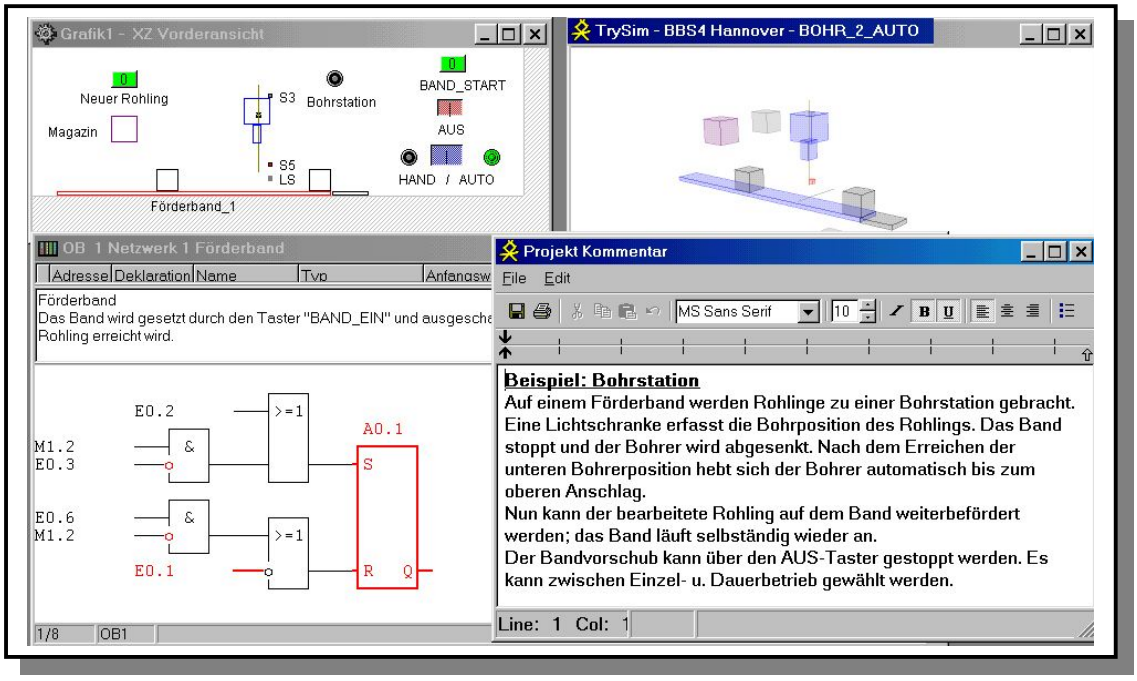
Einzellizenz CD-Nr. 2753

U. OHM <www.U-OHM.de>

SPS - STEP®7

Lernen und Testen mit TrySim®

Teil 1



Alle Programme lauffähig auf beiliegender CD

Eine systematische Einführung in die Automatisierungstechnik in Theorie und Praxis.
45 praxisnahe Projekte werden detailliert erläutert und in vorbereiteten oder in frei gestalteten Anlagen simuliert.

Inhaltsverzeichnis **Ver 1.1**

Was kann eine SPS?

C) Weitere Operationen

1. Programmierung mit TrySim® / Die Arbeit mit diesem Buch

Die in diesem Buch und im TrySim-Programm verwendeten Programmiersprachen AWL, FUP und KOP sind in der Darstellung und in den Funktionen sehr stark an Step®7 von Siemens angelehnt. Wer mit TrySim die SPS-Programmierung gelernt hat, kann sie "1 zu 1" bei Siemens-Anlagen anwenden.
Ohne weiteren Aufwand sind Sie in der Lage, ganz unterschiedliche Aufgabenstellungen hoher Komplexität zu lösen und was das Entscheidende ist: Ihre Anlagen laufen in dem Maße automatisch, wie Sie es geplant haben. Ein projektierter Endtaster wird direkt aus der Anlage angesteuert und sorgt z.B. von allein für den Drehrichtungswechsel eines Motors und muss nicht von Hand - und vielleicht falsch - simuliert werden.
Damit Sie sich auf das Erlernen der SPS-Programmierung konzentrieren können, sind auf beiliegender CD 45 Projekte als Anlagen mit allen benötigten Sensoren und Aktoren vorbereitet. Sie bekommen jeden Schritt in aufbauender Weise erklärt, so dass Sie parallel zum Lesen des Buches Ihr Projekt zum Leben erwecken können. Wenn es dabei einmal gar nicht klappen sollte, können Sie auch eine komplett programmierte, lauffähige Version Ihrer Aufgabe laden und ausprobieren.

Ferner kann das getestete TrySim-Programm in der TrySim-Professional-Version von der Siemens-software Step®7 geladen und an die SPS-Steuerung S7 übertragen werden. Dieses Buch erläutert Ihnen auch dafür Schritt für Schritt die Umsetzung.

Natürlich können Sie auch eigene Anlagen erstellen und programmieren. Allerdings würde die systematische Anleitung für den "Bau" eigener Anlagen den Rahmen dieses Buches sprengen.

Die folgenden Projekte ermöglichen es, die firmenunabhängige Systematik und Logik einer Aufgabenstellung zu begreifen. Eine UND-Verknüpfung bleibt bei allen Fabrikaten und software-Produkten ein UND. Somit sind Ihre erworbenen Kenntnisse systemübergreifend.

