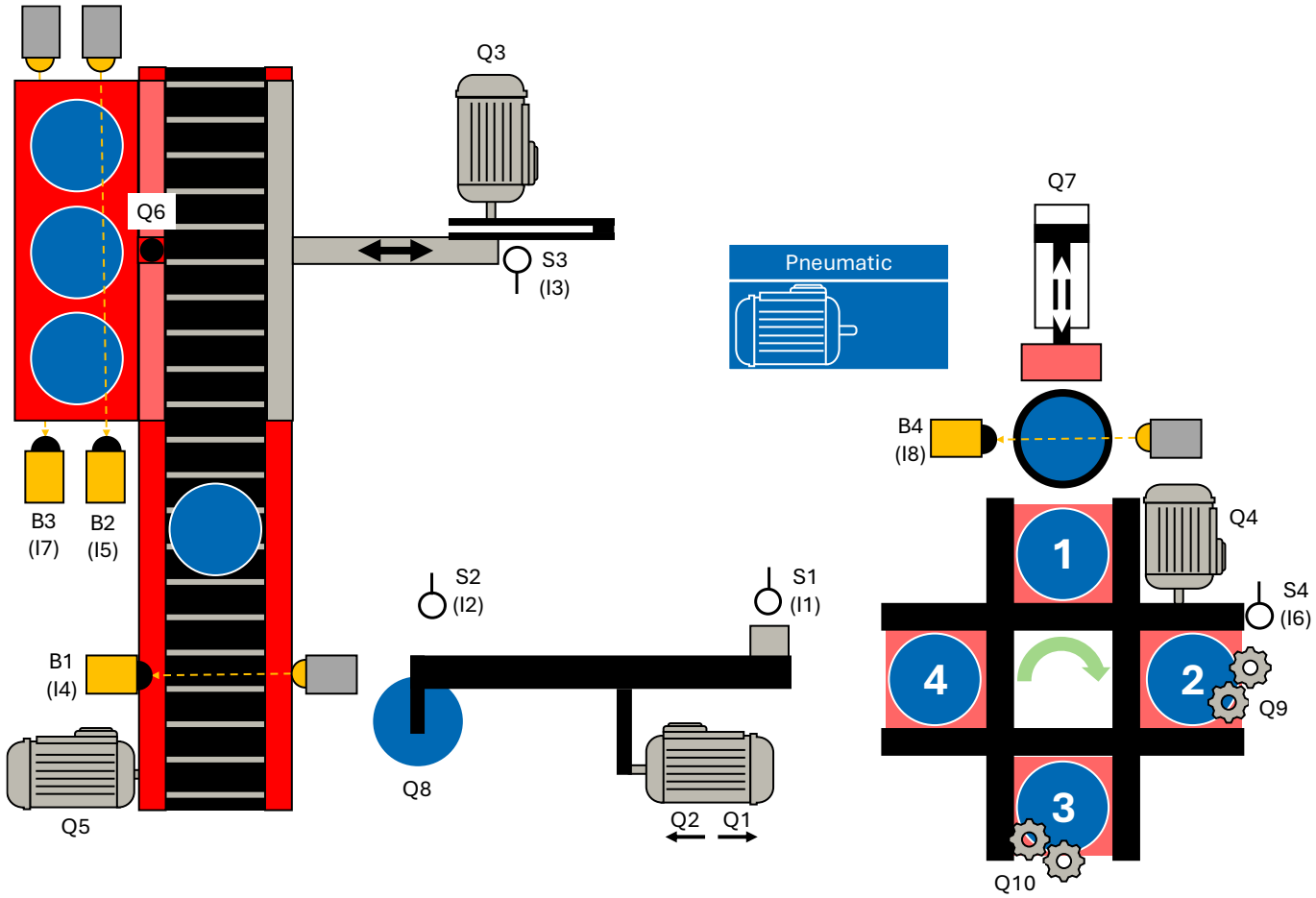
fischertechnik

GROLLMUS

WISSEN | SOFTWARE | AUTOMATISIERUNG

Fertigungslinie 24V

Umwandlung GRAFCET in Programmcode



Inhaltsverzeichnis

9 Umwandlung GRAFCET in Programmcode

9.1 Funktionsbeschreibung der Ablaufkette

Aus dem zu einer Beispielanlage gegebenen GRAFCET soll nun ein SPS-Programm erstellt werden. Das prinzipielle Vorgehen zur Wandlung eines GRAFCET in Programmcode wird im nachfolgenden zunächst an einer einfachen Schrittkette verdeutlicht:

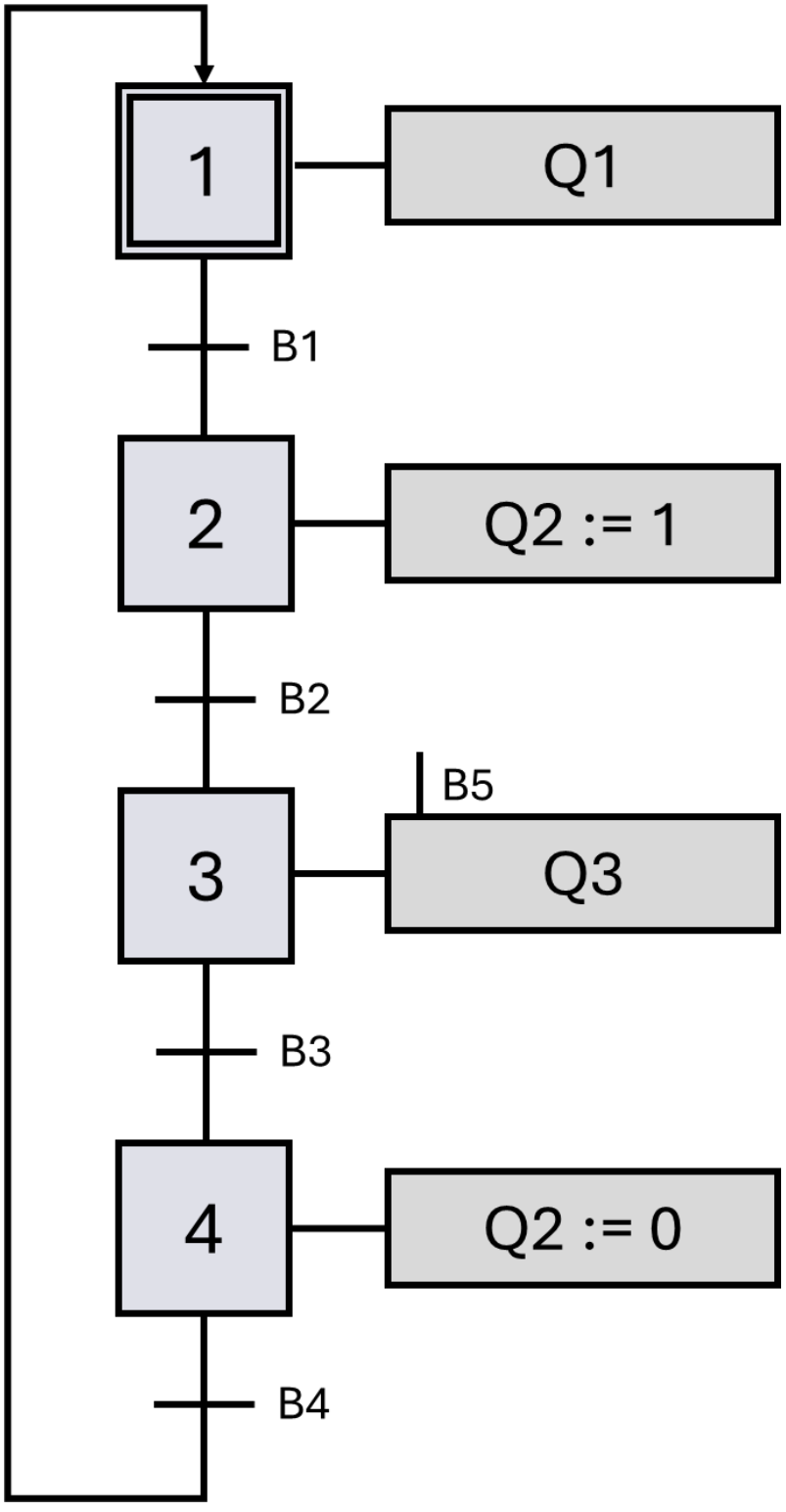


Bild 1 Exemplarische GRAFCET Kette

Funktionsbeschreibung

Schritt 1 (Initialschritt)

Bei Schritt 1 handelt es sich um den Initialschritt.
Solange die Anlage sich in diesem Schritt befindet, wird mittels kontinuierlich wirkender Aktion der Aktor Q1 angesteuert.
Liefert der Sensor B1 1-Signal, wird in den Folgeschritt gesprungen.

