

Ablaufprogramme

In diesem Kapitel wird ein einfaches Ablaufprogramm mit Alternativverzweigungen realisiert.

Tutorial 2

Die Übung zeigt, wie ein Ablaufprogramm mit Alternativzweig erstellt wird. Das fertige Programm befindet sich unter [examples/sfc/tutorial2.rvw2](#)

Das fertige Ablaufprogramm sieht wie in Abbildung 1 dargestellt aus.

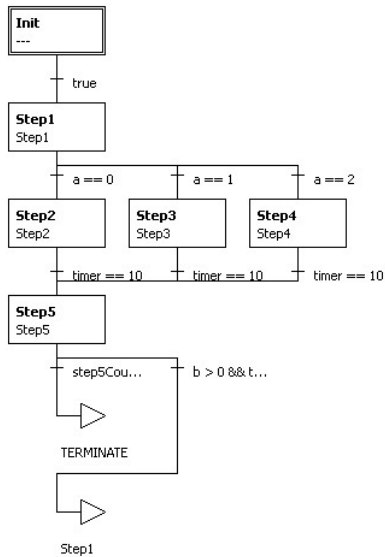



Abbildung 1: Das vollständige Ablaufprogramm

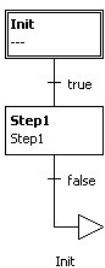
In Step1 wird der Wert von a verändert. Dadurch wird in jedem Durchlauf des Programms einer der Schritte Step2, Step3 und Step4 ausgeführt. Step5 vergleicht von den vorherigen Schritten gelieferte Ergebnisse. Nach dem sechsten Durchlauf von Step5 wird das Programm angehalten. Ansonsten wird mit Step1 fortgefahren.

[Ein neues Projekt anlegen](#)

Legen Sie ein neues Projekt an, in dem Sie

- den Menüpunkt Datei ► Neu auswählen
- die Tastenkombination Strg+N drücken oder
- in der Werkzeugleiste das Symbol für das Anlegen eines neuen Projektes auswählen .

Das Hauptprogramm enthält die beiden Schritte Init und Step1.



[Globale Variablen anlegen](#)

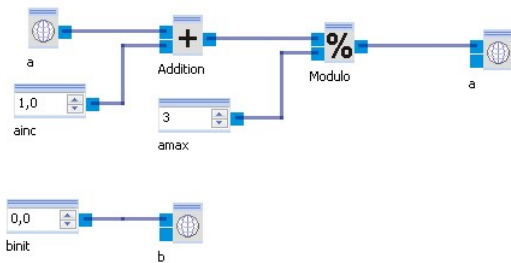
Legen Sie zunächst folgende [globalen Variablen](#) an:

- timer
- a
- b
- step2count
- step3count
- step4count
- step5count

Belegen Sie Variable "a" mit dem Startwert -1. Alle anderen Variablen haben den Startwert 0.

Step1 programmieren

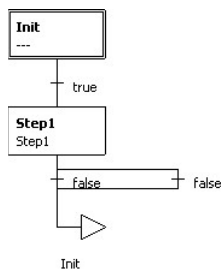
In diesem Unterprogramm wird die globale Variable "a" um 1 erhöht. Damit sich der Wert von "a" in den Grenzen 0 bis einschließlich 2 bewegt, wird "a" Modulo 3 genommen und dann wieder nach "a" geschrieben. Der Wert von "b" wird lediglich auf 0 gesetzt.

Die Schritte Step2, Step3 und Step4 anlegen

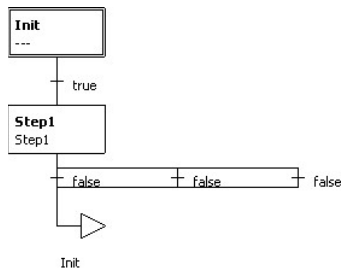
Nun werden die Schritte in einem Alternativzweig nebeneinander angelegt. Markieren Sie dazu die Transitionsbedingung unterhalb von Step1.

Dass die Transitionsbedingung markiert ist, erkennen Sie an der gestrichelten Linie um die Bedingung herum.

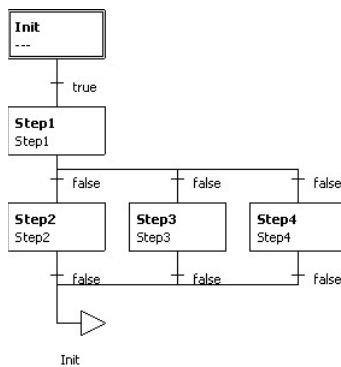
Klicken Sie jetzt auf das Symbol für das Einfügen eines Alternativzweiges nach rechts \rightarrow .



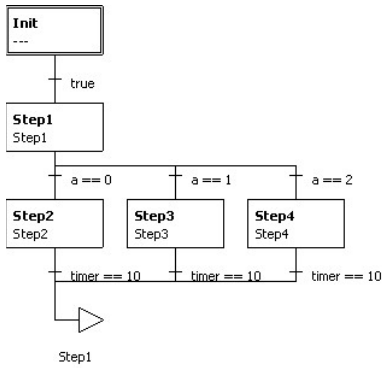
Erweitern Sie die erzeugte Verzweigung, in dem Sie die rechte Transitionsbedingung markieren und erneut das Alternativzweig nach rechts Einfügen \rightarrow Symbol anklicken.



Erzeugen Sie nun in den drei Alternativzweigen drei Schritte, die Sie Step2, Step3 und Step4 nennen. Dazu markieren Sie die Eingangsbedingung eines Zweiges und klicken auf das Schritt einfügen danach \downarrow Symbol. Ordnen Sie danach den neu erzeugten Schritten gleichnamige Unterprogramme zu. Dazu doppelklicken Sie auf den jeweiligen Schritt und geben im folgenden Dialogfeld den Namen des Unterprogramms ein. Alternativ können Sie mit der Schaltfläche "Neues Unterprogramm" ein neues Unterprogramm erstellen und danach dem jeweiligen Schritt zuordnen.



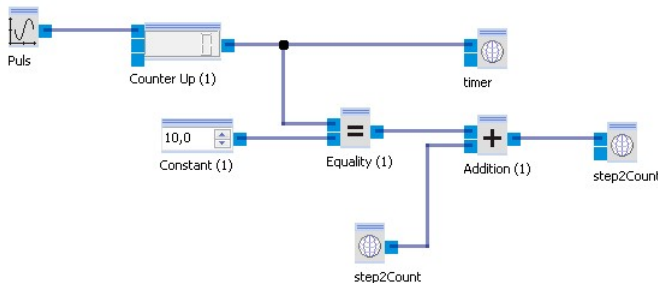
Die Eingangs und Ausgangsbedingungen aller drei Alternativzweige sind zur Zeit false. Ändern Sie die Eingangsbedingungen auf $a == 0$, $a == 1$ und $a == 2$. Als Ausgangsbedingung nehmen Sie für alle Zweige $timer == 10$. Ändern Sie den abschließenden Sprung noch von Init auf Step1.



