal angewendet. [siehe Beispiele](#)

Maskengröße: Höhe und Breite der Filtermaske.

FESTO

Beispiel



siehe [Farbverfolgung](#)

Beispiel für den Maximum-Filter:

Maskengröße=3, Eingangsbild (7x5), Ausgangsbild (5x3)

1	2	3	4	5	6	7
1	2	3	4	5	6	7
2	3	4	5	6	7	8
2	3	4	5	6	7	8
3	4	5	6	7	8	9

4	5	6	7	8
4	5	6	7	8
5	6	7	8	9

Beispiel für den Minimum-Filter:

Maskengröße=3, Eingangsbild (7x5), Ausgangsbild (5x3)

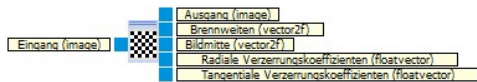
1	2	3	4	5	6	7
1	2	3	4	5	6	7
2	3	4	5	6	7	8
2	3	4	5	6	7	8
3	4	5	6	7	8	9

1	2	3	4	5
1	2	3	4	5
2	3	4	5	6

FESTO



Kalibrierung



Funktionsblock zur Kalibrierung der Kamera.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Ausgang	image		Bild mit Kalibrierungsdaten
Brennweite	vector2f		Informationen zur Brennweite der Kamera
Bildmitte	vector2f		Informationen zur Bildmitte des Kamerasensors
Radiale Verzerrungskoeffizienten	floatvector		
Tangentiale Verzerrungskoeffizienten	floatvector		

FESTO



Dialog



FESTO

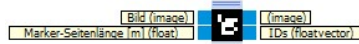


Beispiel

FESTO



Markerdetektion



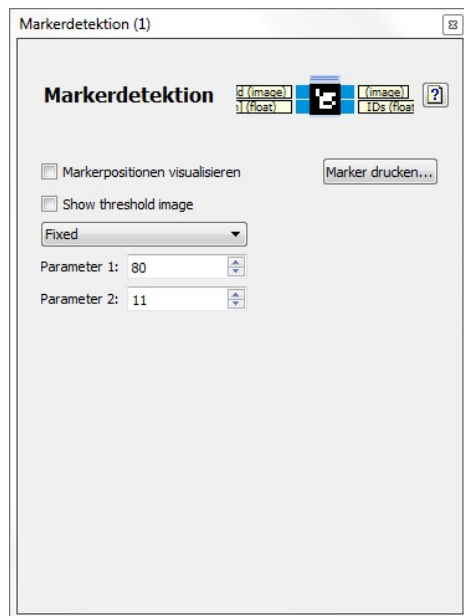
Funktionsblock zum Auffinden von Markern in einem Bild.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Ausgang	image		Bild mit Informationen zu gefundenen Markern
IDs	floatvector		IDs aller gefundenen Marker

FESTO



Dialog



FESTO



Markerposition



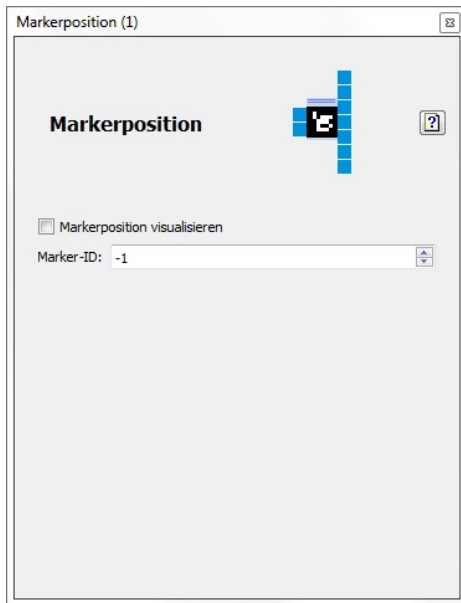
Funktionsblock zur Ausgabe der Position eines Markers relativ zur Kamera.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
ID	int	0	Die ID des Markers, dessen Position ermittelt werden soll.
Ausgänge			
Marker gefunden	bool	false	Wahr wenn Marker mit der vorgegebenen ID gefunden wurde. Ansonsten falsch.
x	float	0	x-Position des Markers relativ zur Kamera.
y	float	0	y-Position des Markers relativ zur Kamera.
z	float	0	z-Position des Markers relativ zur Kamera.
a	float	0	Neigung des Markers relativ zur Kamera.
b	float	0	Neigung des Markers relativ zur Kamera.
c	float	0	Neigung des Markers relativ zur Kamera.

FESTO



Dialog



FESTO



Beispiel

FESTO



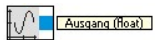
Generatoren

Diese Klasse enthält Funktionsblöcke zur Erzeugung von Signalen.

FESTO



Arbiträrgenerator



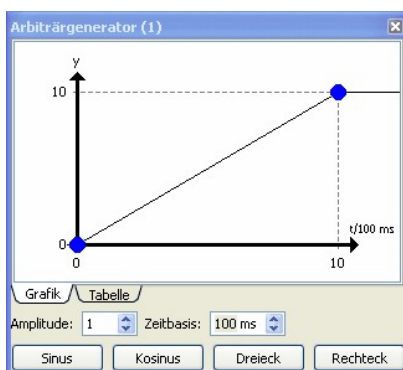
Der Arbiträrgenerator erzeugt beliebig geformte Ausgangssignale. Siehe auch <http://de.wikipedia.org/wiki/Arbitr%C3%A4rgenerator>.

Eingänge	Typ	Standard	Beschreibung
Ausgänge			
Ausgang	float		Das erzeugte Signal.

FESTO



Dialog



Der obere Teil des Dialogs besteht aus der von der [Transferfunktion](#) bekannten grafischen und tabellarischen Ansicht zur Definition einer beliebigen Funktion durch Stützpunkte.

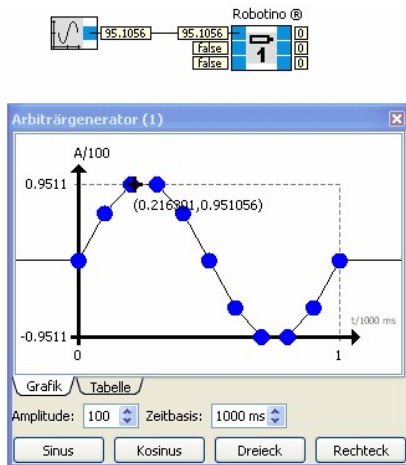
Amplitude Der Ausgangswert des Generators wird mit der Amplitude multipliziert.

Zeitbasis	Die Einheit der x-Achse. Mit einer Zeitbasis von 100ms wird der Wert 10 auf der x-Achse nach 1s erreicht.
Sinus	Erzeugt Stützpunkte, die eine Sinusschwingung approximieren.
Kosinus	Erzeugt Stützpunkte, die eine Kosinusschwingung approximieren.
Dreieck	Erzeugt Stützpunkte, die eine Sägezahnkurve approximieren.
Rechteck	Erzeugt Stützpunkte, die einen Rechteckpuls approximieren.

FESTO



Beispiel

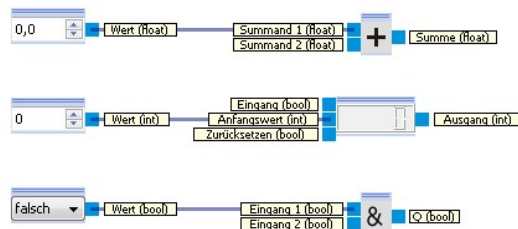


Robotinos Motor 1 dreht mit einem Sinus angesteuert vorwärts und rückwärts.

FESTO



Konstante



Erzeugt einen konstanten Wert. Der Typ der Konstanten und damit auch die grafische Darstellung ändert sich mit dem Datentyp des angeschlossenen Eingangs.

Die Eingabe des Wertes der Konstanten erfolgt direkt im Programm. Eingaben über die Tastatur führen direkt zu einer Änderung des Wertes.

Eingänge	Typ	Standard	Beschreibung
Ausgänge			
Wert	float, int, bool	0 / false	Der Wert der Konstanten.

FESTO



Zeitbaustein



Misst die Zeit in Millisekunden seit Programmstart oder seit einem Übergang wahr (true) nach unwahr (false) an dem Zurücksetzen Eingang.

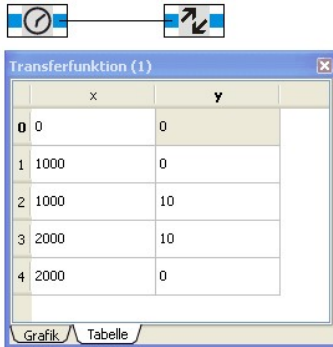
Eingänge	Typ	Standard	Beschreibung
Zurücksetzen	bool	false	Wenn wahr, liefert der Ausgang des Zeitbausteins 0. Wenn unwahr, ist der Zeitbaustein aktiv.
Ausgänge			

Zeit	float	Zeit in Millisekunden seit Programmstart oder seit Zurücksetzen von wahr nach unwahr gewechselt hat.
------	-------	--

FESTO



Beispiel



Der Zeitbaustein erzeugt zusammen mit der [Transferfunktion](#) 1s nach Programmstart einen Puls der Höhe 10 für 1s.

FESTO



Zufallsgenerator



Der Zufallsgenerator erzeugt beliebige Zahlen innerhalb eines definierten Wertebereichs.

Eingänge	Typ	Standard	Beschreibung
Maximum	float	1	Obere Grenze des Wertebereichs.
Minimum	float	0	Untere Grenze des Wertebereichs.
Ausgänge			
Wert	float	0	Zufallszahl zwischen Minimum und Maximum .

FESTO



Filter

Dieser Klasse enthält Funktionsblöcke zum Filtern und Glätten von Signalen.

FESTO



Mittelwertfilter



Bildet den Mittelwert des Eingangssignals über bis zu 1000 Zeitschritte.

Eingänge	Typ	Standard	Beschreibung
Eingang	float	0	Eingangssignal
Ausgänge			
Ausgang	float		Geglättetes Eingangssignal

FESTO



Dialog



Die Tiefe bestimmt die Anzahl der vorangegangenen Zeitschritt, über welche der Mittelwert gebildet wird.