Deshalb werden allgemeingültige Zusammenhänge der Projektprogrammierung voran gestellt. Etwas Theorie ist hier sicherlich hilfreich, soll aber auch nicht überbewertet werden oder gar abschreckend wirken. Die folgenden Seiten sollen in aller Kürze einen Abriss der Begriffe und Normen geben. Es ist nicht erforderlich, dieses Kapitel systematisch durchzuarbeiten, um mit der praktischen Arbeit beginnen zu können, da die erforderliche Theorie bei der jeweiligen Projektbeschreibung erläutert wird. Dieser Abschnitt basiert auf den Erläuterungen der Lernsoftware-CD "SPS-Wissen für Einsteiger", deren Umsetzung als Druckerzeugnis die Firma Siemens AG freundlicherweise gestattet hat. Es wurden zusätzliche Erläuterungen eingearbeitet.

Entscheidender ist das Verständnis über das Zusammenspiel außenliegender Signalgeber / Sensoren und deren Signalverarbeitung. Im Klartext geht es um die Frage: "Was wird von der SPS abgefragt, eine logische "1" oder eine "0""? Hier liegt die häufigste Fehlerquelle! Ebenso häufig ist die Unsicherheit, in welcher Reihenfolge logische Verknüpfungen stattfinden. Zu diesen Themen folgen im Praxisteil - hoffentlich ausreichende - Übungen.

Es wurde im Rahmen dieses Buches bewusst auf die Präsentation aller Step®7-Feinheiten verzichtet. Trotz der Stoffreduzierung werden fundierte Kenntnisse vermittelt, um anspruchsvolle Projekte zu erstellen.

Dieses Buch sollte nicht wie ein Krimi auf dem Sofa gelesen werden, sondern in Reichweite des PCs. Es ist als direkte Anleitung zur Umsetzung von Steuerungsaufgaben gedacht. Sie lösen von Anfang an anspruchsvolle Aufgaben und lernen gleichzeitig die angepassten erforderlichen Schritte kennen, die zum hoffentlich "lustvollen" Bedienen der Software gehören.

Es empfiehlt sich, die Projekte der Reihe nach zu bearbeiten, da sie in den Erläuterungen entsprechend aufeinander aufbauen.
Der Autor erhebt nicht den Anspruch, für jedes Projekt die optimale oder technisch ausreichende Lösung vorzuschlagen. Die Projekte können gern von Ihnen verbessert und den restlichen Lesern zugänglich gemacht werden.

Es ist eine intensiver Meinungs- und Erfahrungsaustausch mit den Lesern beabsichtigt. Jeder Käufer kann bei Nennung seiner Rechnungsnummer unter U.OHM@freenet.de 5 kostenlose Unterstützungen per e-mail für die Projektgestaltung dieses Buches erhalten.

Ferner erhalten Sie auf der Website www.U-OHM.de Antworten auf FAQs zum Buch. Sie können auch eigene Projekte zur Veröffentlichung mailen, bzw. von dort downloaden.

Für Hinweise jeder Art bin ich dankbar, damit das Produkt weiter reifen kann. Notfalls stecke ich auch Kritik ein; über ein bisschen Lob würde ich mich aber auch freuen.

Die Projekte sind für eine Bildschirmauflösung von 1024x768 konzipiert. Bei einer Auflösung 800x600 Ist das Zusammenspiel der einzelnen Teilfenster nicht optimal zu verfolgen. Alle Teilfenster können jedoch in Lage und Größe verändert werden.
Für die Darstellung bzw. Erkennbarkeit der Text- und Digitalanzeigen ist die Windows-Einstellung von kleinen Schriftarten erforderlich.
(<Start>|<Einstellungen>|<Systemsteuerung>|<Anzeige>|<Einstellungen>, dort: <weitere Optionen...>)

Haftungsausschluss:
Weder der Autor des Buches noch die Cephalos GmbH als Produzentin der TrySim-Software können für irgendwelche Folgen der Nutzung beiliegender CD oder der Übertragung von Projekten oder Teilen davon in SPS-Steuerungen verantwortlich gemacht werden. Die Projektierung liegt ausschließlich in der Verantwortung des Anwenders. Die in diesem Buch und auf der CD erstellten Projekte erheben nicht den Anspruch, im Rahmen von erforderlichen Sicherheitsanforderungen ausreichend zu sein, bzw. voll praxistauglich zu sein. Die Projekte sind bewusst reduziert angelegt, um für Einsteiger eine Überschaubarkeit zu ermöglichen.

Typografische Konventionen:

Beispiel	Beschreibung
`"Eingabe"`	Vom Benutzer einzugebende Zeichen (ohne "" eingeben)
<AUS>	Im Projekt vorgegebene Bezeichnung (z.B. für einen Taster)
<LM>	Abkürzung für: "Benutzen Sie bitte die linke Maustaste"
<rM>	Abkürzung für: "Benutzen Sie bitte die rechte Maustaste"
<SPS> | <Öffnen>	Eine Reihenfolge von Befehlen, die nacheinander mit der Maustaste auszuführen sind.
"MOT_1"	Darstellung von symbolischen Adressen. Die " müssen im Editierfenster nur eingegeben werden, wenn der Name Sonderzeichen oder Leerzeichen enthält.
TrySim-Hilfe |<Index> „Und“	Wählen Sie in der TrySim-Menüzeile <Hilfe> aus, öffnen dort <Index> und schreiben: und.
„0“	eine logische 0 (Null), d.h. es liegt keine Spannung am Ein- bzw. Ausgang an (gegen Masse gemessen).
„1“	eine logische 1,), d.h. es liegt Spannung am Ein- bzw. Ausgang an (gegen Masse gemessen).