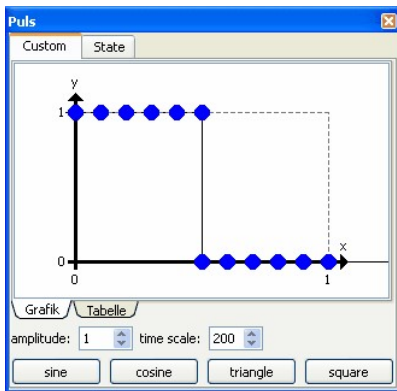
Wenn Sie das Hauptprogramm jetzt starten, bleibt das Programm in Step2 hängen, weil "a" beim ersten Durchlauf 0 ist und die globale Variable "timer" nicht verändert wird.

[Die Schritte Step2, Step3 und Step4 programmieren](#)

Die den Schritten Step2 bis Step4 zugeordneten Unterprogramme sind z.Zt. leer. Nachfolgend ist das zu erstellende Unterprogramm Step2 gezeigt.



Der Arbiträrgenerator erzeugt alle 200ms einen 100ms breiten Puls der Höhe 1. Nachfolgend sind die Einstellungen des Arbiträrgenerators gezeigt.




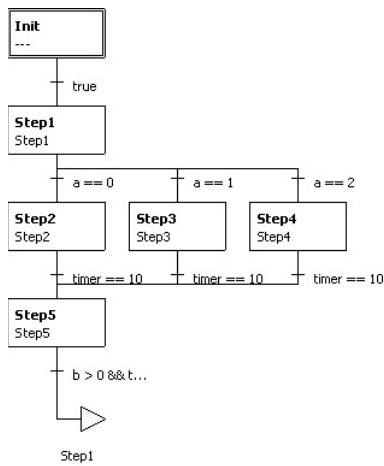
D.h. alle 200ms erfolgt ein Flankenwechsel von 0 nach 1. Der Zähler erhöht bei jedem Flankenwechsel seinen Wert um 1. Nach 2s ist der Wert des Zähler daher 10. Wenn der Wert des Zähler 10 ist, wird zu dem aktuellen Wert von step2Count das Ergebnis des Vergleichs zwischen der Konstante und dem Zählerwert addiert. Solange der Vergleich falsch liefert, wird 0 addiert. Sobald der Vergleich wahr ist, wird 1 addiert. Zum Abschluss jedes Berechnungsschrittes des Unterprogramms, wird die im Hauptprogramm folgende Transitionsbedingung ausgewertet. Wenn die globale Variable "timer" den Wert 10 hat, wird das Unterprogramm verlassen.

Die Unterprogramme der Schritte Step3 und Step4 sind äquivalent aufgebaut. Markieren Sie alles im Unterprogramm Step2 (Strg+A) und kopieren es nach Step3 und Step4. Der einzige Unterschied besteht darin, dass step3count bzw. step4count gelesen und geschrieben werden.

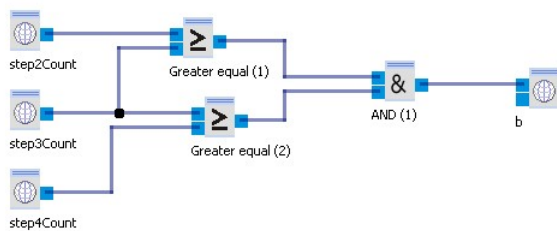
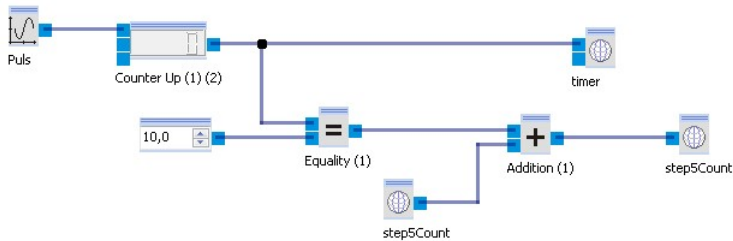
Startet man das Hauptprogramm durch Klick auf das Symbol , so werden zyklisch die Schritte Step2, Step3 und Step4 für jeweils 2s ausgeführt.

[Step5 anlegen und programmieren](#)

Um nach der Alternativverzweigung einen neuen Schritt anzulegen, markieren Sie den abschließenden Sprung und klicken das Symbol zum Anlegen eines Schrittes davor . Erstellen Sie ein Unterprogramm mit dem Namen Step5 und ordnen Sie es dem gerade erzeugten Step5 zu. Ändern Sie die Transitionsbedingung unterhalb von Step5 auf $b > 0 \ \&\& \ \text{timer} == 10$.



Das Unterprogramm Step5 ähnelt den Programmen Step2 bis Step4. Kopieren Sie Step2 nach Step5 und ändern Sie step2count in step5count. Neben dem Setzen der globalen Variablen "timer" und "step5count" wird geprüft, ob die Bedingung $\text{step2count} \geq \text{step3count} \geq \text{step4count}$ gilt. Wenn ja, wird die globale Variable "b" auf 1 gesetzt. Ansonsten ist "b" 0. Die Bedingung muss bei korrektem Programmablauf aber immer wahr sein, da die Schritte Step2, Step3 und Step4 nacheinander ausgeführt werden, weil Step1 die globale Variable "a" in jedem Zyklus um 1 erhöht.



Wenn Sie das Hauptprogramm jetzt starten, bleibt Step5 2s aktiv, falls "b" größer 0 gilt.

[Programmabbruch und Sprung zu Step1 anlegen](#)

Jetzt soll noch das Programm abgebrochen werden, wenn die globale Variable "step5count" den Wert 6 erreicht. Um dies zu erreichen fügen Sie einen Alternativzweig unterhalb von Step5 ein. Markieren Sie die Transitionsbedingung unterhalb von Step5 und klicken Sie auf das Symbol für das Einfügen eines Alternativzweiges links . Markieren Sie die Transitionsbedingung des neuen Zweiges (die Bedingung ist z.Zt. false) und klicken Sie auf das Symbol zum Einfügen eines Sprunges . Ändern Sie die Transitionsbedingung in $\text{step5count} == 6$ und wählen Sie als Sprungziel "TERMINATE".

Das Hauptprogramm sieht nun aus, wie am Anfang gezeigt.

Der Alternativzweig mit dem Sprung nach TERMINATE muss übrigens links von dem Zweig mit der bisherigen Bedingung $b > 0 \ \&\& \ \text{timer} == 10$ stehen, weil die Anfangsbedingungen eines Alternativzweiges von links nach rechts ausgewertet werden. In den ersten 6 Durchläufen ist die Bedingung $\text{step5count} == 6$ nicht erfüllt, so dass die Bedingung des zweiten Zweiges geprüft wird.

Ein Durchlauf des Hauptprogramms dauert nun 24s.