FESTO



Navigation

Diese Klasse enthält Funktionsblöcke zur Navigation.



Positionsfahrer



Mit dem Positionsfahrer können Positionen angefahren werden.

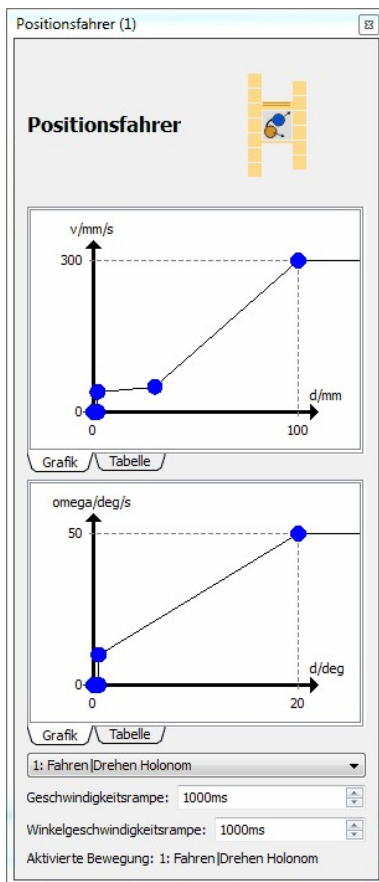
Der Positionsfahrer berechnet aus einer Soll- und einer Istposition Geschwindigkeit und Winkelgeschwindigkeit, so dass Robotino von der Ist- zur Sollposition fährt.

Eingänge	Typ	Einheit	Beschreibung
x Soll	float	mm	x Koordinate der Soll Position im Weltkoordinatensystem.
y Soll	float	mm	y Koordinate der Soll Position im Weltkoordinatensystem.
phi Soll	float	Grad	Winkel phi der Soll Position im Weltkoordinatensystem.
x Ist	float	mm	x Koordinate der Ist-Position im Weltkoordinatensystem.
y Ist	float	mm	y Koordinate der Ist-Position im Weltkoordinatensystem.
phi Ist	float	Grad	Winkel phi der Ist-Position im Weltkoordinatensystem.
Neustart	bool		Startet die Bewegung erneut
Ausgänge			
vx	float	mm/s	x Geschwindigkeit.
vy	float	mm/s	y Geschwindigkeit.
omega	float	Grad/s	Winkelgeschwindigkeit.
Position erreicht	bool		Sobald die Ausgänge vx und vy 0 liefern, wird die Zielposition als erreicht angesehen und der Ausgang liefert wahr.
Orientierung erreicht	bool		Sobald der Ausgang omega 0 liefern, wird die Zielorientierung als erreicht angesehen und der Ausgang liefert wahr.
Pose erreicht	bool		Der Ausgang liefert wahr, wenn sowohl die Zielposition als auch die Zielorientierung erreicht sind. Ansonsten unwahr.

Siehe [Bewegungen](#)



Dialog

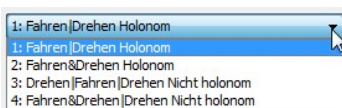


Der Dialog zum Positionsfahrer gliedert sich in drei Bereiche.

Im oberen Bereich wird die Beziehung zwischen Distanz zur Zielposition d (gemessen in mm) und der zu fahrenden Geschwindigkeit v (gemessen in mm/s) angegeben.

Im mittleren Bereich wird die Beziehung zwischen Winkeldistanz zum Zielwinkel d (gemessen in 1°) und der Rotationsgeschwindigkeit ω (gemessen in $1^\circ/\text{s}$) angegeben. Die Winkeldistanz variiert dabei zwischen 0° und 180° . Drehungen im oder gegen den Uhrzeigersinn werden symmetrisch behandelt. Die Drehung wird im oder gegen den Uhrzeigersinn ausgeführt, so dass die Drehdistanz minimal ist.

Mittels der ComboBox



wird die Art der Bewegung festgelegt (siehe [Bewegungen](#)). Die hier gewählte Bewegung wird zur aktivierten Bewegung bei

1. Programmstart
2. Wenn der Eingang Neustart auf wahr gesetzt wird

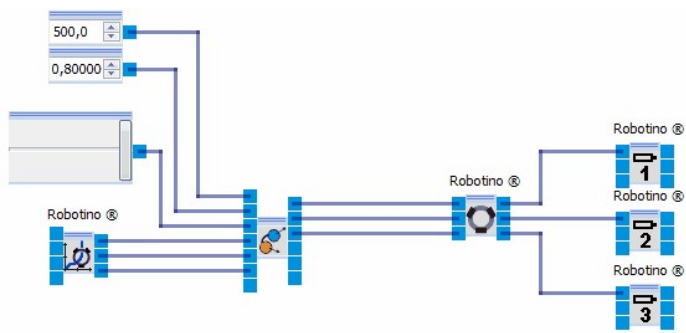
Die Geschwindigkeitsrampe ist die Zeit in Millisekunden, nach welcher 100% der gewünschten Geschwindigkeit erreicht werden. Dadurch wird ein abrupter Anstieg der Geschwindigkeit zu Beginn der Bewegung vermieden.

Die Winkelgeschwindigkeitsrampe ist die Zeit in Millisekunden, nach welcher 100% der Rotationsgeschwindigkeit erreicht werden. Dies dient ebenfalls zur Dämpfung der Bewegung, wenn eine neue Rotation beginnt.

FESTO



Beispiel



FESTO



Bewegungen

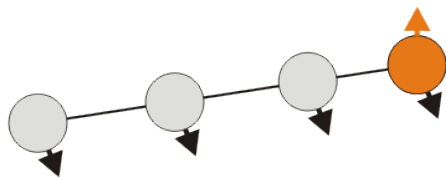
Es sind vier Arten von Bewegungen möglich. Davon sind jeweils zwei für holonome und nicht holonome Fahrzeuge geeignet. Da Robotino über einen holonomen Antrieb verfügt - alle drei Freiheitsgrade in der Ebene können unabhängig verändert werden - kann Robotino alle vier Bewegungen ausführen. Bei den nicht holonomen Bewegungen ist der Ausgang v_y konstant gleich 0.

Bewegungen beginnen bei Programmstart oder wenn der Eingang Neustart wahr wird. Effektiv beginnt die Bewegung im 2. Fall erst dann, wenn der Eingang Neustart wieder zurück auf unwahr gesetzt wird.

Bewegung 1 - fahren, drehen - (holonom)

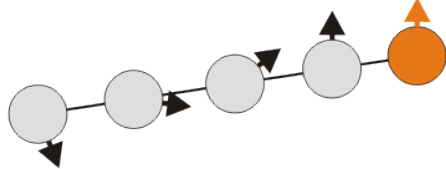
Schritt 1: Fahren zur Zielposition unter Beibehaltung der Orientierung an der Startposition

Schritt 2: Nach Erreichen der Zielposition drehen bis zur Zielorientierung



Bewegung 2 - fahren & drehen - (holonom)

Schritt 1: Fahren und gleichzeitig drehen zur Zielorientierung

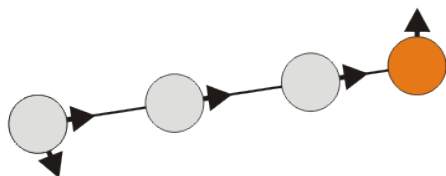


Bewegung 3 - drehen, fahren, drehen - (nicht holonom)

Schritt 1: Drehen in Fahrtrichtung

Schritt 2: Fahren zur Zielposition

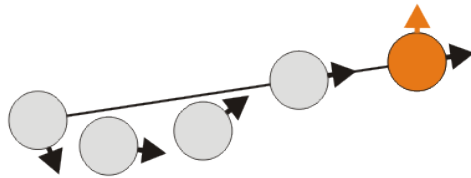
Schritt 3: Nach Erreichen der Zielposition drehen bis zur Zielorientierung



Bewegung 4 - fahren & drehen, drehen - (nicht holonom)

Schritt 1: Fahren und drehen in Fahrtrichtung

Schritt 2: Nach Erreichen der Zielposition drehen bis zur Zielorientierung



FESTO



Konstante Pose

0 0 0 [Pose (pose)]

Über das Eingabefeld wird die Pose definiert. Koordinaten werden durch Leerzeichen getrennt.

Eingabe	Resultierende Pose
x y phi	(x, y, phi)
x y	(x, y, ungültig)
x	ungültige Pose
	ungültige Pose

Die Orientierung phi wird im Eingabefeld in Grad angegeben.

Beispiel:

10.5 20 120

ergibt x=10.5 y=20 und Orientierung=120°

Eingänge	Typ	Standard	Beschreibung
Ausgänge			
Pose	pose	ungültige Pose	Der Wert der konstanten Pose. Der Wert für die Orientierung wird am Ausgang in Radiant dargestellt.

FESTO



Posenzusammensetzer

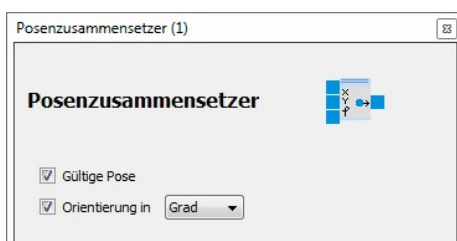
x (float) y (float) phi (float) [Pose (pose)]

Eingänge	Typ	Einheit	Standard	Beschreibung
x	float		0	Die x Komponente der Pose.
y	float		0	Die y Komponente der Pose.
phi	float	Grad	0	Die Orientierung der Pose in Grad. Die Einheit kann im Dialog auf Radiant umgestellt werden.
Ausgänge				
Pose	pose		(0, 0, 0)	Die aus den Einzelwerten zusammengesetzte Pose (x, y, phi). Der Wert für die Orientierung wird am Ausgang in Radiant dargestellt.