2. Begriffe und Erläuterungen zur SPS

Der Text dieses Kapitels ist z. T. der CD "SPS-Wissen für Einsteiger" entnommen, wofür die Siemens AG freundlicherweise ihre Genehmigung gab. Weitere Informationen zu dieser CD und anderen Lernhilfen der Siemens AG, Bereich Automatisierungs- und Antriebstechnik (A&D) erhalten Sie unter www.siemens.com/sce.

2.1. Vom Schütz zur SPS

Es gibt verschiedene Arten von automatischen Steuerungen.
In **v**erbindungs**p**rogrammierten **S**teuerungen (VPS) wurde die Funktion durch Verdrahtung und Kombination der Schaltelemente festgelegt.
Als Grundlage dienten Stromlaufpläne, Belegungspläne und Verdrahtungslisten. Die Anlage konnte aber erst installiert werden, nachdem bekannt war, welche Aufgabe zu lösen war und welche Schaltelemente, wie z.B. Schalter, Schütze etc. zur Verfügung standen.
Wurden Fehler gemacht, musste die Verdrahtung gelöst werden; die Verbindung musste neu erstellt werden. Jede Funktionsänderung bzw. -erweiterung war immer mit Bauteiländerungen, Umverdrahtung und Montagearbeiten verbunden.
Speicher**p**rogrammierbare **S**teuerungen (SPS) können ohne Umverdrahtung je nach Einsatzzweck neu programmiert werden. Es ist sehr einfach möglich, Funktionsfehler zu beseitigen oder die Anlage zu erweitern.
Zur Steuerung muss ein Programm geschrieben werden, das die Funktion der Anlage festlegt. Es ist aber deutlich einfacher, ein Programm zu schreiben bzw. zu ändern, als eine Schaltung zu verdrahten bzw. umzuarbeiten.
Die Funktion der SPS wird durch das Programm im Programmspeicher des Automatisierungssystems (AS) festgelegt. Erstellt wird das Programm in einem Programmiergerät (z.B. PC), von dem es mittels Verbindungsleitung zum AS übertragen wird.

Automatische Steuerungen sind in vielen Bereichen des Lebens anzutreffen. Sie können grundsätzlich mechanisch, hydraulisch, elektronisch / elektrisch, pneumatisch oder speicherprogrammierbar ausgeführt werden.

2.2. Hardware

Das Automatisierungssystem (AS) bearbeitet das Programm. Neben **kompakten** Automatisierungssystemen, die auf kleinem Raum die wichtigsten Komponenten beinhalten, gibt es auch **modular aufgebaute** Automatisierungsgeräte, bei denen die Module individuell nach Anforderungen zusammengestellt werden können.
Zu einer modular aufgebauten SPS gehören mindestens:

- **Stromversorgung:** Die Netzspannung 230 V AC / 50 Hz wird in die Versorgungsspannung 24 V DC für die Eingangs- u. Ausgangssignale umgewandelt.
  Intern arbeitet die SPS mit einer Spannung von 5 V.
- **Zentralbaugruppe (CPU):** In diesem "Herzstück" der SPS wird das Programm gespeichert und verarbeitet. Der Betriebsartenschalter dient dazu, die SPS von Hand in die gewünschten Betriebszustände zu bringen. Eine ständige Verbindung zwischen Automatisierungsgerät und Programmiergerät ist nicht erforderlich. Die 2 wichtigsten Betriebsarten sind "**STOP**" (keine Bearbeitung des Programms) und "**RUN**" (das Anwenderprogramm wird im Programmspeicher des AS bearbeitet).
  Eine Pufferbatterie dient dazu, die Daten und das Programm bei Abschaltung der Steuerung oder bei Netzspannungsausfall zu erhalten.

Weitere Einheiten:

- **Anschaltungsbaugruppen:** Hiermit werden mehrere Baugruppenträger miteinander verbunden.

- **Signalmodule** bilden die Schnittstelle zwischen der CPU und der Anlage. Grundsätzlich wird zwischen **digitalen** und **analogen** Signalmodulen unterschieden.

- **Digitalbaugruppen** sind Baugruppen, die die Signale der angeschlossenen Signalgeber und Stellglieder auf die SPS-Pegel umsetzen. Mit ihrer >Hilfe ist es möglich, "**binäre**" Signale zu empfangen und auszugeben.

- **Analogbaugruppen** sind in der Lage, analoge Signale in digitale Signale umzuwandeln, so dass diese von der Zentraleinheit verarbeitet werden können. Deshalb werden diese Baugruppen besonders für **Regelungsaufgaben, wie z.B. Temperatur-, Niveau- oder Drehzahlregelung** eingesetzt.

- **Funktionsbaugruppen** erweitern die Vielseitigkeit des AS. Es gibt Module z.B. für schnelles Zählen, Regeln oder Positionieren.

Zu einer SPS gehören neben dem Automatisierungssystem auch **Signalgeber** (z.B. Schalter, Taster, Sensoren) und **Stellgeräte** (z.B. Schütze, Ventile). Diese sind an die Signalmodule der AS angeschlossen.

Jede CPU ist über ein **Bussystem** mit allen Komponenten bzw. Modulen verbunden. Der gesamte Datenaustausch und die Weitergabe von Steuerbefehlen findet über dieses Bussystem statt.

Die eigentliche Funktion der Anlage wird durch das **Programm** bestimmt. Das Programm besteht aus einer Vielzahl von Anweisungen, die vom Automatisierungssystem **nacheinander abgearbeitet** werden. Dabei verknüpft das Automatisierungssystem die Signalzustände der Geber und schaltet die Stellgeräte nach Anweisung des Programms ein bzw. aus.