Schritt 2

Der Aktor Q2 wird speichernd aktiviert ("TRUE").
Liefert der Sensor B2 1-Signal, wird in den Folgeschritt gesprungen.

Schritt 3

Solange die Anlage sich in diesem Schritt befindet, wird mittels kontinuierlich wirkender Aktion mit Bedingung der Aktor Q3 angesteuert.
Damit der Aktor mit dem Wert "TRUE" angesteuert wird, muss zusätzlich zum aktiven Schritt, der Sensor B5 ebenfalls 1-Signal liefern.
Liefert der Sensor B3 1-Signal, wird in den Folgeschritt gesprungen.

Schritt 4

Der Aktor Q2 wird zurückgesetzt ("FALSE").
Liefert der Sensor B4 1-Signal, wird zurück in den Initialschritt gesprungen.

9.2 GRAFCET Ablaufkette in FUP umsetzen

In diesem Abschnitt werden wir die in GRAFCET geplante Ablaufkette in der Programmiersprache FUP umsetzen.

9.2.1 Schritte und Transitionen in FUP umsetzen

Das Rücksetzdominante Flipflop bildet das zentrale Element bei der Umsetzung der GRAFCET Ablaufkette in FUP. Ein aktiver Schritt wird durch ein aktiviertes Flipflop dargestellt. Die Aktivierung eines Schritts erfolgt über den Setzeingang und kann somit, als Teil der vorangehenden Transition betrachtet werden. Das Verlassen eines aktiven Schritts erfolgt über den Rücksetzeingang und gehört damit zur nachfolgenden Transition.
Als Speicher des Flipflops wird jeweils ein eigener Schrittmerker verschaltet, welcher den aktivierten Schritt repräsentiert.
Als Variable für den Schrittmerker können globale Merker, Variablen aus globalen Datenbausteinen oder lokale Variablen aus dem statischen Bereich der Bausteinschnittstelle verwendet werden.

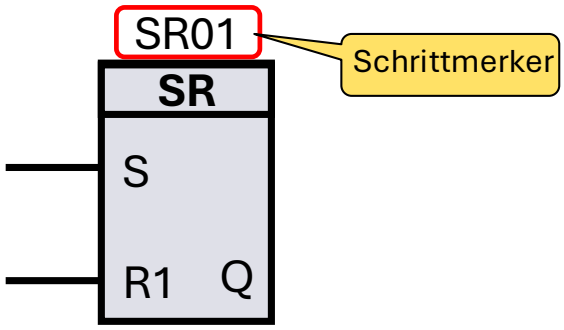


Bild 2 Flipflop mit Schrittmerker

Initialschritt
Bei Schritt 1 handelt es sich um den Initialschritt, dieser muss aktiviert werden, wenn die Kette initialisiert (Init) wird. Zusätzlich wird der Schritt auch aktiviert, wenn sich die Kette im letzten Schritt (SR04) befindet und die nachfolgende Transition (B4) erfüllt ist.

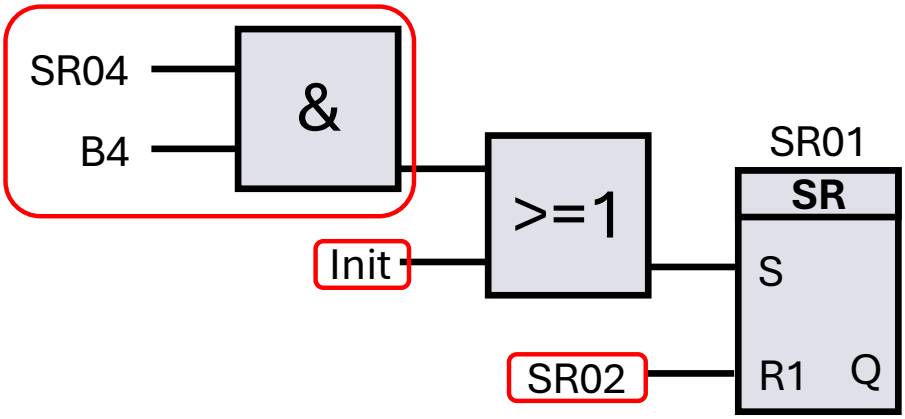


Bild 3 Programmierung Initalschritt

Der Schritt muss rückgesetzt werden, wenn der nachfolgende Schritt (SR02) aktiv ist.