FESTO



Dialog

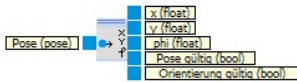


Pose gültig	Legt fest, ob die Pose gültig ist. Ungültige Posen werden in Wegen ignoriert.
Orientierung	Legt fest, ob die Orientierung gültig ist, sowie deren Einheit (Grad oder Radiant).

FESTO



Posenzerleger

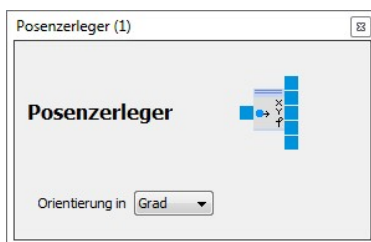


Eingänge	Typ	Einheit	Standard	Beschreibung
Pose	pose		(0, 0, 0)	Die zu zerlegende Pose.
Ausgänge				
x	float		0	Die x Komponente der Pose.
y	float		0	Die y Komponente der Pose.
phi	float	Grad	0	Die Orientierung der Pose in Grad. Die Einheit kann im Dialog auf Radiant umgestellt werden.
Pose gültig	bool		false	Zeigt an, ob die Pose gültig ist.
Orientierung gültig	bool		false	Zeigt an, ob die in der Pose gespeicherte Orientierung gültig ist.

FESTO



Dialog



FESTO



Wegzusammensetzer

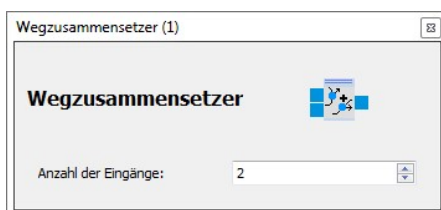


Eingänge	Typ	Standard	Beschreibung
Weg 1	path	leerer Weg	Der erste Teilweg. Hier kann auch eine einzelne Pose angelegt werden, weil pose in path konvertierbar ist. Siehe Typkonvertierung .
...			
Weg 20	path	leerer Weg	Der letzte Teilweg. Hier kann auch eine einzelne Pose angelegt werden, weil pose in path konvertierbar ist. Siehe Typkonvertierung .
Ausgänge			
Weg	path	leerer Weg	Der aus den Teilwegen zusammengesetzte Weg Weg 1 + ... + Weg 20

FESTO



Dialog



FESTO



Wegzerleger



Schneidet aus einem Weg einen Teilweg aus. Ein Weg besteht aus einer Liste von Posen.

Index	Pose
1	p1
2	p2
...	
N	pN

Die Eingänge **Start** und **Länge** bestimmen Startpose und Länge des zerlegten Weges. **Start** muss in dem Wertebereich [1;N] liegen. Ist **Start** <1, wird intern der Wert 1 verwendet. Ist **Start** > der Länge des Weges, so wird ein leerer Weg ausgegeben. **Länge** muss in dem Wertebereich [0;N-**Start**+1]. Bei einer **Länge** <=0 wird ein leerer Weg ausgegeben. Ist die Länge >N-**Start**+1, wird der mit Index **Start** beginnende Weg ausgegeben.

Beispiele:

Weg = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Start = 3

Länge = 5

Teilweg = p3, p4, p5, p6, p7

Weg = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Start = 0

Länge = 1

Teilweg = p1

Weg = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Start = 11

Länge = 1

Teilweg = leerer Weg

Weg = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Start = 1

Länge = 0

Teilweg = leerer Weg

Weg = p1, p2, p3, p4, p5, p6, p7, p8, p9, p10

Start = 2

Länge = 20

Teilweg = p2, p3, p4, p5, p6, p7, p8, p9, p10

Eingänge	Typ	Standard	Beschreibung
Weg	path	leerer Weg	Der zu zerlegende Weg
Start	int	1	Die Pose am Index Start des zu zerlegenden Weges wird die erste Pose des zerlegten Weges.
Länge	int	1	Der zerlegte Weg besteht aus Länge Posen beginnend mit Pose am Index Start des zu zerlegenden Weges.
Ausgänge			
Teilweg	path	leerer Weg	Der ausgegebene Weg beginnt mit der Pose am Index Start und hat besteht aus Länge Posen.

FESTO



Wegbefahrer



Mit dem Wegbefahrer können Wege abgefahren werden.

Aus dem Weg und der aktuellen Pose wird die Geschwindigkeit und Winkelgeschwindigkeit berechnet, so dass Robotino gradlinig die einzelnen Posen des Weges abfährt.

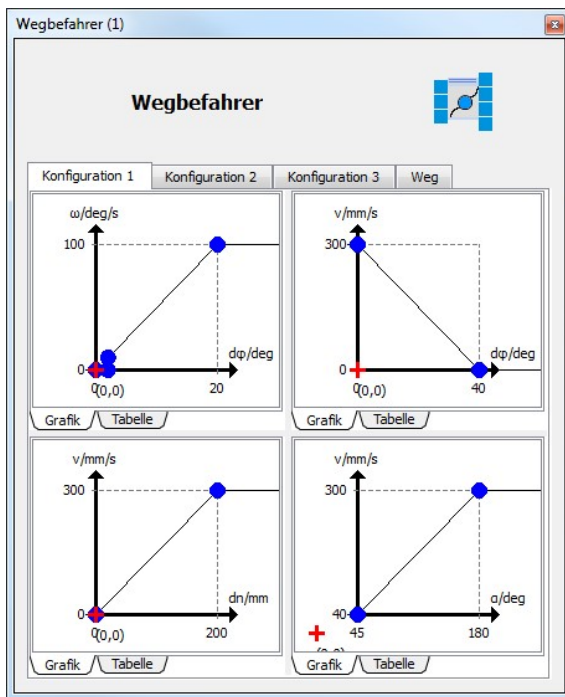
Eingänge	Typ	Einheit	Standard	Beschreibung
Weg	path		leerer Weg	Der zu fahrende Weg.

aktuelle Pose	pose	(0, 0, 0)	Die aktuelle mittels Odometrie oder SLAM ermittelte Pose.
Neustart	bool	false	Startet die Bewegung erneut
Ausgänge			
Geschwindigkeit	float	mm/s	Vorwärtsgeschwindigkeit.
Winkelgeschwindigkeit	float	Grad/s	Winkelgeschwindigkeit.
Position erreicht	bool		Im Falle eines leeren Weges liefert der Ausgang wahr zurück. Ansonsten liefert der Ausgang wahr, wenn sich der virtuelle Punkt auf dem letzten Wegabschnitt befindet und $v(d) = 0$ ist.
Nächster Wegpunkt	pose		Nächster Wegpunkt, der angefahren wird.

FESTO



Konfigurationsdialog 1



Oben Links

Zusammenhang zwischen Drehgeschwindigkeit und Winkelfehler $d\phi$.

Oben Rechts

Zusammenhang zwischen Vorwärtsgeschwindigkeit und Winkelfehler $d\phi$.

Unten Links

Zusammenhang zwischen Vorwärtsgeschwindigkeit und Abstand zum nächsten Wegpunkt.

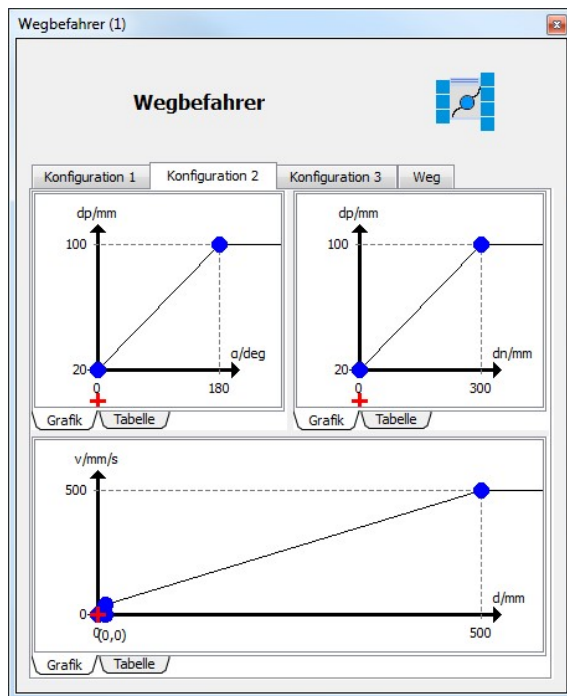
Unten Rechts

Zusammenhang zwischen Vorwärtsgeschwindigkeit und Winkel zur nächsten Wegstrecke.

FESTO



Konfigurationsdialog 2

**Oben Links**

Zusammenhang zwischen Abstand des Roboters zu virtuellem Wegpunkt und Winkel zur nächsten Wegstrecke.

Oben Rechts

Zusammenhang zwischen Abstand des Roboters zu virtuellem Wegpunkt und Abstand zum nächsten Wegpunkt.

Unten

Zusammenhang zwischen Vorwärtsgeschwindigkeit und Abstand zum Wegende.

**Konfigurationsdialog 3**

The screenshot shows the 'Wegbefahrer (1)' window with the 'Konfiguration 3' tab selected. It contains two sections: 'Geschwindigkeitskopplung' and 'Winkelgeschwindigkeitskopplung'.