2.3. Software

Das **Betriebssystem** ist ein Programm, das den Betrieb des AS überhaupt erst ermöglicht. Es ist in jeder CPU einer SPS vorhanden. Es kontrolliert auch den Datenaustausch mit dem Programmiergerät.

Anwenderprogramme sind Programme, die der SPS-Programmierer für die Realisierung der Steuerung erstellt oder benutzt.

Damit die SPS ein Anwenderprogramm versteht, muss es in die **Maschinensprache** übersetzt werden. Die Übersetzung geschieht durch das Programmiergerät.

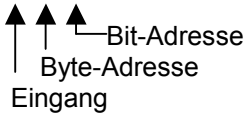
Das übersetzte Anwenderprogramm wird vom Programmiergerät über die Verbindungsleitung in den **Programmspeicher des Automatisierungsgerätes** übertragen.
Nach dem Starten des SPS-Programms werden die einzelnen Steueranweisungen vom Betriebssystem **zyklisch** bearbeitet.

2.4. Adressierung

Damit man versteht, wie ein AS das Steuerprogramm bearbeitet, ist es notwendig, die Begriffe **Bit, Byte, Wort** und **Doppelwort** zu kennen.

- Ein **Bit** ist die Einheit für eine Binärstelle oder ein Binärzeichen. Es ist die kleinste informationstechnische Einheit und kann nur den Signalzustand "**0**" oder "**1**" annehmen. Mehrere Bits können zu größeren Einheiten zusammengefasst werden.
  Damit die Signalgeber und Stellgeräte angesprochen werden können, müssen sie eine Adresse im AS besitzen.
  Jedes Bit erhält im AS eine Nummer, die so genannte **Bit-Adresse**. 0 - 7 stehen dafür zur Verfügung.

- Ein **Byte** stellt eine Einheit von **8 Bit** dar. Es wird z.B. für die Zusammenfassung von Signalzuständen von 8 Ein- oder Ausgängen benutzt, um damit z.B. Zahlen mit 2 Ziffern darzustellen.
  Die einzelnen Bytes erhalten ebenfalls eine Nummer, die **Byte-Adresse**. Die Nummer hängt bei einigen CPUs vom Steckplatz der Baugruppe ab, bei anderen kann sie frei gewählt werden.
  Die Byte-Adressen werden zusätzlich durch Bezeichnungen wie Eingang (E) und Ausgang (A) noch näher festgelegt. Spricht man von einem Eingangs-Byte (EB), so meint man alle 8 Signale (0 - 7), die an einem Byte angeschlossen sind.
  Die Bit-Adresse wird durch einen Punkt von der Byte-Adresse getrennt (z. B. E 3.2).



- Ein **Wort** besteht aus **2 Byte** bzw. 16 **Bit**. Mit einem Wort können z.B. 16 Eingänge oder Ausgänge zusammengefasst werden, um damit z.B. Zahlen mit 4 Ziffern darzustellen.
  Zwei Eingangs-Bytes sind z.B. ein Eingangswort (EW). Das Eingangswort 0 (EW0) enthält die Eingangs-Bytes 0 (EB0) und 1 (EB1). Bei einem Wort entspricht die Wortadresse immer der des Bytes mit der kleineren Nummer.

- Ein **Doppelwort** (**ED** bzw. **AD**) besteht aus **2 Wörtern** oder **4 Byte** oder **32 Bit**. Ein Doppelwort ist die größte Einheit, die von einem Automatisierungssystem in einem Schritt verarbeitet werden kann.
  Das ED0 besteht aus den Eingangs-Bytes EB0, EB1, EB2 und EB3.
  Wenn z.B. das ED4 verwendet wird, ist erst das Byte 8 frei für zusätzliche Abfragen.

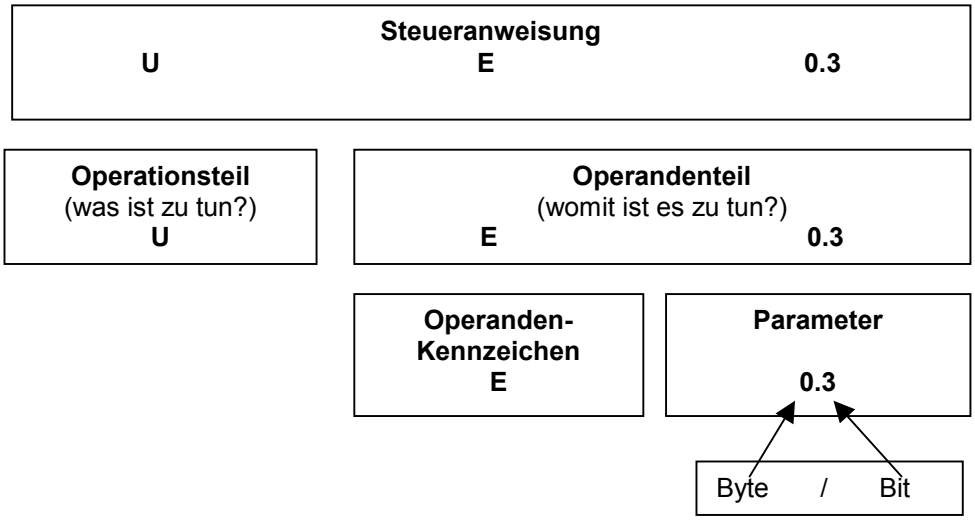
Bei der **absoluten Adressierung** wird direkt die Adresse (z.B. des Eingangs E 1.0) angegeben. In diesem Fall ist keine Symboltabelle erforderlich, das Programm ist aber schlechter lesbar.
Die **symbolische Adressierung** ermöglicht es, anstelle von direkten Adressen mit Symbolen (z.B. "MOTOR_EIN") zu arbeiten. Diese Symbole werden in der Symboltabelle den tatsächlichen Adressen zugeordnet. Man nannte sie daher auch "Zuordnungsliste".