FESTO



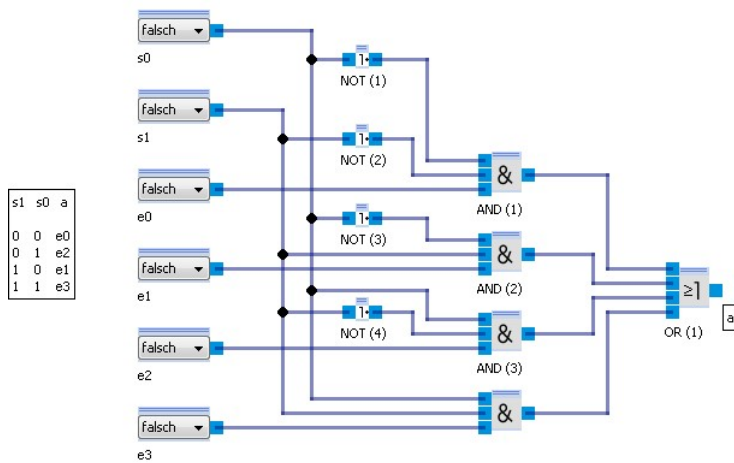
Logik

In diesem Kapitel werden bekannte elektrische Schaltungen mit Logik-Bausteinen realisiert.

FESTO



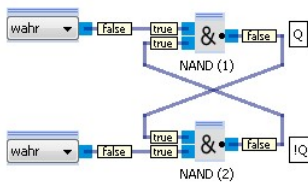
Multiplexer



FESTO



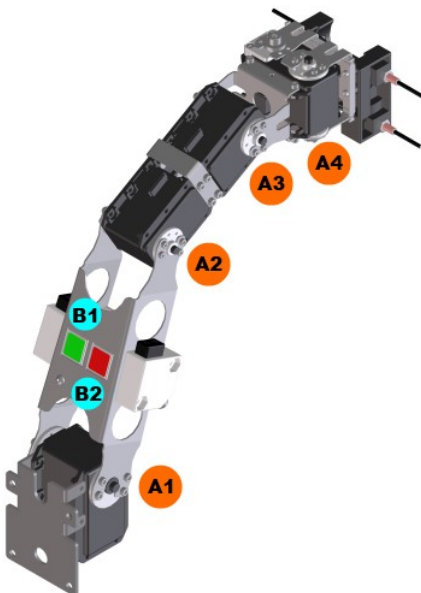
FlipFlop



FESTO



Elektrischer Greifarm für Robotino®



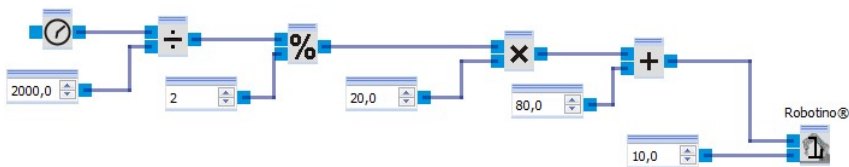
Das Bild zeigt den elektrischen Greifarm für Robotino®. Der Arm besteht aus vier Achsen (A1 bis A4). Über die Druckknöpfe B1 und B2 kann der Benutzer Eingaben direkt am Greifarm durchführen.

Druckknopf	Funktion
B1	Speichern der aktuellen Konfiguration
B2	Umschalten zwischen normalem Betrieb und stromlosem Zustand

Gespeicherte Positionen (siehe auch [Manipulatordialog](#)) können mit dem [Positions Auswahl](#) Funktionsblock angefahren werden. Die Positions Auswahl nimmt entweder die Nummer der Position oder den Positionsnamen entgegen.



Die Achsen des elektrischen Greifarms können auch einzeln angesteuert werden. Dies geschieht mit dem [Achse-Schreiber](#) Funktionsblock. Das folgenden Beispiel lässt A1 alle 2s zwischen 80° und 100° wechseln.



Informationen zu den einzelnen Achsen erhält man mit dem [Achsen-Leser](#) Funktionsblock.

FESTO



Bildverarbeitung

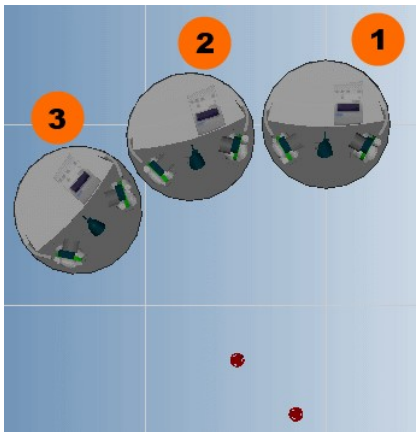
Geben Sie hier den Text ein.

FESTO

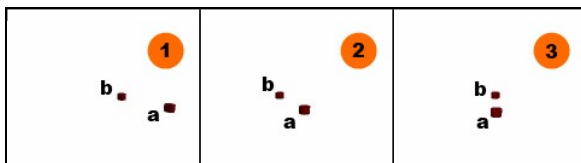


Farbverfolgung

Im folgenden wird vorgestellt, wie eine Kombination aus [Farbbereichssuche](#), [Maximum-Filter](#) und [Segmentverfolger](#) genutzt werden kann, um Robotino an zwei roten auf dem Boden liegenden Werkstücken auszurichten.



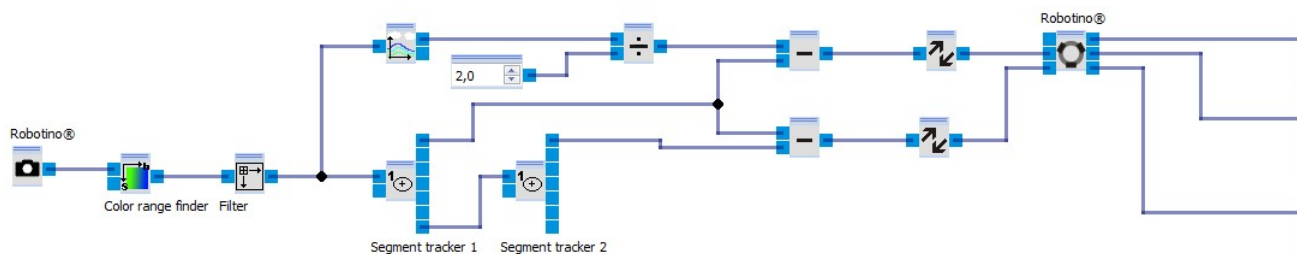
Bei Programmstart steht Robotino an Position 1 und bewegt sich dann zu der Endposition 3. Die von Robotinos Kamera gelieferten Bilder sehen an den Drei Position wie folgt aus.



In der Ausgangsstellung liegt das Robotino nächste Werkstück (a) am rechten Bildrand. Über den Abstand von Werkstück (a) zur Bildmitte kann eine Bewegung von Robotino in y-Richtung gesteuert werden, so dass sich Werkstück (a) in die Bildmitte bewegt.