Geschwindigkeitskopplung

- Anstiegszeit: 2000ms (Min: 0,00, Max: 1,00)
- $\delta\varphi$: 5° (Min: 0,50)
- Progress bar: 0%

Winkelgeschwindigkeitskopplung

- Anstiegszeit: 500ms (Min: 0,00, Max: 1,00)
- $\delta\varphi$: 5° (Min: 0,50)
- Progress bar: 0%

Oben

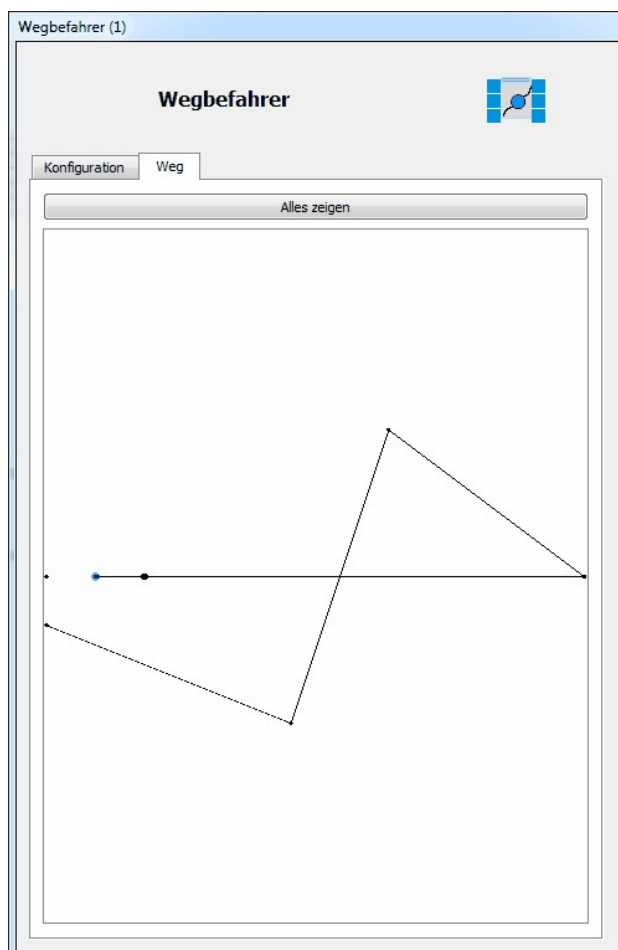
Anpassung des Kopplungsfaktors zwischen der aufgrund der Konfiguration in den Dialogen 1 und 2 errechneten und der tatsächlich ausgegebenen Geschwindigkeit.

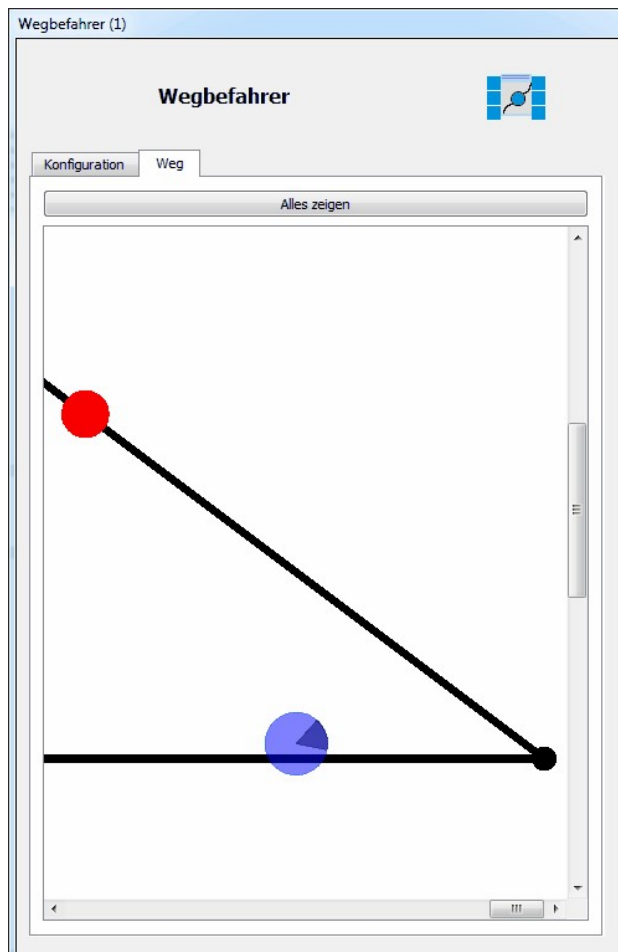
Unten

Anpassung des Kopplungsfaktors zwischen der aufgrund der Konfiguration in den Dialogen 1 und 2 errechneten und der tatsächlich ausgegebenen Drehgeschwindigkeit.

FESTO

Wegansicht

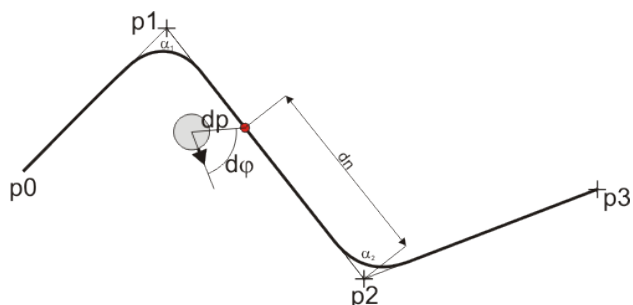




FESTO



Strategie



Der Wegbefahrer Funktionsblock erzeugt einen Weg, der zunächst die Wegpunkte gradlinig verbindet.

Der Roboter wird mittels eines virtuellen Wegpunktes (hier als roter Punkt eingezeichnet) geführt. Von der aktuellen Roboterposition ausgehend wird der virtuelle Wegpunkt so auf dem Weg platziert, dass der Abstand zwischen Roboter und virtuellem Wegpunkt d_p (distance virtual point) entspricht. Der virtuelle Wegpunkt kann sich auf dem Weg nur in Richtung des Wegendes bewegen, d.h. entfernt sich der Roboter vom virtuellen Wegpunkt, so bleibt dieser unverändert. Durch die Regelung auf den virtuellen Punkt kommt es zu einer Glättung des Weges. Diese Glättung ist um so größer, je größer der Abstand d_p ist.

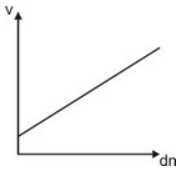
Winkelgeschwindigkeitsparametrierung

Die Winkelgeschwindigkeit $\omega(d\phi)$ wird über den Funktionsblockdialog abhängig vom Winkelfehler $d\phi$ angegeben. $d\phi$ ist der Winkel zwischen aktueller Roboterorientierung und der Verbindung Robotermittelpunkt zu virtuellem Wegpunkt.

Geschwindigkeitsparametrierung

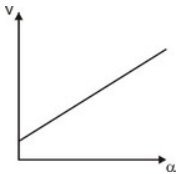
Die Geschwindigkeit wird ebenfalls in Abhängigkeit von $d\phi$ angegeben, bezeichnet als $v(d\phi)$. Damit besteht die Möglichkeit die Bewegung zu verlangsamen, wenn der Roboter nicht mehr korrekt ausgerichtet ist.

Um die Geschwindigkeit verlangsamen zu können, wenn der Weg abknickt, gibt man die Geschwindigkeit auch als Funktion $v(d_n)$ des Abstandes zwischen dem virtuellen Punkt und dem nächsten Wegpunkt an. Ein typischer Verlauf von $v(d_n)$ ist



D.h. die Geschwindigkeit soll abnehmen, je näher der Roboter dem Wegpunkt (und damit der nächsten Biegung) kommt.

Allerdings will man den Roboter in Abhängigkeit des Winkels α_n abbremsen. α_n ist der Winkel zwischen der aktuellen und der nächsten Wegstrecke. Für den Fall das $\alpha_n 180^\circ$ ist (d.h. der Weg geht gerade durch den Wegpunkt), soll die Geschwindigkeit nicht verlangsamt werden. Für den Fall das α_n gegen 0° geht (ein sehr starker Knick), muss der Roboter auch sehr stark verlangsamt werden. Aus diesem Grund benötigt man die Funktion $v(\alpha_n)$. Ein typischer Verlauf von $v(\alpha_n)$ sieht so aus



D.h. je kleiner α_n , desto kleiner ist auch die zu fahrende Geschwindigkeit.

Die drei Geschwindigkeitsprofile $v(d\phi)$, $v(dn)$ und $v(\alpha)$ werden zu der Gesamtweggeschwindigkeit $V(d\phi, dn, \alpha)$ verrechnet:

$$V_p(d\phi, dn, \alpha) = \min(v(d\phi), \max(v(dn), v(\alpha)))$$

Anfahrt des letzten Wegpunktes

Zum abbremsen bei Erreichen des Wegendes, wird die Geschwindigkeit in Abhängigkeit zum noch zu fahrenden Weg angegeben, bezeichnet als $v(d)$. Das Ziel wird als erreicht angenommen, wenn die Geschwindigkeit als Funktion des noch zu fahrenden Weges gleich Null ist.

Die ungeglättete Geschwindigkeit berechnet sich zu:

$$V(d, d\phi, dn, \gamma) = \min(v(d), V_p(d\phi, dn, \gamma))$$

Glättung der Geschwindigkeit und Winkelgeschwindigkeit

Es stehen zwei weitere Parameter zur Glättung der Bewegung zur Verfügung.

Die **Geschwindigkeitskopplung** bezeichnet die Zeit in Millisekunden, die benötigt wird, damit die Kopplung vCC zwischen errechneter Geschwindigkeit $V_p(d\phi, dn, \alpha)$ und der ausgegebenen Geschwindigkeit $velocity$ den Wert 1 erreicht.

Die **Winkelgeschwindigkeitskopplung** bezeichnet die Zeit in Millisekunden, die benötigt wird, damit die Kopplung ωCC zwischen errechneter Winkelgeschwindigkeit $\omega(d\phi)$ und der ausgegebenen Winkelgeschwindigkeit ω den Wert 1 erreicht.

$$dv = vCC * (V_{p_t} - V_{p_{t-1}})$$

$$velocity = V_{p_{t-1}} + dv$$

$$d\omega = \omega CC * (\omega(d\phi)_t - \omega(d\phi)_{t-1})$$

$$velocity = \omega(d\phi)_{t-1} + d\omega$$

Das tiefgestellte t bezeichnet den Wert zum Zeitpunkt t . $t-1$ bezeichnet den Wert einen Zeitschritt vor t .

Bei Neustart wird vCC mit 0 initialisiert und steigt innerhalb der mit der **Geschwindigkeitskopplung** gegebenen Zeit auf der Wert 1 an.