Bei einer **festen Adressierung** sind die Adressen durch den Steckplatz vorgegeben.
Bei der **variablen Adressierung** können die Adressen unabhängig vom Steckplatz vergeben werden.
Welche Adressierungsart angewandt wird, hängt von der CPU ab.

"Globale Symbolik" bedeutet, die Adressen von Signalen werden durch Symbole bezeichnet, die für alle Bausteine des Programms gültig sind.

2.5. Steueranweisung

Die Steueranweisung ist die kleinste Einheit eines Steuerprogramms. Sie besteht in der Anweisungsliste und auch im Programmspeicher aus dem **Operationsteil** und dem **Operandenteil**.
Beispiel: U E 0.3
Der Operationsteil (U für die logische Verknüpfung UND) bestimmt, **was** zu tun ist.
Der Operandenteil (E 0.3) gibt an, **an welcher Adresse** es zu tun ist.
Die "Klemmleiste" E oder A wird als **Operandenkennzeichen** bezeichnet.
Der **Parameter** ist die Adresse des Operanden (0.3)



2.6. Signale

Grundsätzlich unterscheidet man zwischen 2 Signaltypen:

Digitale Signale nehmen nur ganz bestimmte Werte in einem vorgegebenen Bereich an (Stufenschalter).
Von einem **binären Signal** wird gesprochen, wenn ein Eingangs- oder Ausgangszustand nur 2 unterschiedliche Werte annehmen kann. Diese Werte werden mit den Ziffern "0" und "1" bezeichnet.

Analoge Signale können in einem durchgehenden Bereich jeden Wert annehmen, z.B. die Spannung an einem Potenziometer.

Signalzustand

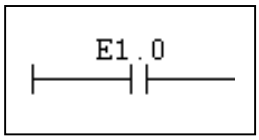
Signalgeber	Zustand	Signalzustand
Schließer	betätigt	1
	nicht betätigt	**0**
Öffner	**betätigt**	**0**
	nicht betätigt	1

Prozess			Programmtechnische **Auswertung** in der SPS		
Der Geber ist ein...	Der Geber ist ...	Spannung am Eingang ist...	Signalzustand am Eingang	Abfrage auf Signalzustand ...	
				"1" / Ergebnis	**"0"** / Ergebnis
Schließer 	betätigt 	vorhanden	1	"JA" 1	"NEIN" 0
	nicht betätigt 	nicht vorhanden	0	"NEIN" 0	"JA" 1
Öffner 	betätigt 	nicht vorhanden	0	"NEIN" 0	"JA" 1
	nicht betätigt 	vorhanden	1	"JA" 1	"NEIN" 0

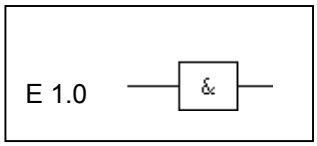
Die Symbole der verschiedenen Programmiersprachen werden wie folgt dargestellt:

"1"-Abfrage
Abfragesymbol / Anweisung in:

KOP:



FUP:



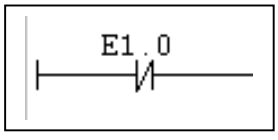
AWL:

```
U E 1.0
.....
```

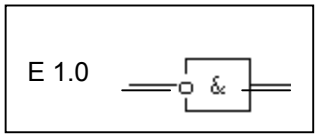
In allen Sprachen /Symbolen lautet der Auftrag:
"Hallo CPU, kontrolliere den Spannungspegel an der Klemme (dem Operanden) E 1.0. Wenn dort Spannung gegen Null gemessen wird, also eine logische "1" anliegt, setze die Verknüpfung mit dem Rest des Netzwerkes fort".

"0"-Abfrage
Abfragesymbol / Anweisung in:

KOP:



FUP:



AWL:

```
UN E 1.0
.....
```

In allen Sprachen /Symbolen lautet der Auftrag:
"Hallo CPU, kontrolliere den Spannungspegel an der Klemme (dem Operanden) E 1.0. Wenn dort keine Spannung gegen Null gemessen wird, also eine logische "0" anliegt, setze die Verknüpfung mit dem Rest des Netzwerkes fort".

Bei einem Automatisierungsgerät werden entsprechend dem Programm die Operanden auf ihre Signalzustände abgefragt.
Ist **Spannung** vorhanden, so hat der Operand den Signalzustand "**1**".
Ist **keine Spannung** vorhanden, so hat der Operand den Signalzustand "**0**".

Bitte beachten Sie:
Das AS kann Eingänge (und Ausgänge) lediglich auf den Signalzustand "1" oder "0" abfragen.