Gleichzeitig wird eine Drehbewegung ausgeführt, so dass der x-Abstand der beiden Werkstücke im Bild minimiert wird.

Die Bildauswertung und die Bewegung von Robotino kann mit folgendem Robotino View Programm bewerkstelligt werden.

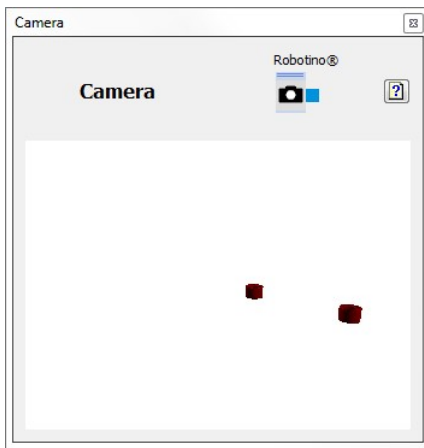


Durch klicken auf die Funktionsblöcke gelangt man zu weiter führenden Informationen in Bezug auf die jeweilige Funktion der Funktionsblöcke in dem Beispielprogramm. Das Beispiel unter [examples/colortracking.rvw2](#)

FESTO



Kamera



Robotino [Kamera](#) liefert das Ausgangsbild.

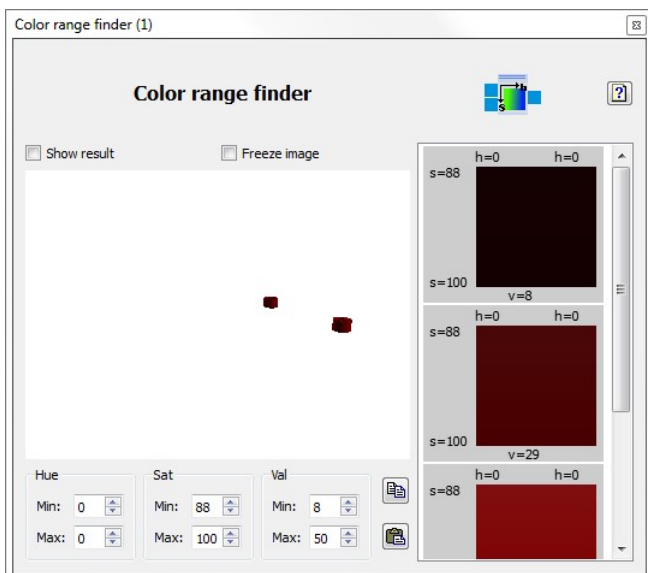
[zurück zum Beispiel](#)

FESTO



Farbbereichssuche

In der [Farbbereichssuche](#) wird ein Werkstück markiert, so dass die Farbe rot inkl Abstufungen den Bereich der zu suchenden Farben ausmacht.



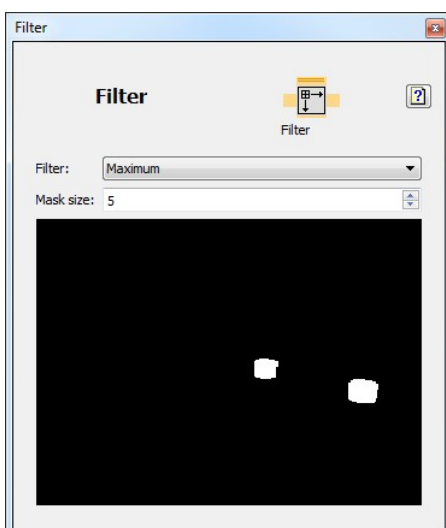
Das Ausgangsbild der [Farbbereichssuche](#) hat jetzt weiße Pixel dort, wo sich die Werkstücke befinden. Der Rest des Bildes ist schwarz.

[zurück zum Beispiel](#)

FESTO



Filter



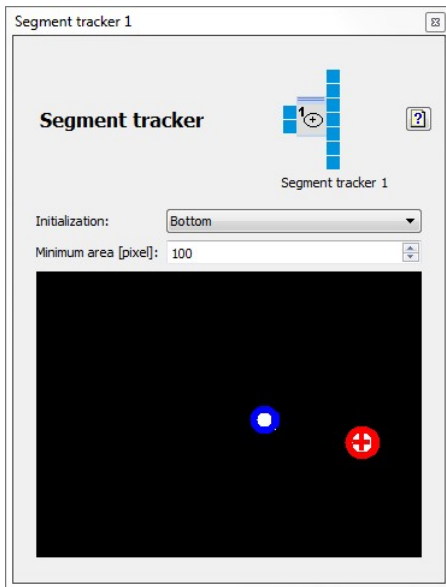
Ein [Maximum-Filter](#) vergrößert die weißen Bereiche, so dass selbst bei starker Variation der Beleuchtung zusammenhängende Bereiche für jedes Werkstück bestehen bleiben.

[zurück zum Beispiel](#)

FESTO



Segmentverfolgung 1



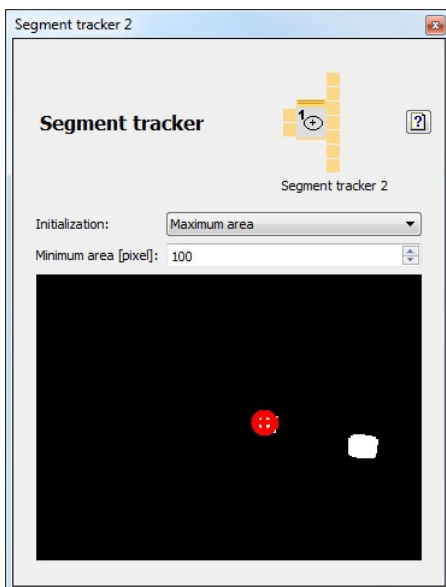
Der erste [Segmentverfolger](#) findet das zu Programmstart dem unteren Bildrand nächste Segment und verfolgt dieses.

[zurück zum Beispiel](#)

FESTO



Segmentverfolgung 2



Der zweite [Segmentverfolger](#) findet das zweite Werkstück und verfolgt dieses.

[zurück zum Beispiel](#)

FESTO

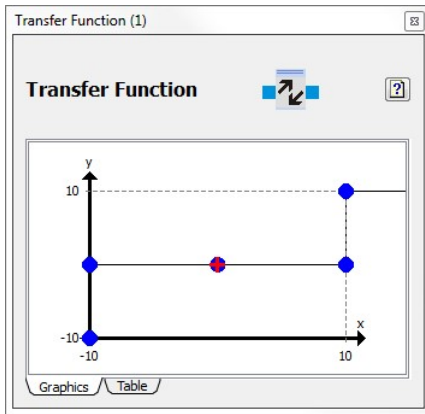


Berechnung der y-Bewegung

Die [Bildinformation](#) liefert die Breite und Höhe des Bildes. Die Breite wird genutzt, um den Abstand von Werkstück (a) zur Bildmitte zu berechnen.