Standard Schritt

Alle weiteren Schritte werden gesetzt, wenn der vorherige Schritt und die entsprechende Transition erfüllt ist. Sie werden rückgesetzt, wenn der nachfolgende Schritt aktiv ist, oder die Kette initialisiert wird.

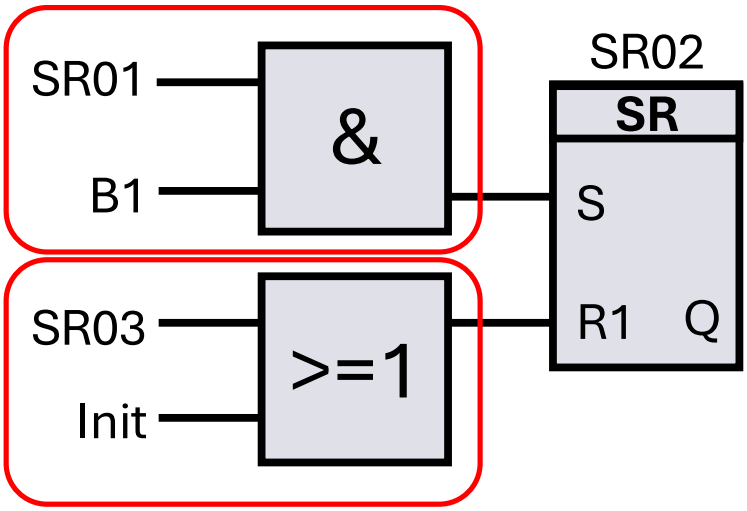


Bild 4 Programmierung Schritt

Nach dem letzten Schritt folgt ein Sprung zum ersten Schritt der Kette, somit wird der letzte Schritt mit dem ersten Schritt zurückgesetzt, da dies der nachfolgende Schritt ist.

Somit ergibt sich für den eingangs gezeigten GRAFCET folgende Schrittkette:

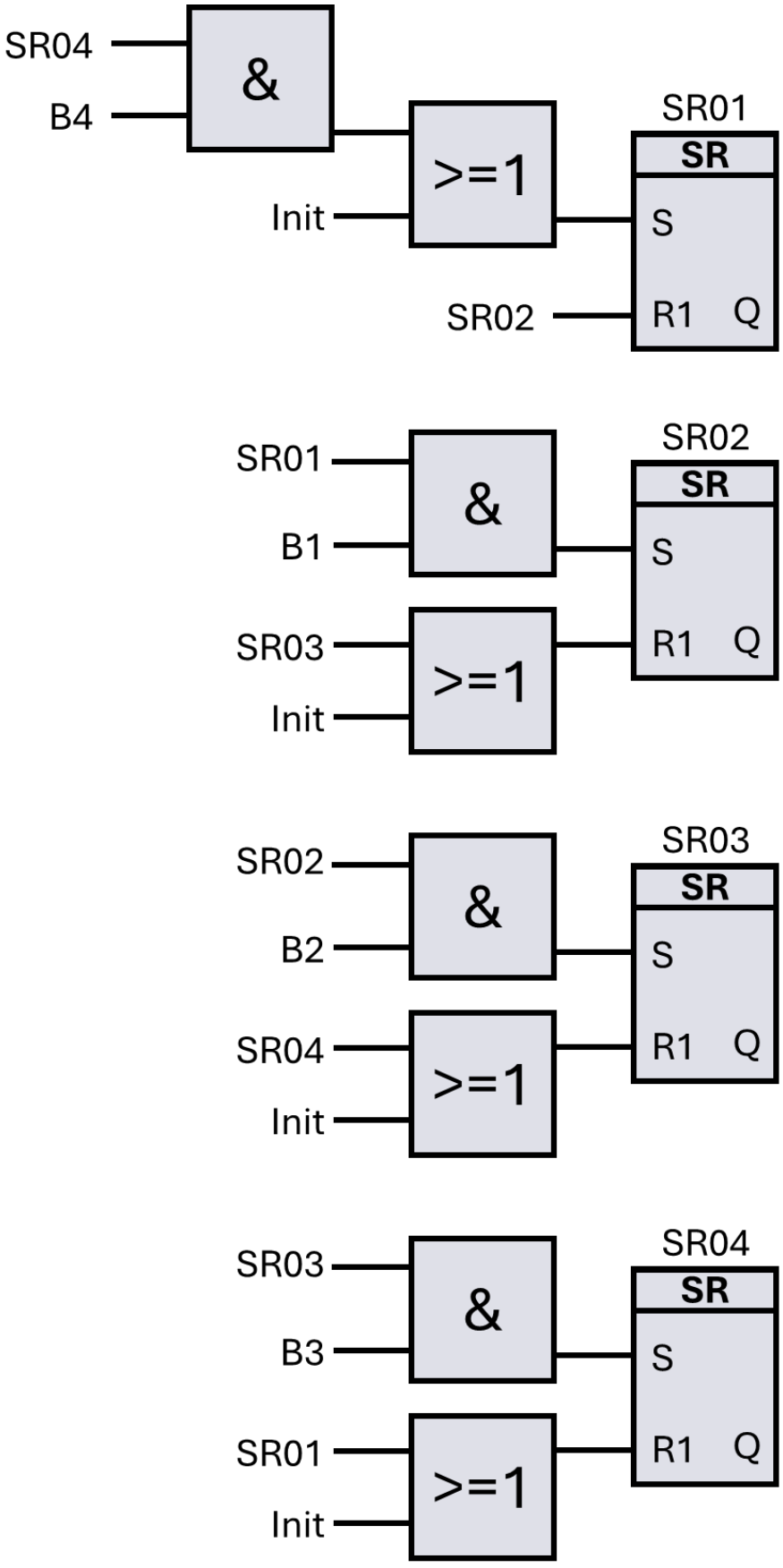


Bild 5 Programmierung Schrittkette

9.2.2 Aktionen in FUP umsetzen

Die Programmierung der Aktionen in den jeweiligen Schritten der Ablaufkette kann aufgrund von Gesichtspunkten der funktionalen Strukturierung entweder in einem separaten Ansteuerbaustein oder direkt in den Netzwerken unmittelbar nach der Kette im gleichen Baustein erfolgen.

Programmierung in einem separaten Ansteuerbaustein

Eine Möglichkeit besteht darin, die Aktionen der Schritte in einem separaten Ansteuerbaustein zu programmieren. Diese Methode hat den Vorteil, dass die Steuerungslogik der Schrittkette und die Ausführung der Aktionen klar voneinander getrennt sind. Dadurch wird die Übersichtlichkeit des Programms erhöht, was insbesondere bei komplexen Steuerungen von Vorteil ist. Zudem erleichtert diese Trennung die Wartung und Erweiterung des Programms, da Änderungen in der Steuerungslogik oder den Aktionen unabhängig voneinander vorgenommen werden können.

In diesem Ansatz wird im Hauptbaustein die Schrittkette implementiert. Wenn ein Schritt aktiv wird, wird ein Signal an den Ansteuerbaustein gesendet, der dann die entsprechenden Aktionen ausführt. Als Signale für den Austausch werden meist die Schrittmerker hergenommen. Diese Strukturierung ermöglicht eine modulare Programmierung, bei der jeder Baustein eine klar definierte Aufgabe hat.

Programmierung in den Netzwerken direkt nach der Schrittkette

Eine alternative Methode ist die Programmierung der Aktionen direkt in den Netzwerken unmittelbar nach der Schrittkette im gleichen Baustein. Diese Vorgehensweise hat den Vorteil, dass die gesamte Logik an einem Ort konzentriert ist, was die Nachvollziehbarkeit der Programmausführung erleichtert. Besonders bei kleineren und weniger komplexen Projekten kann dies eine effizientere und schnellere Lösung darstellen.

Hierbei werden die Aktionen direkt in den Netzwerken programmiert, die auf die entsprechenden Schritte der Schrittkette folgen. Diese Methode kann die Programmlaufzeit optimieren, da keine zusätzlichen Bausteinaufrufe erforderlich sind, und die Struktur des Programms bleibt kompakt.

Kontinuierlich wirkende Aktion in FUP:

Vorgabe:

Die Aktion wird so lange ausgeführt, wie der Schritt aktiv ist.

Umsetzung:

Der entsprechende Schrittmerker wird direkt über eine Zuweisung auf den Ausgang geschrieben.