Das AS verknüpft keine Schließer und Öffner, sondern Signalzustände

2.7. Speicherfunktion

In einem AS wird eine Speicherfunktion mit Speichergliedern realisiert. Es handelt sich um einen Speicher, der einen **Setzeingang S** und einen **Rücksetzeingang R** besitzt.
Ein Signal "1" am Setzeingang setzt die Speicherfunktion. Der Ausgang Q des Speichergliedes führt dann den Signalzustand "1". Dieser Signalzustand bleibt auch erhalten, wenn am Setzeingang das Signal wieder von "1" auf "0" wechselt.
Ein Signal "**1**" **am Rücksetzeingang setzt die Speicherfunktion auf "0" zurück.**
Ein Speicherglied (**SR**) führt am Ausgang "0", wenn an beiden Eingängen der Signalzustand "1" anliegt; man spricht von der Eigenschaft "**dominierend Rücksetzen**".
Ein Speicherglied (**RS**) führt am Ausgang "1", wenn an beiden Eingängen der Signalzustand "1" anliegt; man spricht von der Eigenschaft "**dominierend Setzen**".
Das liegt daran, dass das Programm zeilenweise gelesen wird und am Zyklusende der zuletzt gültige Signalzustand im Speicher an die Ausgabegruppe übertragen wird.

2.8. Merker

Teil-Ergebnisse können zwischengespeichert werden.
Komplizierte Verknüpfungen können übersichtlicher gestaltet werden, wenn man nur Teilbereiche zusammenfasst. Diese **Verknüpfungsergebnisse** (**VKE**) können in internen Merkern abgelegt (zwischengespeichert) werden. Diese Merkerergebnisse können dann an anderer Stelle (in einem anderen Netzwerk) wieder weiter verknüpft werden.
Um die einzelnen Merker von einander zu unterscheiden, haben sie absolute Adressen, die genau wie die Ein- und Ausgangsadressen aufgebaut sind. Das **Operandenkennzeichen** ist "**M**".

2.9. Zahlensysteme

Zur Beschreibung der Signalzustände wird das Dualzahlsystem verwendet, weil...

- Der Prozessor des AS nur die Signalzustände „0“ und „1“ kennt.
- Das Dualzahlsystem die Zustände EIN /AUS bzw. Spannung / keine Spannung kennt.

Der Anteil, den eine **Ziffer** zum Zahlenwert beisteuert, hängt von ihrer Position innerhalb der Zahl ab. Jede Stelle hat eine bestimmte Wertigkeit (**Stellenwert**), welche sich aus den einzelnen Potenzen zur gewählten **Basis** ergibt. Man erhält den Zahlenwert, wenn man jede Ziffer mit ihrem Stellenwert multipliziert und die erhaltenen Werte addiert.

Stellenwerte der Zahlensysteme

Potenz	2	1	0
Dualzahl, Basis: 2	4	2	1
Dezimalzahl, Basis: 10	100	10	1
Hexadezimalzahl, Basis: 16	256	16	1

Beispiele:

$101_2 = 5$

$101_{10} = 101$

$101_{16} = 257$

Mit 8 Stellen können Dualzahlen bis zum Wert 255, mit 16 Stellen bis zum Wert 65535 dargestellt werden.

Für das Sedezimal- oder Hexadezimalsystem werden die zehn Ziffern des Dezimalsystems verwendet (0- 9). Für die restlichen 6 Ziffern nimmt man die ersten Buchstaben des Alphabets.
A = 10, **B** = 11, **C** = 12, **D** = 13, **E** = 14, **F** = 15.

Diese Zeichen bedeuten Ziffern! $1CD_{16} = 1 \times 256 + 12 \times 16 + 13 \times 1$

DUALZAHL	**1 1 0 0**
Wertigkeit der Stellen	8 4 2 1
Dezimalwert	8+4 = 12
Hexadezimalwert	**C**

Jeweils 4 Stellen einer Dualzahl lassen sich durch ein Zeichen als Hexadezimalzahl darstellen.
Aus diesem Grunde werden häufig Dualzahlen rechts beginnend in Vierergruppen angezeigt.

$$1000\ 1111_2 = 8F_{16}$$

Bei den **BCD-Zahlen** handelt es sich nicht um ein weiteres Zahlensystem mit den bereits beschriebenen Merkmalen, sondern um Dezimalzahlen, bei denen jede einzelne Ziffer in eine Dualzahl umgewandelt wird, so dass sie das AS verarbeiten kann.
Die einzelnen Ziffern einer Dezimalzahl werden mit je 4 Binärstellen (Bits) verschlüsselt.
4 Bits je Stelle sind erforderlich, da die größte Dezimalziffer 9 in der binären Darstellung 1001 bedeutet.

$$\mathbf{4327_{10} = 0100\ \ 0011\ \ 0010\ \ 0111_{BCD}}$$

Die (möglichen) Dualzahlen 1010 bis 1111 ergeben für die jeweils einstellige Übersetzung keinen Sinn. Es gibt also nur 10 verwertbare Zahlen.
Da die BCD-Zahlen kein eigenes Zahlensystem bilden, ist es nur bedingt möglich, mit ihnen mathematische Operationen durchzuführen.

Also Vorsicht bei Siemens-Zähler- und Zeitbausteinen, die auch einen BCD-Ausgang haben!!!

Man benutzt sie deshalb in der Regel nur für Ein- und Ausgaben, die vorwiegend der Kommunikation mit dem Menschen dienen.

2.10. Programmbearbeitung

Der Prozessor des AS bearbeitet das in den Programmspeicher geschriebene Steuerprogramm in einer ständig auflaufenden Wiederholungsschleife.
Dieser Vorgang wird **zyklische Programmbearbeitung** genannt.
Um den Signalzustand des jeweiligen Eingangs nicht in jeder Abfrage direkt an der entsprechenden Eingabebaugruppe abfragen zu müssen, lädt die CPU vor jedem Programmdurchlauf die Signalzustände aller Eingänge in ihren Speicher. Dieser Speicherbereich nennt sich **Prozessabbild der Eingänge (PAE)**.