Die x-Koordinate der Bildmitte liegt bei Bildbreite / 2. Die [Konstante](#) liefert den konstanten Divisor.

Durch [Subtraktion](#) der x-Koordinate von Werkstück (a) von der halben Bildbreite erhält man den Abstand von Werkstück (a) zur Bildmitte.



Die [Transferfunktion](#) übersetzt den Abstand zur Bildmitte in Pixel in die Stellgröße für die y-Bewegung.

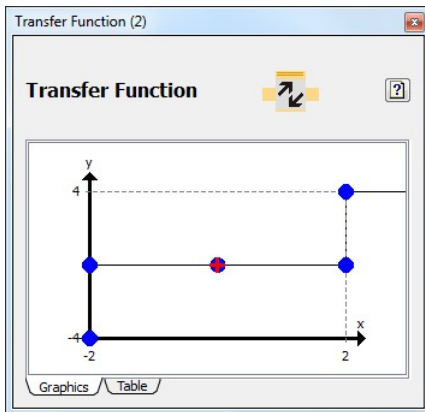
[zurück zum Beispiel](#)

FESTO



Berechnung der Rotation

Durch [Subtraktion](#) der x-Koordinate von Werkstück (b) von x-Koordinate von Werkstück (a) erhält man den x-Abstand der beiden Werkstücke.



Die [Transferfunktion](#) übersetzt den Abstand der Werkstücke in Pixel in die Stellgröße für die Rotation.

[zurück zum Beispiel](#)

FESTO



Antrieb

Die Stellgrößen werden über den [Omniantrieb](#) an die [Motoren](#) von Robotino geleitet.

[zurück zum Beispiel](#)

FESTO

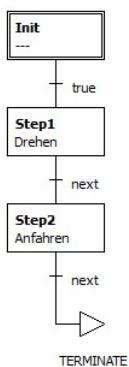


Suchen und annähern

In diesem Beispiel wird ein Robotino View Programm erstellt, welches über Robotinos Kamera ein farbiges Objekt findet und sich diesem nähert. Das Programm gliedert sich in zwei Unterprogramme.

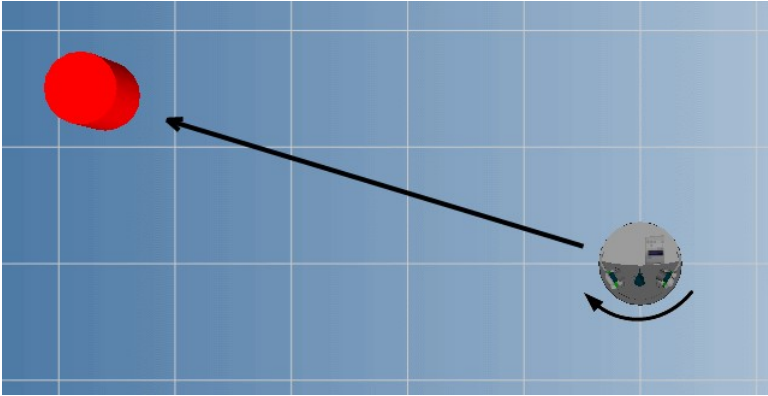
- Drehen: In diesem Unterprogramm dreht sich Robotino zu dem farbigen Objekt
- Anfahren: In diesem Unterprogramm fährt Robotino auf das farbige Objekt zu und stoppt, sobald die von dem Objekt eingenommene Fläche im Kamerabild eine Grenze überschreitet.

Das Hauptprogramm sieht wie unten gezeigt aus:



Aus dem leeren Init-Schritt springt das Hauptprogramm sofort in den Schritt "[Step1](#)". In diesem Schritt bleibt das Hauptprogramm solange die globale Variable "next" gleich 0 ist. Das Unterprogramm "[Drehen](#)" setzt "next" auf 1, wenn sich Robotino zu dem farbigen Objekt gedreht hat.

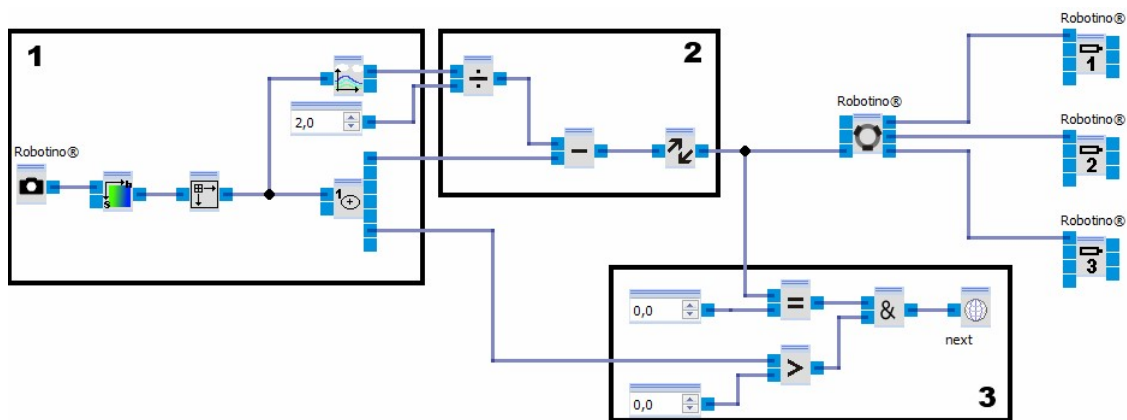
Das Unterprogramm "[Anfahren](#)" setzt "next" zunächst wieder auf 0, so dass "[Step2](#)" aktiv bleibt. Wenn Robotino sich dem farbigen Objekt weit genug angenähert hat, wird "next" durch das Unterprogramm "[Anfahren](#)" auf 1 gesetzt und das Hauptprogramm wird durch den Sprung zu "Terminate" beendet.



FESTO



Drehen

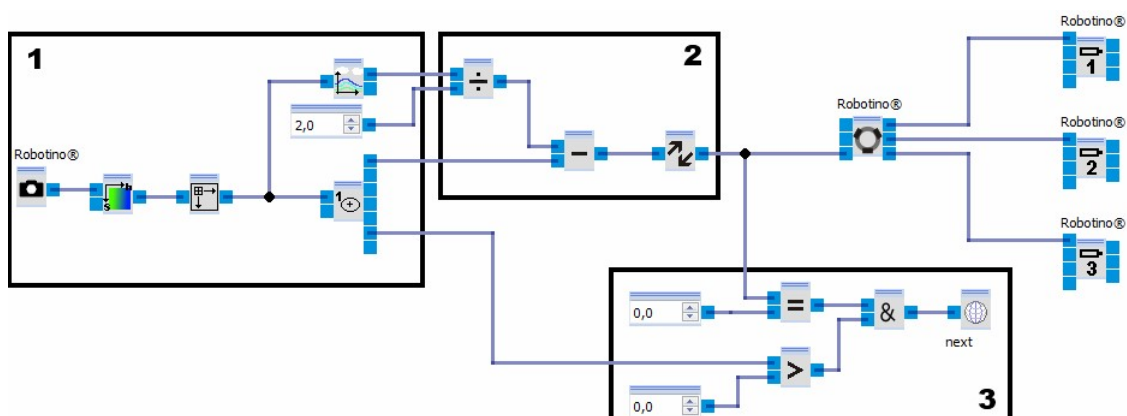


1. [Auswertung des Kamerabildes](#)
2. [Berechnung des Stellgröße für die Rotationsgeschwindigkeit](#)
3. [Überprüfung der Abbruchbedingung](#)