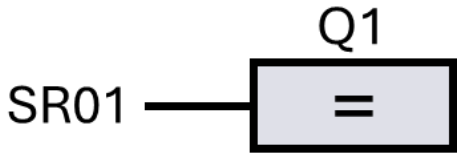


Bild 6 Kontinuierliche Zuweisung

Kontinuierlichen Zuweisung mit zusätzlicher Zuweisungsbedingung:

Vorgabe:

Die Aktion wird so lange ausgeführt, wie der Schritt aktiv ist und die Zuweisungsbedingung erfüllt ist.

Umsetzung:

Der entsprechende Schrittmerker wird mit der Zuweisungsbedingung logisch verknüpft, das Verknüpfungsergebnis wird dem Ausgang zugewiesen.

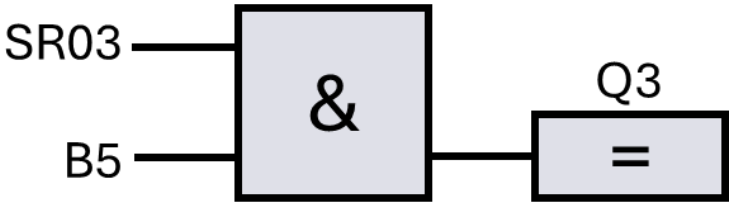


Bild 7 Kontinuierliche Zuweisung mit Bedingung

Speichernd wirkende Aktion:

Vorgabe:

Die gespeichert wirkenden Aktion bleibt über mehrere Schritte hinweg aktiv.

Umsetzung:

Die entsprechenden Schrittmerker zum Setzen und Rücksetzen der Aktion werden an einem Flipflop verschaltet. Das Flipflop wirkt speichernd auf den Ausgang.

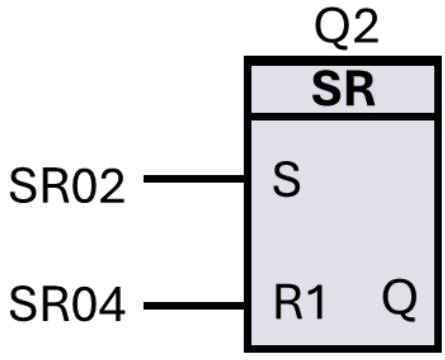


Bild 8 Speichernd wirkende Aktion

Mit dem Initialisieren der Kette würde die Aktion möglicherweise weiterhin gesetzt bleiben, da Schritt 4 (SR04) nicht aktiviert wurde. Um dies zu verhindern, kann am Rücksetzeingang mittels ODER-Verknüpfung zusätzlich auch die Variable für die Initialisierungsanforderung (Init) verschalten werden.

9.3 GRAFCET Ablaufkette in ST / SCL umsetzen

In diesem Abschnitt werden wir die in GRAFCET geplante Ablaufkette in der Programmiersprache ST / SCL umsetzen.

Es empfiehlt sich eine CASE-Struktur zur Abbildung der einzelnen Schritte des GRAFCETs zu verwenden. Hierfür muss zunächst eine Indexvariable mit ganzzahligem Datentyp (z.B. INT) definiert werden. Diese ist die Zählvariable, welche den aktuellen Schritt repräsentiert.