Durch das PAE bleibt der Signalzustand der Eingänge über den ganzen Programmzyklus konstant.

Auch für die Ausgänge wird während der Abarbeitung des Anwenderprogramms ein **Prozessabbild der Ausgänge (PAA)** erstellt. Nach dem Durchlauf eines jeden Prozesszyklusses werden die Informationen zu den Ausgabebaugruppen transferiert.

Die Zykluszeit ist die Zeit, die während eines Programmzyklusses vergeht. Sie setzt sich zusammen aus:
- Abfragen der Signale an den Eingangsbaugruppen und Aktualisieren des PAE.
- Bearbeiten des Anwenderprogramms.
- Übertragen der Werte aus dem PAA in die Ausgabebaugruppen.
- Betriebssystemlaufzeit.

Die **Reaktionszeit** eines Programms ist die Zeit vom Erkennen eines Signalwechsels bis zur Änderung eines damit verknüpften Ausgangssignals. Das AS erkennt einen veränderten Signalzustand erst mit dem nächsten Einlesen des PAE. Die entsprechende Reaktion erfolgt erst nach der darauffolgenden Programmbearbeitung mit der Übertragung des PAA auf die Ausgangsbaugruppen. Die Reaktionszeit einer SPS beträgt damit minimal einen Zyklus und maximal 2 Zyklen. Das AS bemerkt deshalb besonders schnelle Signalwechsel nicht. Ein Eingangssignal sollte deshalb mindestens eine Zykluszeit lang anstehen, damit es sicher erkannt wird.

2.11. Lineare Programmierung

Für kleine Projekte eignet sich die Methode der **linearen Programmierung im Baustein OB1.**
Dieser Baustein enthält dabei alle Anweisungen des Programms.
Die maximale Länge ist gerätespezifisch beschränkt, da jeder Baustein nur eine bestimmte Größe haben kann. Nachteilig ist, dass das Programm leicht unübersichtlich wird und damit die Fehlersuche erschwert. Das Betriebssystem eines AS beginnt die zyklische Programmbearbeitung immer mit dem **O**rganisations**b**austein "OB1". Deshalb muss sich ein lineares Programm immer im OB1 befinden.

2.12. Strukturierte Programmierung

Ein strukturiertes Programm enthält mehrere Bausteine. Wenn man ein Programm für einen Prozess oder eine Anlage erstellt, so werden für in sich abgeschlossene Teilfunktionen eigene Bausteine verwendet.
Die Anzahl der Ebenen, in die ein Programm geschachtelt ist, bezeichnet man als **Schachtelungstiefe**.

Die strukturierte Programmierung bietet folgende **Vorteile**:
- Einfache und übersichtliche Programmierung großer Programme
- Möglichkeit zur Standardisierung von Programmteilen
- Leichte Änderungsmöglichkeiten
- Einfache Inbetriebnahme und Fehlersuche
- Unterprogrammtechnik (Aufruf eines Baustein von verschiedenen Stellen aus).

2.13. Bausteinarten und Funktionen

Das Siemens-AS stellt verschiedene Arten von Bausteinen zur Verfügung, in denen das Anwenderprogramm und die dazugehörigen Daten gespeichert werden können.
Je nach Prozessanforderung kann man das Programm mit verschiedenen Bausteinen strukturieren.

- **Organisationsbausteine** (**OB**s) sind die Schnittstelle zwischen Betriebssystem und Programm. Das gesamte Programm kann im zyklisch bearbeiteten OB1 gespeichert werden (lineare Programmierung). Das Anwenderprogramm kann aber auch in mehreren Bausteinen gespeichert werden, wobei der OB1 verwendet wird, um diese Bausteine in der richtigen Reihenfolge aufzurufen (strukturierte Programmierung).
- Ein **Funktionsbaustein (FB)** ist ein Baustein mit einem zugeordneten Speicherbereich. Er kann ein "Gedächtnis" haben und eignet sich zur Programmierung von häufig wiederkehrenden komplexen Funktionen.
- Eine **Funktion (FC)** ist ein Baustein ohne Gedächtnis. Hier werden Programmteile bearbeitet, die ihre Ergebnisse lediglich an andere Programmteile weitergeben.
- **Datenbausteine (DB)** sind Datenbereiche im Anwenderprogramm, in denen Prozessdaten dauerhaft gespeichert und wieder aufgerufen werden können.

2.14. Wichtige Fragen bei der Projektierung

- Gibt es gefährliche Zustände für Menschen, Tiere, Material oder Maschine?
- Verhält sich die Aufgabe richtig bei Drahtbruch und nach Wiederkehr der ausgefallenen Spannung?
- Gibt es eine ausreichende NOT-AUS-Schutzeinrichtung?

Bei Störungen im AS müssen Befehle vom NOT-AUS-Schalter und von Sicherheitsgrenzschaltern auf alle Fälle wirksam bleiben.

Durch Fehler in den Geberkreisen, wie Leiterbruch oder Erdschluss, darf das Ausschalten nicht blockiert werden.

- **Einschalten mit Schließern**
- **Ausschalten mit Öffnern**

Ein Leiterbruch im Sicherheitskreis (Grenztaster, Not-Aus, Aus) muss **jederzeit** - also von allein! - zum Ausschalten führen. Deshalb sind hier immer "Ruhestromkreise" erforderlich, d.h., der Stromkreis ist im Betriebsfall geschlossen (an der SPS-Klemme liegt eine "1" an) und muss im Störungsfall unterbrochen werden. Deshalb müssen Öffner verwendet werden.