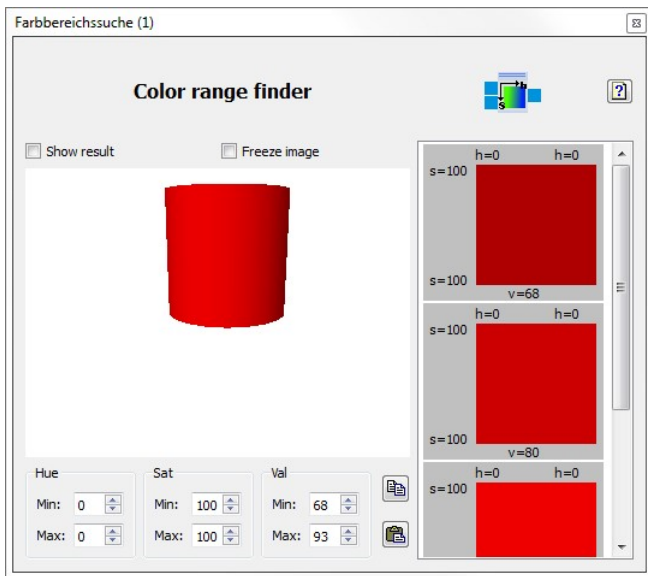
FESTO



Bildauswertung (1)

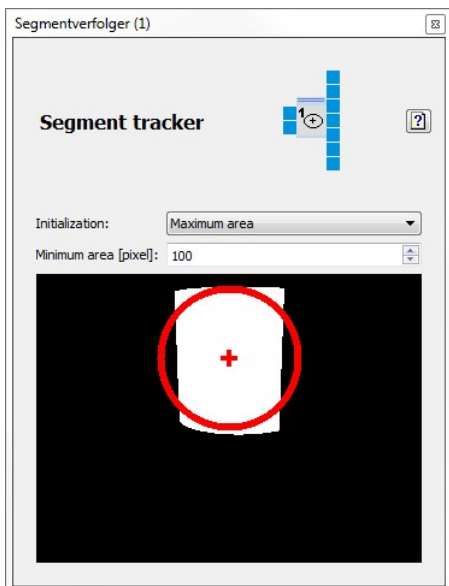


Aus dem Kamerabild wird in der [Farbbereichssuche](#) das farbige Objekt (in diesem Beispiel der rote Zylinder) extrahiert.



Bei dem nachgeschalteten [Filter](#) handelt es sich um einen Maximum-Filter, um eventuell separierte Bereiche zusammenzufassen.

Der [Segmentverfolger](#) ermittelt schließlich die Koordinaten des roten Zylinders im Bild.

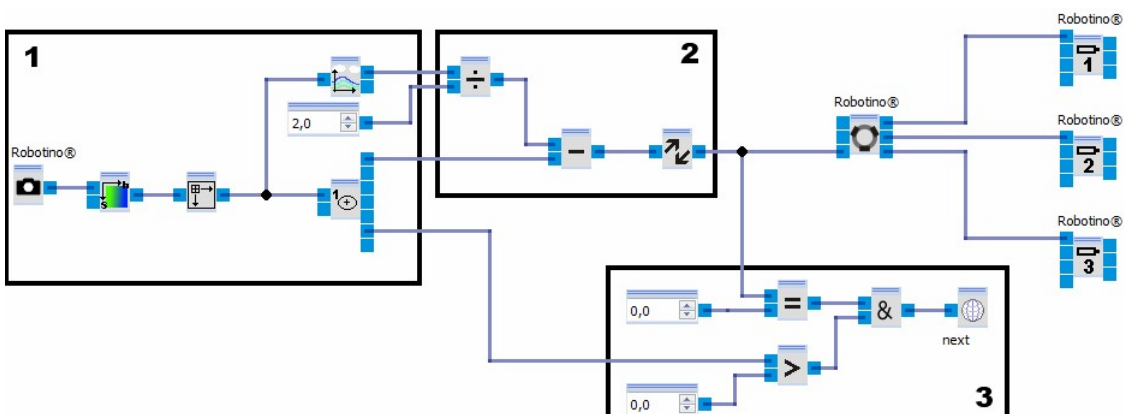


Die [Bildinformation](#) gibt die Höhe und Breite des Bildes nach dem Filter aus. Diese Werte werden für die Berechnung der [Stellgröße](#) benötigt.

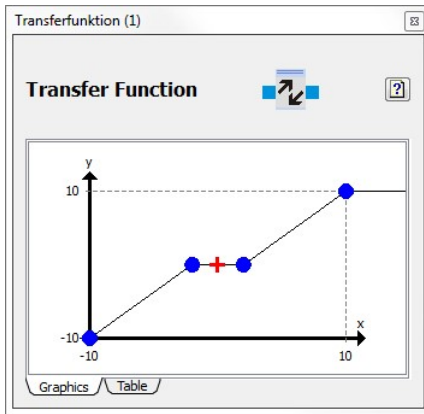
FESTO



Stellgröße (2)



Die Stellgröße für die Rotation berechnet sich aus der halben Bildbreite minus der x-Koordinate des roten Zylinders im Bild. Danach bildet eine [Transferfunktion](#) diese Werte in der Einheit Bildpunkte auf die Rotationsgeschwindigkeit in Grad pro Sekunde ab.