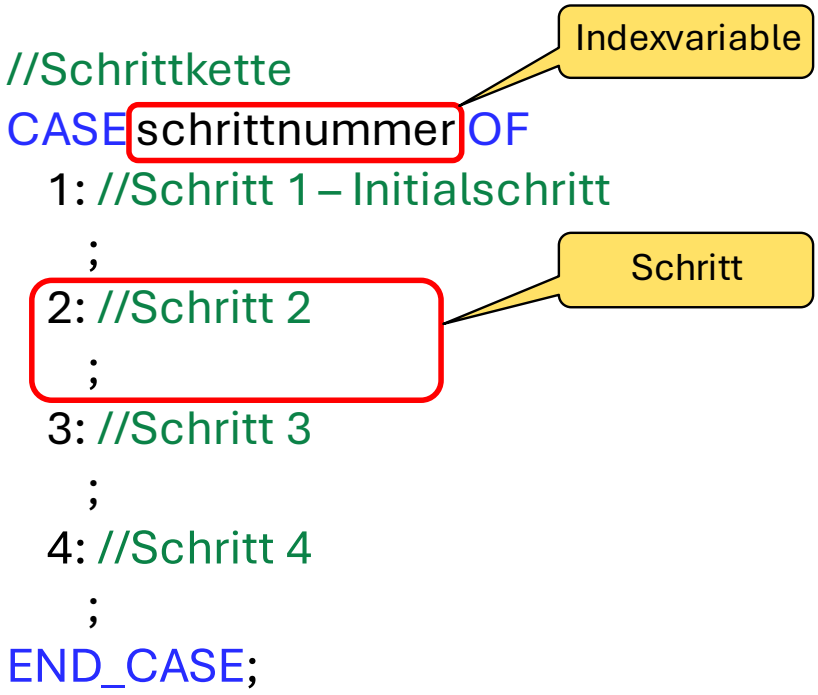


Bild 9 CASE-Struktur

9.3.1 Initialisieren der Ablaufkette

Bei Schritt 1 handelt es sich um den Initalschritt, dieser muss aktiviert werden, wenn die Kette initialisiert (Init) wird. Hierfür ist vor die CASE-Struktur eine IF-Anweisung zu setzen, welche bei Initialisierungsanforderung die Indexvariable (Schrittnummer) auf 1 zurücksetzt.

```
//Kette Initialisieren
IF Init THEN
  schrittnummer := 1;
END_IF;
```

Bild 10 Initialisierung

9.3.2 Struktur eines Schrittes (CASE)

Jeder Schritt (CASE) wird nach nachfolgendem Schema aufgebaut:
Zunächst werden die Aktionen implementiert, welche im Schritt durchgeführt werden sollen. Anschließend folgt eine IF-Anweisung, welche die Transition darstellt und bei Erfüllung der Transitionsbedingungen die Indexvariable, der Schrittkette, auf den nächsten Schritt setzt.

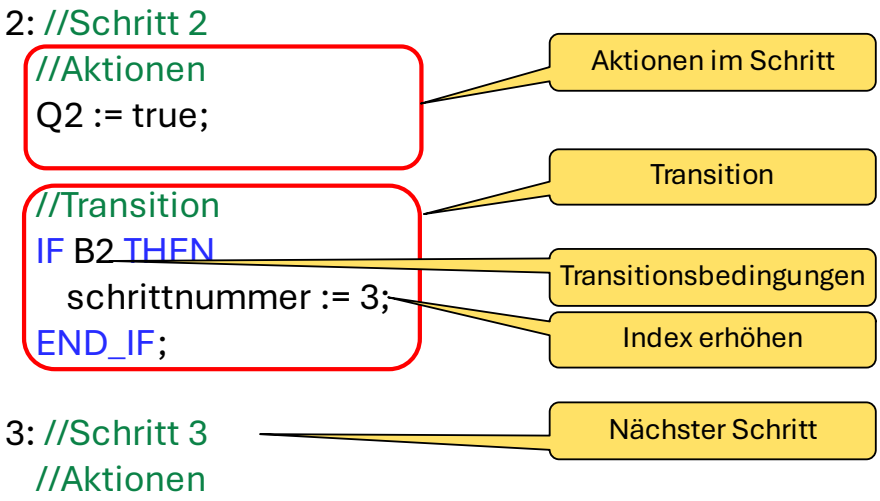


Bild 11 Struktur Schritt

Abhängig der geplanten Aktion, sind diese verschieden umzusetzen:

Kontinuierliche Zuweisung:

Vorgabe

Die Aktion wird so lange ausgeführt, wie der Schritt aktiv ist.

Umsetzung

Durch das Zuweisen des Signalzustandes "TRUE" auf die Variable, wird ein speicherndes Verhalten erreicht. Dies ist an dieser Stelle allerdings nicht gewünscht. Deshalb muss beim Verlassen des Schrittes, was dem Erfüllen der Transitionsbedingung folgt, die Aktion zurückgesetzt werden.