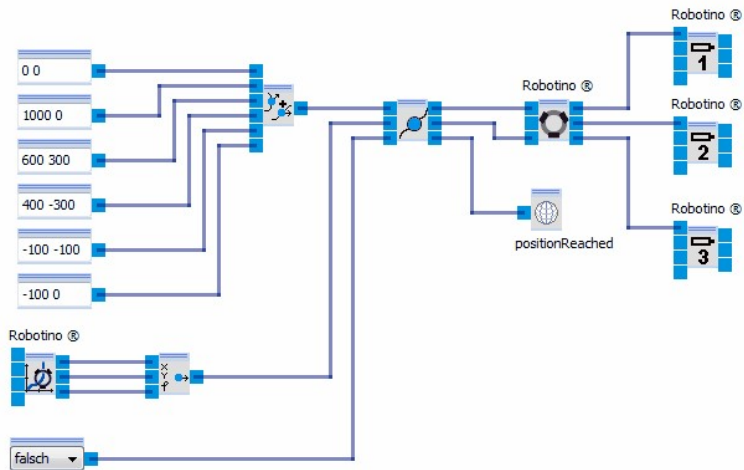
Bei Neustart wird ωCC mit 0 initialisiert und steigt innerhalb der mit der **Winkelgeschwindigkeitskopplung** gegebenen Zeit auf der Wert 1 an.

Springt der virtuelle Punkt auf einen neuen Wegabschnitt, so werden vCC und ωCC ebenfalls wieder auf 0 gesetzt.

FESTO



Beispiel



FESTO



Hindernisvermeidung



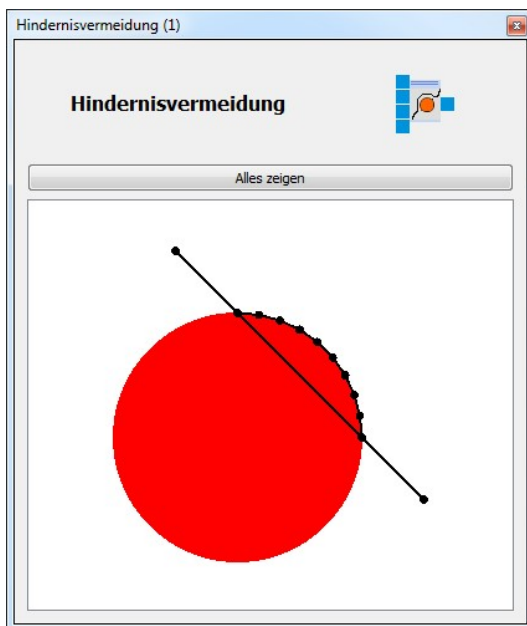
Das Modul Hindernisvermeidung berechnet für einen gegebenen Weg eine Umleitung um ein kreisförmiges Hindernis.

Eingänge	Typ	Einheit	Standard	Beschreibung
Weg	path		leerer Weg	Der zu fahrende Weg.
Hindernispose	pose		(0, 0, 0)	Die Position des kreisförmigen Hindernisses.
Hindernisradius	float	mm	100	Der Radius des kreisförmigen Hindernisses. Addieren Sie den Radius von Robotino und einen Sicherheitsabstand, damit Robotino das Hindernis umfahren kann.
Winkelentfernung	float	Grad	10	Der maximale Winkelabstand zwischen zwei Stützpunkten des Umwegs um das Hindernis.
Ausgänge				
Umweg	path		leerer Weg	Umweg um das Hindernis herum.

FESTO



Dialog



Der Dialog zeigt den ursprünglichen Weg sowie das Hindernis und den Umweg.

FESTO

Eingabegeräte



Diese Klasse liefert Funktionsblöcke, um Interaktionen mit dem Anwender zu realisieren.

FESTO

Steuerungsfeld

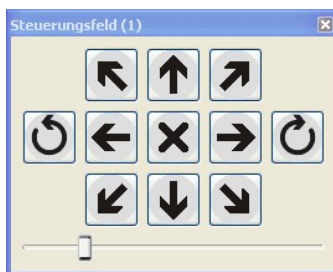


Dieser Funktionsblock stellt ein Steuerungsfeld bereit, das mit der Maus bedient werden kann.

Ausgänge	Typ	Beschreibung
vx	float	Geschwindigkeit in x-Richtung
vy	float	Geschwindigkeit in y-Richtung
omega	float	Drehgeschwindigkeit

FESTO

Dialog

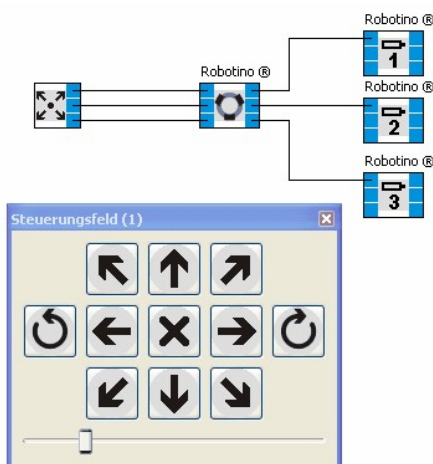


Das Steuerungsfeld kann wie folgt bedient werden:

- Durch Klicken auf eins der Felder wird das Robotersystems in die entsprechende Pfeilrichtung bewegt.
- Durch Mausklick auf eines der beiden kreisförmigen Pfeile wird eine Rotation in die entsprechende Drehrichtung durchgeführt.
- Durch Mausklick auf die mittlere Taste wird die Bewegung gestoppt.
- Über den Schieber wird die Geschwindigkeit der Bewegung eingestellt.

FESTO

Beispiel



FESTO

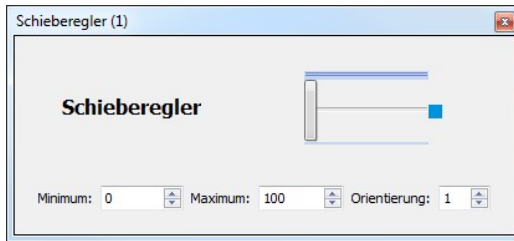
Schieberegler



Mit dem Schieberegler können beliebige ganzzahlige Werte innerhalb eines definierten Wertebereichs erzeugt werden.

FESTO

Dialog



Im Dialog kann der Wertebereich sowie die Orientierung (1 = horizontal, 0 = vertikal) des Schiebereglers eingestellt werden.

FESTO



Datenaustausch

Diese Klasse enthält Funktionsblöcke zum Austausch von Daten innerhalb von Robotino® View oder mit externen Applikationen.

FESTO



Bildleser



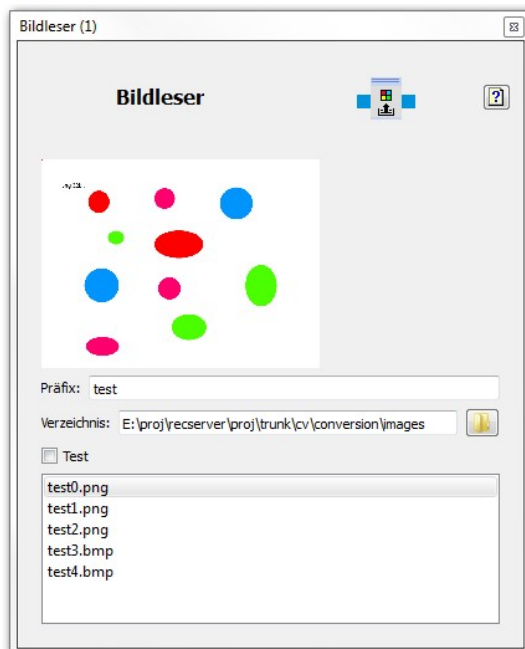
Der Bildleser liest einzelne Bilder vom Dateisystem. Pfad und Präfix können im [Dialog](#) eingestellt werden. Unterstützt werden die Formate ".bmp", ".png", ".jpg", ".jpeg".

Eingänge	Typ	Standard	Beschreibung
Nummer	int16	-1	Nummer des gewünschten Bildes der Folge. Falls Nummer = -1, wird bei 0 beginnend in jedem Schritt automatisch um 1 hochgezählt.
Ausgänge			
Ausgang	image		Das aktuelle Bild.

FESTO



Dialog



Von oben nach unten:

- Anzeige des aktuellen Bildes
- Präfix: Einschränkung bei der Suche nach Bildern mit den Endungen ".bmp", ".png", ".jpg" und ".jpeg".
- Verzeichnis: Das Verzeichnis, in dem nach Bildern gesucht wird.
- Test: Aktivierung der Testfunktion
- Liste der gefundenen Bilddateien. Diese Liste wird erst bei Start des Programms erstellt bzw. aktualisiert.

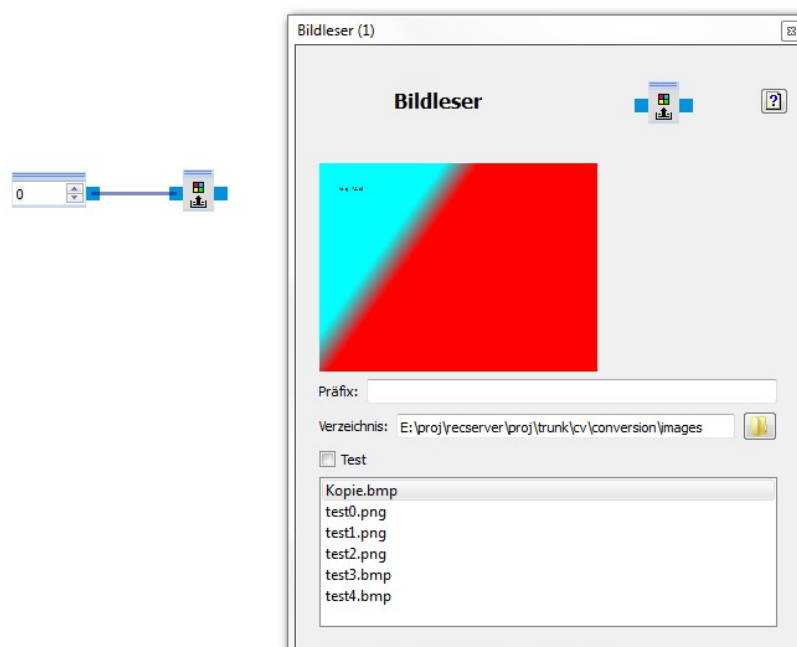
siehe [Beispiel](#)