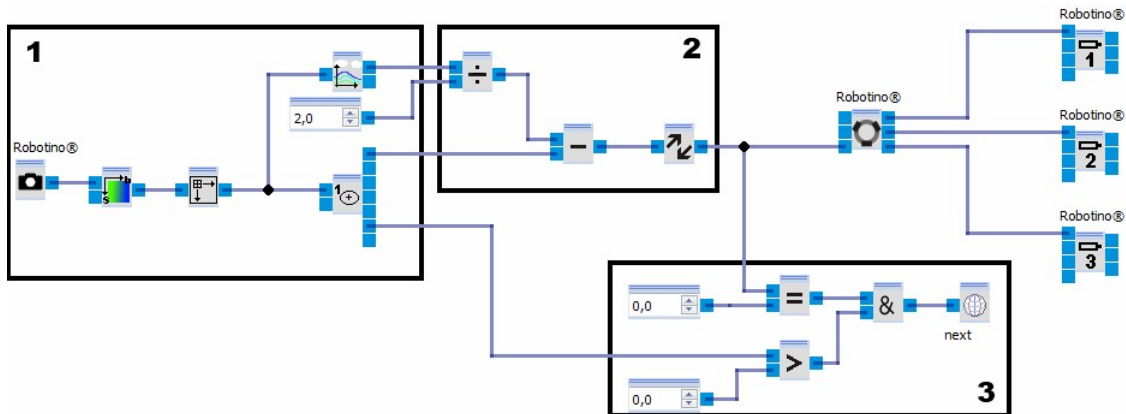
Der Eingangswert der [Transferfunktion](#) ist der Abstand des roten Zylinders zu Bildmitte in Pixel. Ist dieser Abstand kleiner -10 Pixel, so rotiert Robotino mit -10 deg/s. Bis zu einem Abstand von -2 Pixel nimmt die Rotationsgeschwindigkeit auf 0 ab. Ab 2 Pixel Abweichung steigt die Rotationsgeschwindigkeit bis maximal 10 deg/s bei einer Abweichung von 10 Pixel.

Wenn der rote Zylinder nicht sichtbar ist, gibt der [Segmentverfolger](#) x-Koordinate 0 aus. Der Eingangswert der [Transferfunktion](#) ist in diesem Fall die halbe Bildgröße, so dass sich Robotino mit 10 deg/s dreht.

FESTO



Abbruchbedingung (3)



Das Unterprogramm wird beendet wenn der globalen Variable "next" ein Wert ungleich 0 zugewiesen wird.

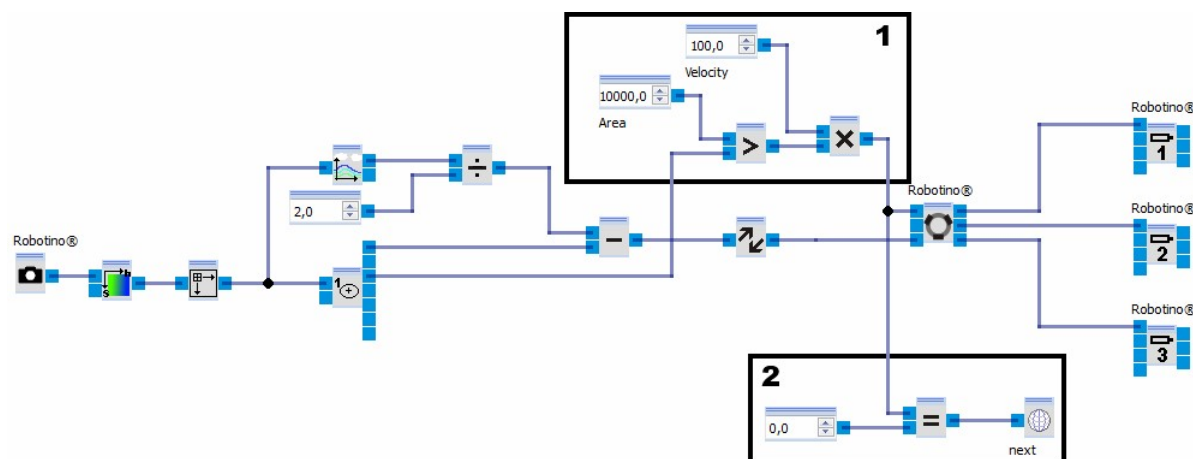
Wenn die Stellgröße für die Rotationsgeschwindigkeit 0 ist, dann befindet sich der rote Zylinder in der Bildmitte, d.h. Robotino hat sich zu dem roten Zylinder hingedreht. Die Stellgröße wird mit 0 verglichen.

Es ist zu beachten, dass bei Programmstart nicht garantiert ist, dass ein Kamerabild verfügbar ist. Wenn noch kein Kamerabild empfangen wurde, liefert die [Bildinformation](#) eine Breite von 0. Der [Segmentverfolger](#) liefert ebenfalls 0 als x-Koordinate, so dass die Stellgröße in diesem Fall ebenfalls 0 ist. Die oben formulierte Bedingung ist also wahr und würde zu einem Abbruch des Unterprogramms führen, obwohl noch gar keine Bewegung durchgeführt wurde. Aus diesem Grund wird geprüft, ob der [Segmentverfolger](#) überhaupt ein Segment gefunden hat. Nur wenn die Anzahl der gefundenen Segmente größer als 0 ist, kann "next" auf wahr gesetzt werden.

FESTO



Anfahren



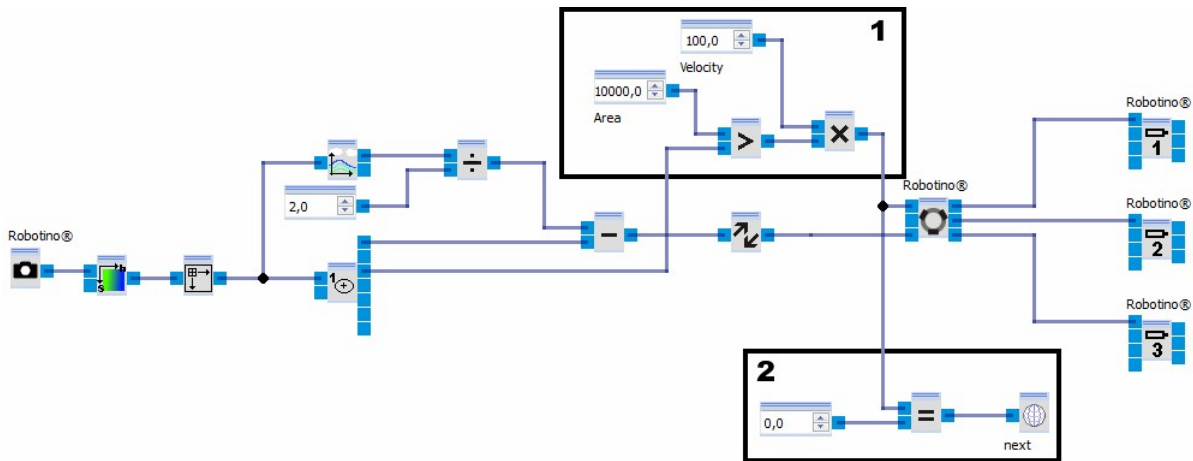
1. [Berechnung der Stellgröße für die Vorwärtsgeschwindigkeit](#)

2. [Berechnung der Abbruchbedingung](#)

Die Bildauswertung und die Berechnung der Stellgröße für die Rotationsgeschwindigkeit sind identisch zu den Berechnungen im Unterprogramm ["Drehen"](#).