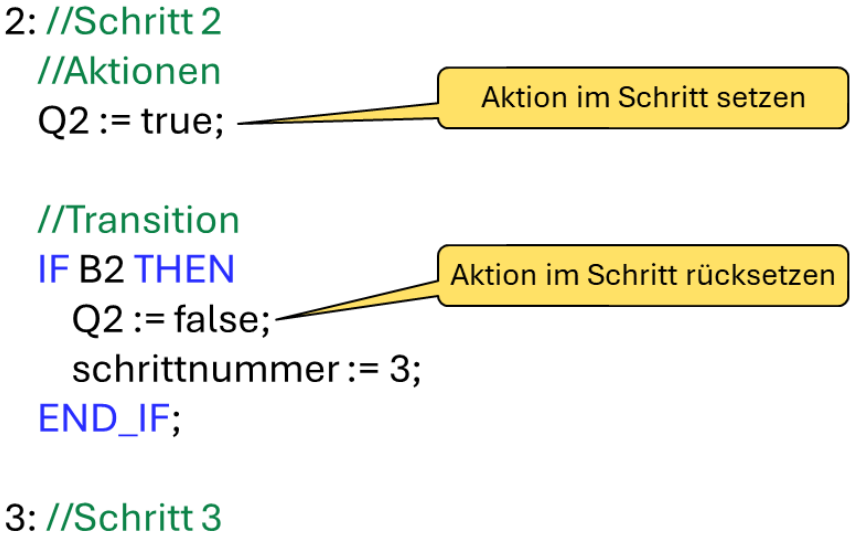


Bild 12 Kontinuierliche Zuweisung mit Bedingung

Kontinuierliche Zuweisung mit zusätzlicher Zuweisungsbedingung:

Vorgabe

Die Aktion wird solange ausgeführt, wie der Schritt aktiv ist und die Zuweisungsbedingung erfüllt ist.

Umsetzung

Der Aktion kann direkt die zusätzliche Zuweisungsbedingung verschaltet werden.

Analog zur Kontinuierlichen Zuweisung, muss auch hier beim Verlassen des Schrittes, die entsprechende Aktion zurückgesetzt werden.

```
3: //Schritt 3
  //Aktionen
  Q3 := B5;

  //Transition
  IF B3 THEN
    Q3 := false;
    schrittnummer := 4;
  END_IF;

4: //Schritt 4
```

Aktion im Schritt setzen, abhängig dem Signal der Zuweisungsbedingung

Aktion im Schritt rücksetzen

Bild 13 Kontinuierliche Zuweisung mit Bedingung

Speichernd wirkende Aktion:

Vorgabe

Bei gespeichert wirkenden Aktionen wird die Aktion über mehrere Schritte hinweg ausgeführt.

Umsetzung

Grundsätzlich sind alle Aktionen durch die direkte Zuweisung der Signalzustände speichernd wirkende Aktionen. Ist dieses Verhalten gewünscht, darf die Aktion in der IF-Anweisung der Transition nicht zurückgesetzt werden, sondern im entsprechenden Schritt.

```
2: //Schritt 2
    //Aktionen
    Q2 := true;

    //Transition
    IF B2 THEN
      schrittnummer := 3;    //In den nächsten Schritt schalten
    END_IF;

4: //Schritt 4
  //Aktionen
  Q2 := false;               //Aktion zurücksetzen

  //Transition
  IF B4 THEN
```

Aktion im Schritt setzen

Aktion im Schritt rücksetzen

Bild 14 Speichernd wirkende Aktion

Somit ergibt sich für den eingangs gezeigten GRAFCET folgende Schrittkette:

```
//Kette Initialisieren
IF Init THEN
  schrittnummer := 1;
END_IF;

//Schrittkette
CASE schrittnummer OF
  1: //Schritt 1 – Initialschritt
    //Aktionen
    Q1 := true;                 //Aktion setzen

    //Transition
    IF B1 THEN
      Q1 := false;              //Aktionen zurücksetzen
      schrittnummer := 2;       //In den nächsten Schritt schalten
    END_IF;

  2: //Schritt 2
    //Aktionen
    Q2 := true;

    //Transition
    IF B2 THEN
      schrittnummer := 3;       //In den nächsten Schritt schalten
    END_IF;

  3: //Schritt 3
    //Aktionen
    Q3 := B5;                   //Aktion setzen, solange B5 aktiv

    //Transition
    IF B3 THEN
      Q3 := false;              //Aktionen zurücksetzen
      schrittnummer := 4;       //In den nächsten Schritt schalten
    END_IF;

  4: //Schritt 4
    //Aktionen
    Q2 := false;                //Aktion zurücksetzen

    //Transition
    IF B4 THEN
      schrittnummer := 1;       //Sprung zum ersten Schritt
    END_IF;
END_CASE;
```

Bild 15 Programmierung Schrittkette