FESTO

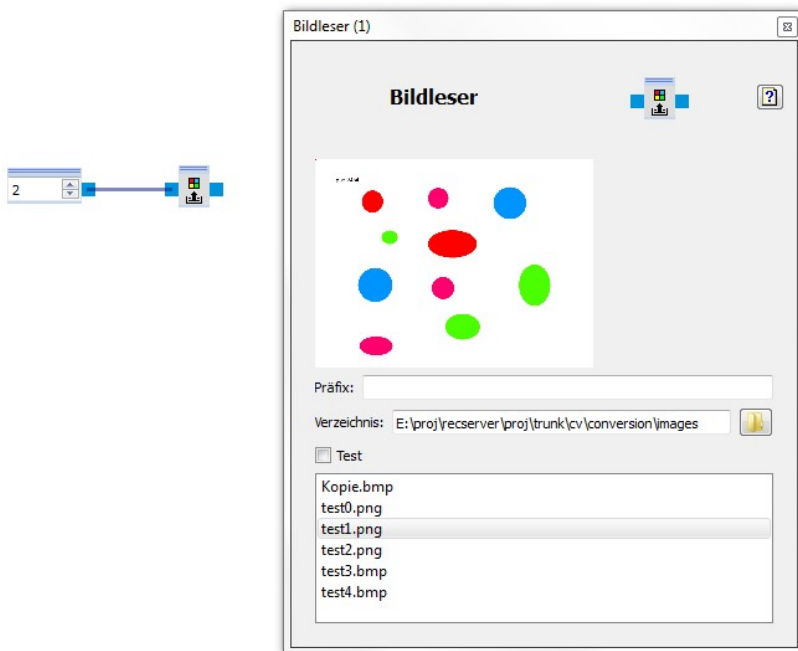
Beispiel



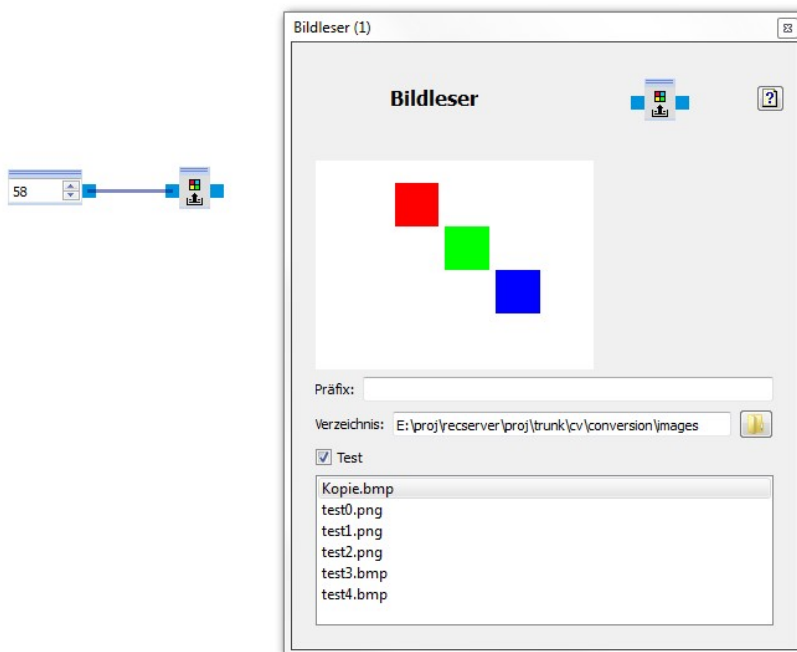
Zunächst gibt man im Dialog des Bildlesers das Verzeichnis an, in welchem sich die zu lesenden Bilder befinden. Mit einem leeren Präfix wird nach alle Bilder mit den Endungen ".bmp", ".png", ".jpg" und ".jpeg" gesucht. Sobald man das Programm startet, wird die Liste der verfügbaren Bilder erstellt und in dem Dialog angezeigt.



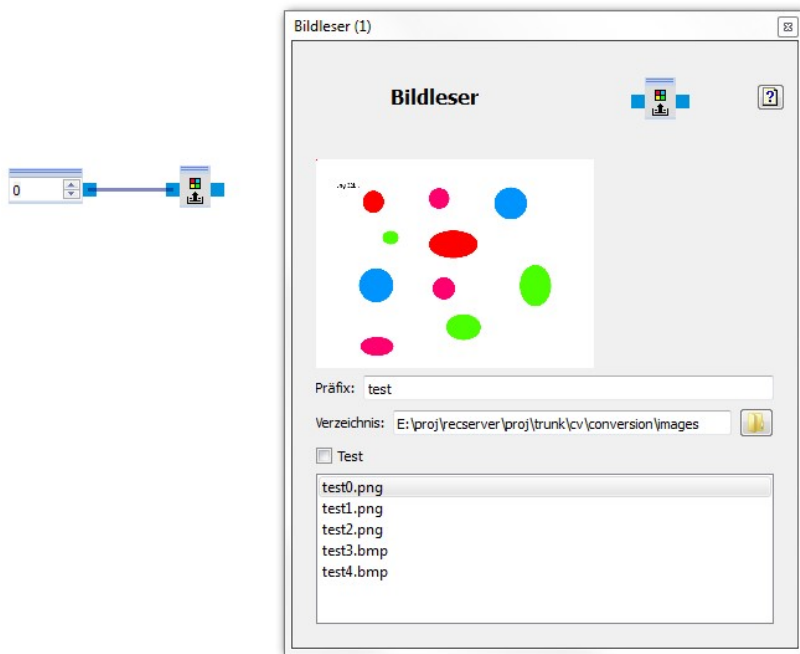
In diesem Beispiel wurden die Bilder "Kopie.bmp", "test0.png" bis "test4.bmp" gefunden. Die Bilder sind nach Dateinamen sortiert. Weil der Eingang des Bildlesers auf 0 gesetzt wird, ist "Kopie.bmp" das aktuelle Bild.



Ist der Eingang des Bildlesers 2, so ist das aktuelle Bild test1.png. Der aktuelle Bildindex berechnet sich zu $\text{index} = \text{"Eingang"} \bmod \text{"Anzahl der Bilder"}$



Schaltet man im Dialog auf "Test", so wird eine Testbild erzeugt, das sich mit der Nummer am Eingang verändert.



Durch die Angabe eines Präfix kann die Suche nach Bildern eingeschränkt werden. In diesem Fall werden nur noch die "test" Bilder gefunden. "Kopie.bmp" bleibt aufgrund des unpassenden Präfix außen vor.



Bildschreiber



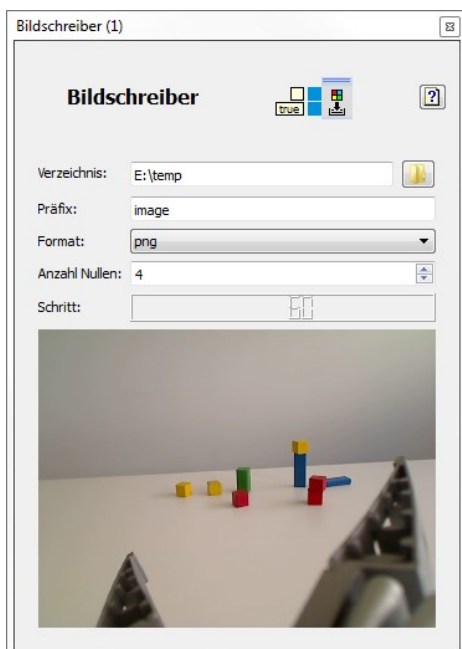
Der Bildschreiber schreibt eine Folge von Bildern in das Dateisystem. Pfad und Präfix können im [Dialog](#) eingestellt werden. Die Nummer des Bildes wird bei 0 beginnend in jedem Schritt um 1 hochgezählt.

Jedes einzelne Bild wird gespeichert unter dem Pfad "<Pfad>/<Präfix>_<Nummer>.jpg", wobei die Nummer mit führenden Nullen dargestellt wird.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Nächstes Bild der Bildfolge
Aktiv	bool	true	Der Bildschreiber ist aktiv.



Dialog



Von oben nach unten:

- Verzeichnis: Das Verzeichnis, in welches die Bilddateien geschrieben werden.
- Präfix: Der erste Teil des Namens der Bilddatei. Der komplette des ersten Bildes in diesem Beispiel ist image_0000.png.
- Format: Das Format der Bilddatei. Hier kann zwischen "png" und "bmp" gewählt werden.
- Anzahl Nullen: Die Feldbreite der Bildnummer.
- Schritt: Die Anzahl der bereits gespeicherten Bilddateien.
- Bildansicht: Anzeige des aktuellen Bildes.

FESTO



Beispiel

Lokale Kamera



Das Bild einer [lokalen Kamera](#) wird auf die Festplatte geschrieben.

FESTO



Werteleser



Liest zeilenweise aus einer Datei, trennt die Werte an Leerzeichen und gibt alle Werte in einer Zeile als Fließkommafeld aus.

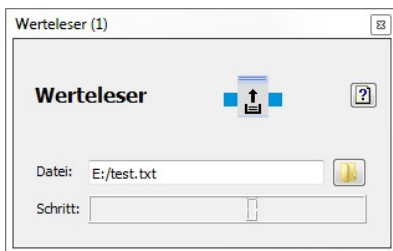
Nachdem die letzte Zeile eingelesen wurde, fängt das Einlesen wieder bei der ersten Zeile an.

Eingänge	Typ	Standard	Beschreibung
Aktiv	bool	true	Wenn wahr, dann wird in jedem Schritt eine neue Zeile eingelesen.
Ausgänge			
Ausgang	float array		Alle Werte innerhalb einer Zeile.

FESTO



Dialog



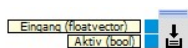
Von oben nach unten:

- Datei: Die Datei, aus der die Werte gelesen werden.
- Schritt: Die gerade eingelesene Zeile.