FESTO

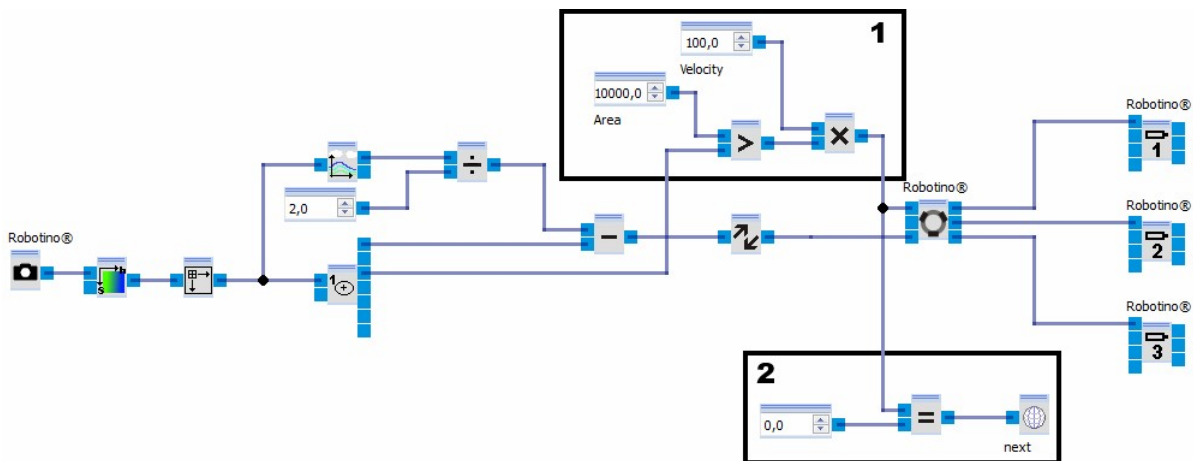
Stellgröße (1)



Als Grundlage der Berechnung der Stellgröße für die Vorwärtsgeschwindigkeit in mm/s dient die Fläche des roten Zylinders, welche von dem [Segmentverfolger](#) ausgegeben wird. Solange diese Fläche in Pixel kleiner als 10000 ist liefert der erste Vergleich wahr oder 1 als Fließkommazahl. Ist die Fläche größer als 10000 liefert der Vergleich unwahr bzw. 0. Durch Multiplikation mit 100 erhält man die Vorwärtsgeschwindigkeit.

FESTO

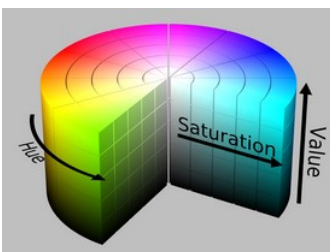
Abbruchbedingung (2)



Das Unterprogramm wird beendet, sobald die Vorwärtsgeschwindigkeit auf 0 sinkt. Dann ist der Zylinder so nah, dass er eine Fläche von mehr als 10000 Pixel im Bild einnimmt.

FESTO

HSV Farbraum



Quelle: http://en.wikipedia.org/wiki/File:HSV_color_solid_cylinder_alpha_lowgamma.png

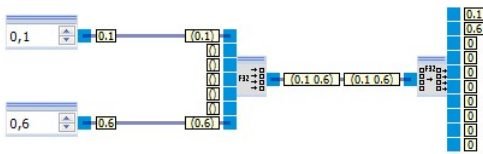
Im Gegensatz zum RGB-Farbraum, in welchem die Farbe über den Rot-, Grün- und Blauanteil definiert wird, legt der HSV-Farbraum den Farbton (Hue), die Farbsättigung (Saturation) und den Hellwert (Value) zugrunde.

- Der Farbton wird als Farbwinkel H angegeben. $H=0^\circ$ =rot. $H=120^\circ$ =grün, $H=240^\circ$ =blau.
- Farbsättigung S in Prozent. 0%=grau. 100%=reine Farbe.
- Hellwert V in Prozent. 0% keine Helligkeit. 100% volle Helligkeit.

FESTO

Felder



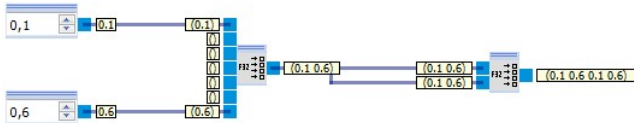


Mit dem Fließkommafeld-Zusammensetzer wird das Feld

0.1	0.6
-----	-----

erzeugt. Leere Eingänge am Fließkommafeld-Zusammensetzer werden nicht in das Ausgabefeld übernommen.

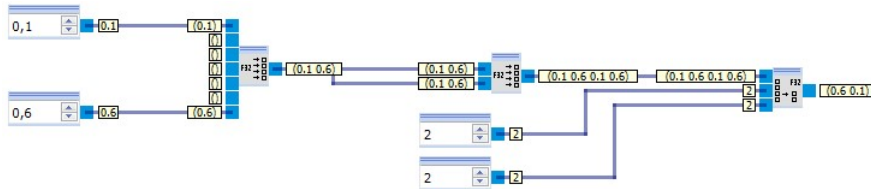
Der Fließkommafeld-Zerhacker bildet das Feld auf seine Ausgänge ab.



Der Fließkommafeld-Zusammensetzer setzt nicht nur einzelne Werte, sondern auch Felder zusammen. Legt man das erste Feld an beide Eingänge des Fließkommafeld-Zusammensetzers, dann erhält man das Feld

0.1	0.6	0.1	0.6
-----	-----	-----	-----

Der Fließkommafeld-Zerleger schneidet aus einem Feld ein Teilfeld heraus.

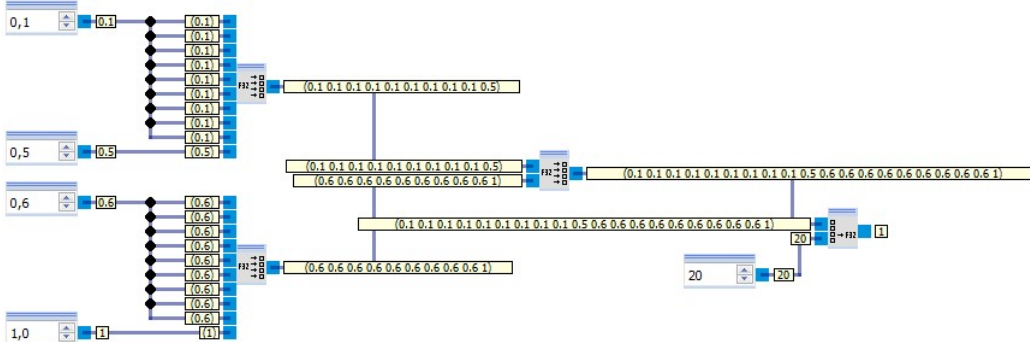


Hier wird aus dem Feld

0.1	0.6	0.1	0.6
-----	-----	-----	-----

ab dem Index 2 ein Feld der Länge 2 extrahiert.

Mit dem Indexzugriff kommt man auch an die einzelnen Werte sehr großer Felder heran.



Aus dem Feld der Größe 20 holt man sich hier den letzten Wert (1).