FESTO



Werteschreiber



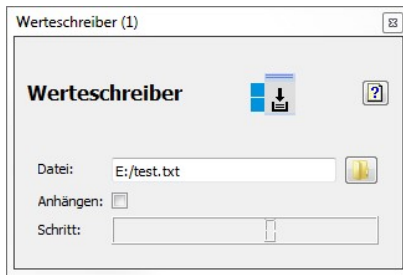
Schreibt die Eingangswerte mit Tab getrennt in eine neue Zeile in eine Datei.

Eingänge	Typ	Standard	Beschreibung
Eingang	float array		Die zu schreibenden Daten.
Aktiv	bool	true	Wenn wahr, dann wird in jedem Schritt eine neue Zeile geschrieben.

FESTO



Dialog



Von oben nach unten:

- Datei: Die Datei, in welche die Werte geschrieben werden.
- Anhängen: Wenn ausgewählt, werden die Werte an eine bestehende Datei angehängt. Ansonsten wird bei Programmstart eine bereits bestehende Datei gelöscht.
- Schritt: Die gerade geschriebene Zeile.

FESTO



Benutzerdefiniert

Diese Klasse enthält Funktionsblöcke, die vom Benutzer selbst mit Hilfe der Skriptsprache Lua programmiert werden können.

FESTO



Lua-Skript



Mit Hilfe von Lua-Skripten können auf einfache Weise nahezu beliebige Berechnungen durchgeführt werden, die mit den verfügbaren Funktionsblöcken nicht oder nur mit großem Aufwand realisiert werden könnten. Die Eingabe des Skripts erfolgt im Dialog des Funktionsblocks. Zu Beginn der Programmausführung wird das Skript, sofern es frei von Fehlern ist, zur Leistungssteigerung in schnell interpretierbaren Bytecode übersetzt.

Jeder Lua-Skript-Funktionsblock kann keinen bis maximal 20 Eingänge und mindestens einen bis höchstens 20 Ausgänge besitzen. Alle Ein- und Ausgänge verwenden als Datentyp Fließkommazahlen einfacher Genauigkeit.

Die Dokumentation der Sprache Lua ist unter der Adresse <http://www.lua.org/manual/5.1/> zu finden.

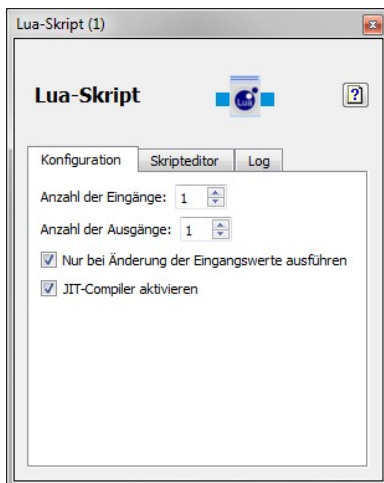
Komplexere Aufgaben, für die der Funktionsumfang von Lua oder die Rechenleistung nicht ausreicht, können alternativ in C++ [als neue Funktionsblöcke realisiert werden](#).

FESTO



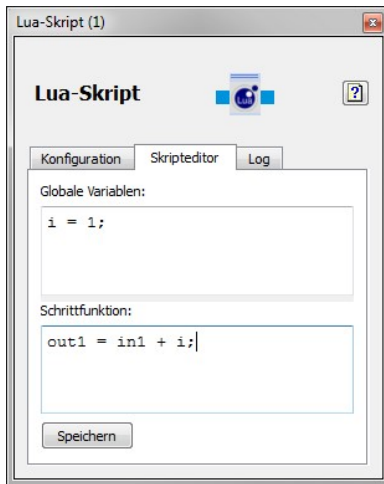
Dialog

Der Dialog besitzt drei Tab-Reiter. Der erste wird zur Konfiguration des Funktionsblocks verwendet:



Hier kann neben der Anzahl der Ein- und Ausgänge eingestellt werden, ob das Skript in jedem Schritt ausgeführt werden soll oder nur, wenn sich einer der Eingangswerte ändert. Zudem kann der JIT-Compiler, der standardmäßig eingeschaltet ist, deaktiviert werden. Wenn der JIT-Compiler deaktiviert ist, dann wird das Skript nicht zu Bytecode übersetzt, sondern zeilenweise interpretiert. Da dies unweigerlich Leistungseinbußen mit sich bringt, sollte der JIT-Compiler nur zu Testzwecken oder zur Fehlersuche deaktiviert werden.

Im Skripteditor wird das Lua-Skript eingegeben:



Der Editor ist in zwei Bereiche aufgeteilt: im oberen Bereich werden Variablen deklariert und initialisiert, die ihre Werte über die einzelnen Ausführungsschritte hinweg behalten sollen. Im unteren Bereich wird die eigentliche Berechnung implementiert. Die Ein- und Ausgänge des Funktionsblocks sind dabei über die lokalen Variablen "in1" bis "in20" und "out1" bis "out20" zugänglich. Ausgaben können mit dem Befehl "print" erzeugt werden. Nach einem Mausklick auf "Speichern" wird das Skript übernommen; das Programm muss dafür nicht neu gestartet werden.

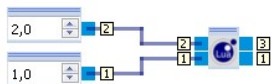
Im dritten Tab "Log" werden die Meldungen des Lua-Interpreters bzw. des Lua-JIT-Compilers sowie alle im Skript mit "print" erzeugten Ausgaben angezeigt.



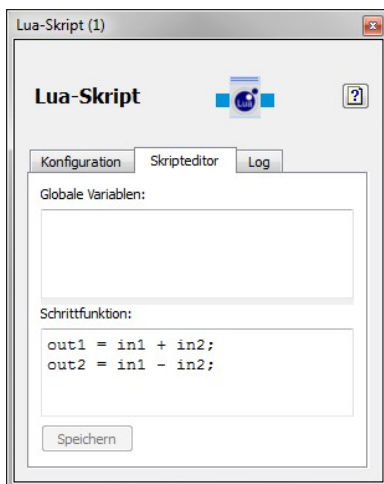
Beispiel



Das Skript in diesem Beispiel liest die Werte der Eingänge 1 und 2 und gibt deren Summe an Ausgang 1 und deren Differenz an Ausgang 2 aus. Das Skript wird nur bei einer Änderung der Eingangswerte ausgeführt.



Das zugehörige Skript sieht folgendermaßen aus:



Variablen



Eine Sonderstellung nehmen die globalen Variablen ein. Für alle globalen Variablen stehen in jedem Unterprogramm Funktionsblöcke zum Lesen und Schreiben zur Verfügung. Diese Funktionsblöcke zeigen in Unterprogrammen immer den Variablennamen an und können nicht umbenannt werden.

Globale Variablen können in der [Variablenverwaltung \(Hauptprogramm-Ansicht\)](#) angelegt, gelöscht, umbenannt und mit Startwerten belegt werden.

Anlegen, Löschen und Umbenennen von globalen Variablen ist auch in der Funktionsblockbibliothek möglich durch Klick mit der rechten Maustaste auf das Gerät "Variablen" und Auswahl des Befehls "Hinzufügen" bzw. Klick auf den Leser oder Schreiber einer globalen Variable und Auswahl des Befehls "Entfernen" oder "Umbenennen".



Veraltete Funktionsblöcke



Hier finden sich Funktionsblöcke, die zwar noch mit alten Programmen geladen werden können, aber nicht mehr gepflegt werden. In der Beschreibung eines jeden veralteten Funktionsblocks wird auf einen Ersatz hingewiesen.



Ausschalten





Ausschalten von Robotino.

Eingänge	Typ	Standard	Beschreibung
Ausschalten	bool	false	Wenn wahr (true), schaltet Robotino sich ab.

FESTO



Segmentierer



Dieser Funktionsblock befindet sich nur noch zur Wahrung der Abwärtskompatibilität in der Bibliothek. Bitte verwenden Sie statt dessen den Funktionsblock [Farbbereichssuche](#).

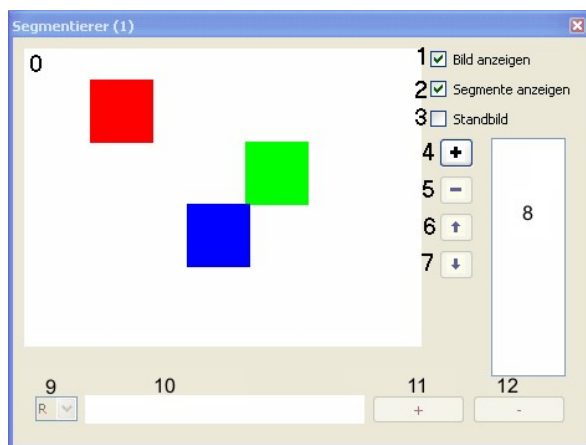
Der Segmentierer findet Flächen gleicher Farbe in einem Bild.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgänge			
Ausgang	image		segmentiertes Bild

FESTO



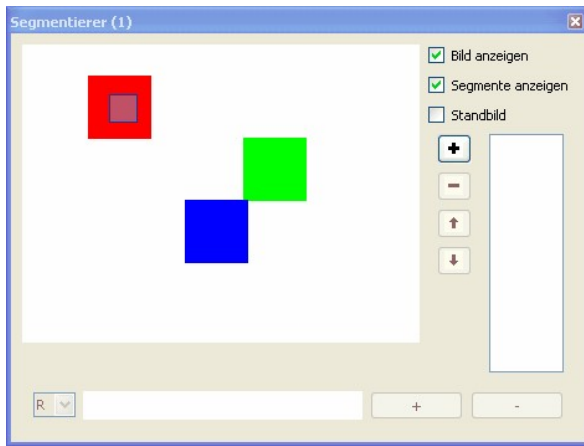
Dialog



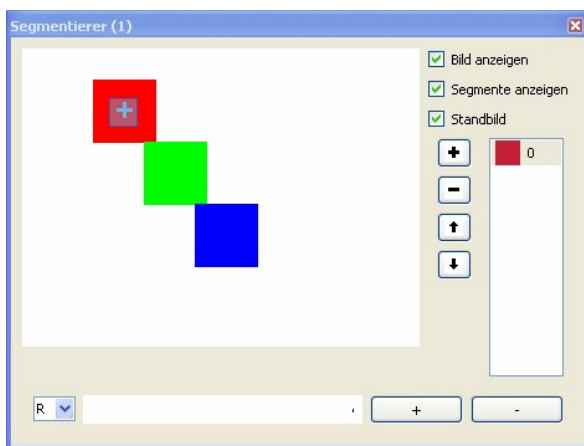
Schaltfläche/AnzeigeBeschreibung

- 0 Anzeige Eingangsbilds und/oder der Segmente
- 1 Wenn aktiviert, dann wird das Eingangsbild angezeigt
- 2 Wenn aktiviert, werden Segmente angezeigt
- 3 Friert das Eingangsbild ein
- 4 Fügt eine Auswahl im Bild als Segment hinzu
- 5 Löscht ein Segment aus der Liste der Segmente
- 6 Schiebt ein Segment in der Liste nach oben
- 7 Schiebt ein Segment in der Liste nach unten
- 8 Liste der Segmente
- 9 Auswahl des Farbkanals für die Segmentoptimierung
- 10 Zeigt die Werte innerhalb eines Kanals des ausgewählten Segments
- 11 Verbreitert die einzelnen Werte innerhalb des gewählten Kanals des ausgewählten Segments
- 12 Dünnt die Werte innerhalb des gewählten Kanals des ausgewählten Segments aus

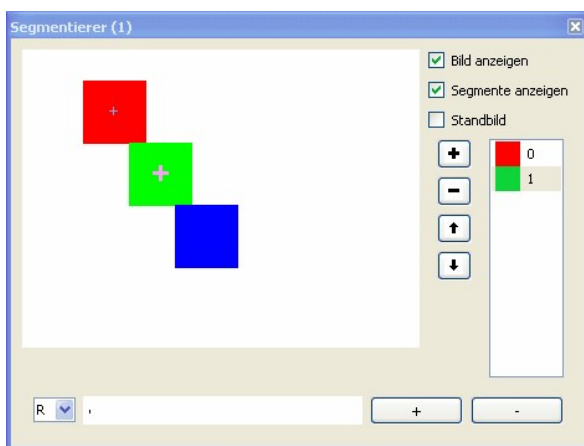
Um das rote Quadrat als zusammenhängende Fläche zu erkennen, markieren Sie einen Bereich innerhalb des Quadrats mit der Maus.



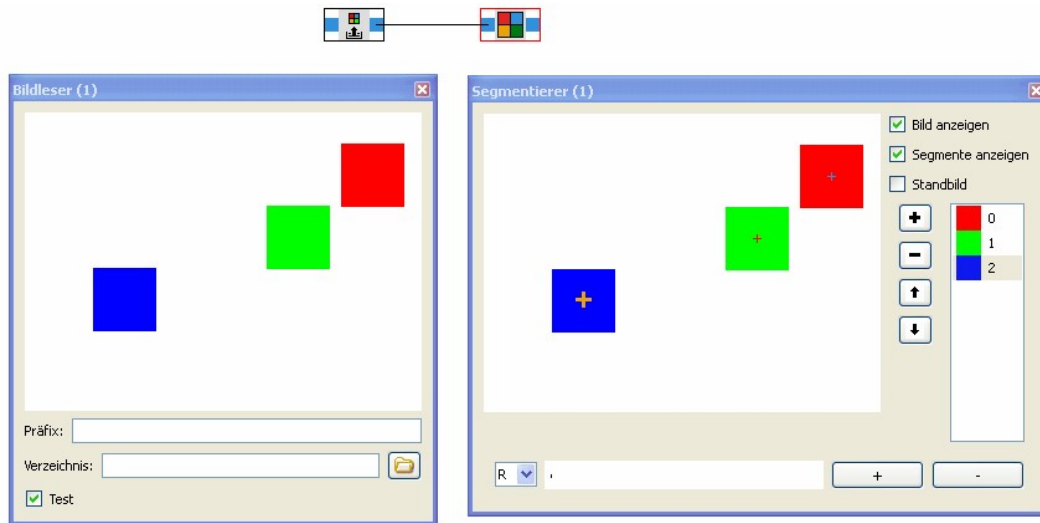
Klicken Sie nun auf + (Schaltfläche 4), um die Auswahl als Segment zu speichern.



Der Schwerpunkt des ausgewählten Segments wird im Bild als dickes Kreuz dargestellt. In einem bewegten Bild wandert das Kreuz jetzt mit dem roten Quadrat mit (vorher Standbild deaktivieren). Markieren Sie nun einen Bereich innerhalb des grünen Quadrats (dazu Standbild wieder deaktivieren) und fügen die Auswahl als Segment hinzu.



Nun sind zwei Segmente in der Liste. Der Schwerpunkt des grünen Segments wird als dickes Kreuz angezeigt, weil nun das grüne Segment ausgewählt ist.

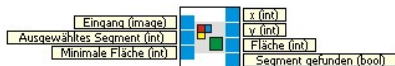


Der Bildleser ist im Test-Modus und erzeugt eine Sequenz von Bildern mit sich bewegendem rotem, grünem und blauem Quadrat. Zu jeder dieser Farben hat der Segmentierer ein Segment abgespeichert. Die Schwerpunkte der Quadrate werden im Segmentierer angezeigt.

FESTO



Segment Extrahierer



Dieser Funktionsblock befindet sich nur noch zur Wahrung der Abwärtskompatibilität in der Bibliothek. Bitte verwenden Sie statt dessen den Funktionsblock [Segmentverfolger](#).

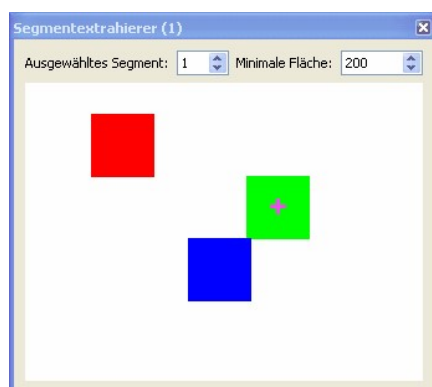
Liefert die Position und Größe eines Segmentes in einem zuvor segmentierten Bild.

Eingänge	Typ	Standard	Beschreibung
Eingang	image		Eingangsbild
Ausgewähltes Segment	int	0	Nummer des Segments, nach dem gesucht werden soll
Minimale Fläche	int	200	Das Segment muss mindestens aus der angegebenen Anzahl von Pixeln bestehen, damit der Ausgang "Segment gefunden" wahr (true) wird.
Ausgänge			
x	int		x-Koordinate in Pixel des Schwerpunkts des gefundenen Segments
y	int		y-Koordinate in Pixel des Schwerpunkts des gefundenen Segments
Fläche	int		Anzahl der Pixel aus denen das gefundene Segment besteht
Segment gefunden	bool		Wahr (true), wenn das Segment gefunden wurde, ansonsten unwahr (false).

FESTO



Dialog



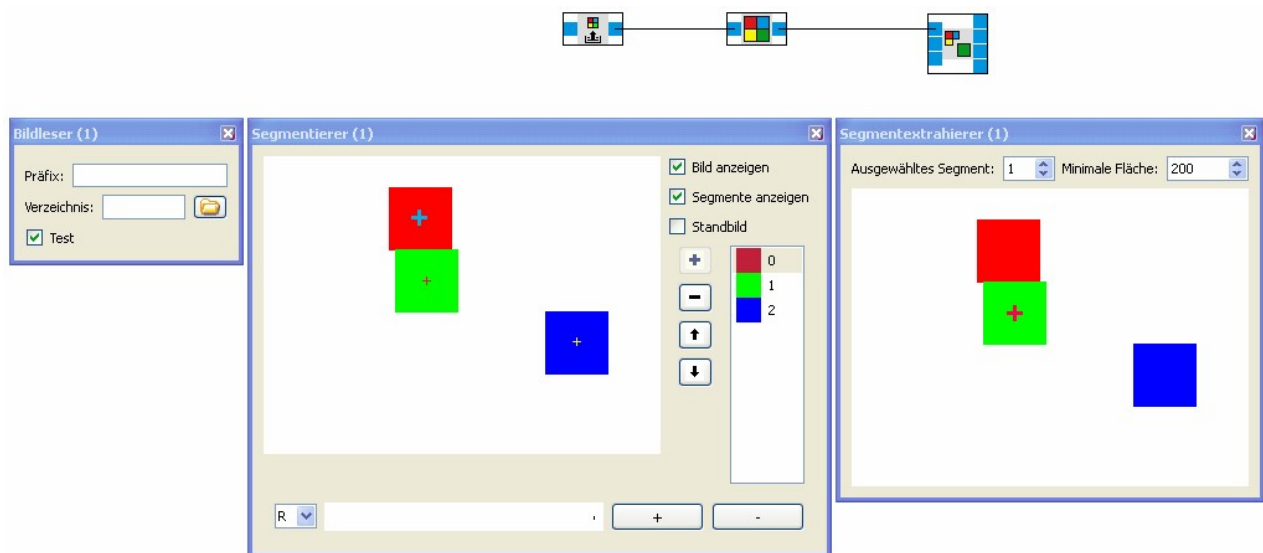
Ausgewähltes SegmentAktiv, wenn der gleichnamige Eingang unbeschaltet ist. Nummer des Segments, nach dem gesucht werden soll.

Minimale Fläche Aktiv, wenn der gleichnamige Eingang unbeschaltet ist. Das Segment muss mindestens aus der angegebenen Anzahl von Pixeln bestehen, damit der Ausgang "Segment gefunden" wahr (true) wird.

Zeigt die im Eingangsbild vorhandenen Segmente. Das ausgewählte Segment wird (falls gefunden) durch ein + markiert.

FESTO

Beispiel



Der Bildleser erzeugt eine Testsequenz mit drei farbigen Quadraten. Der Segmentierer durchsucht das Bild nach roten, grünen und blauen Flächen. Der Segmentextrahierer sucht das Segment mit der Nummer 1 (das grüne Segment) und markiert den Schwerpunkt mit einem Kreuz.