Folgende Schritte sind für eine systematische Projektierung wichtig:

- Sichtung der Unterlagen
- Festlegen der Programmstruktur
- Erstellen der Zuordnungsliste (Symboltabelle)
- Programmierung und Testen der Bausteine
- Zusammenfügen und des Testen des gesamten Programms
- Erstellen einer umfassenden Dokumentation

2.15. Akkumulatoren

Die Akkumulatoren (AKKUs) 1 und 2 sind die Universalregister für die Verarbeitung von Bytes, Wörtern und Doppelwörtern. Da sie auch Doppelwörter aufnehmen können, sind sie 32 Bit breit.
Es können aus dem Speicher Konstanten oder **Werte als Operanden** in den AKKU **geladen** werden, wo sie verarbeitet werden können. Ebenso kann ein **Wert aus dem AKKU 1** (z.B. ein Merkerwort MW) **transferiert,** d.h. auf einen anderen Operanden kopiert werden.

- Eine **Ladeoperation "L"** wirkt immer nur auf AKKU 1 und speichert den alten Wert vom AKKU 1 dann im AKKU 2.

3. Warum benötigt man unterschiedliche Datentypen?

Moderne Automatisierungsgeräte dienen nicht nur zur logischen Verknüpfung von "0"/"1"- Zuständen, sondern sie zeichnen sich gerade durch die Verarbeitung von unterschiedlichen Zahlenwerten aus. Erst durch die Bereitstellung vielfältiger mathematischer Operationen wird es möglich, aus dem reinen Steuergerät ein Automatisierungsgerät zu machen, das den Ansprüchen der Regelungstechnik genügt.
Je nachdem, wie viele Werte (Zahlen) zu unterscheiden sind, und ob es nur positive oder auch negative Zahlen gibt, unterscheidet man verschiedene Datentypen. Alle Datentypen bestehen aus einer unterschiedlich großen Zahl von Stellen, die alle den Wert "0" oder "1" annehmen können.
Je nachdem, welchen Datentyp man einer Zeichenkette zugeordnet hat, wird der entsprechende Speicherbereich unterschiedlich ausgewertet. Deshalb ist es erforderlich, den Datentyp festzulegen.

Die folgenden Datentypen sind u. a. zu wählen:

Datentypen	Beschreibung	Speicherbereich
BOOL	Boolesch (1 Bit)	0 bis 1
BYTE	Byte ohne Vorzeichen (8 Bit)	0 bis 255
WORD	Ganze Zahl (16 Bit) ohne Vorzeichen	0 bis 65.535
INT	Ganze Zahl (16 Bit) mit Vorzeichen	-32.768 bis +32.767
DWORD	Ganze Zahl (32 Bit) ohne Vorzeichen	0 bis 4.294.967.295
DINT	Ganze Zahl (32 Bit) mit Vorzeichen	-2.147.483.648 bis +2.147.483.647
REAL	32-Bit-Gleitpunkt	-3.402823E+38 bis -1.175495E-38 0.0 +1.175495E-38 bis +3.402823E+38

Darüber hinaus gibt es weitere Datentypen, die andere Bedeutungen als Zahlen haben:
z.B. ANY, COUNTER, CHAR, TIMER. Sie finden eine Zusammenstellung in der TrySim-Hilfe unter <Index> | "Datentypen".

3.1. Zahlen

Mit Bits lassen sich nur die Zahlen 0 und 1 darstellen. Das ist zum Rechnen natürlich zu wenig, und daher werden die Speicher-Bits einer SPS zu verschieden großen Gruppen zusammengefasst, die dann gemeinsam betrachtet auch größere Zahlen codieren können. Als die kleinste dieser Gruppen wird üblicherweise die 8er-Gruppe, bestens bekannt als Byte, angesehen. Weil es überschaubarer ist, soll hier aber zunächst die 4-er Gruppe betrachtet werden, für die es auch die Namen Halb-Byte oder Nibble gibt. Es gibt 16 verschiedene Kombinationen dieser Bits. Allgemein gilt: Die Zahl der Kombinationen bei n Variablen ist 2^n. Für 4 Bits ergeben sich 2^4 = 16 Kombinationen. Weil diese Bit-Muster von Menschen schwer zu erkennen sind, werden sie in der Computertechnik häufig hexadezimal dargestellt. Das bedeutet: Man hat sich darauf geeinigt, die 4-er Bit-Muster so abzukürzen:

Dual	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Dual	Hex
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Ganz wichtig ist: dadurch ist dem Bitmuster noch keine Bedeutung verliehen worden, es ist nur eine Darstellung. 1010 bedeutet nicht „A“, sondern das „A“ wird nur verwendet, um das Bitmuster 1010 abgekürzt auf dem Monitor darzustellen, weil es dann leichter zu lesen ist.

3.2. Codierung nur positiver Zahlen

Um mit den Bitmustern rechnen zu können, muss man ihnen aber eine Bedeutung zuweisen. Diese Zuordnung nennt man Codierung. Dafür gibt es mehrere Möglichkeiten. Die als Dual-Codierung fast jedem bekannte Zuordnung dieser Kombinationen zu den normalen Zahlen ist:

Dual-Darstellung	Hex-Darstellung	Bedeutung im Dezimal-System
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Diese Codierung wird (egal um wie viele Bits es geht) auch „vorzeichenlos“ genannt.

3.3 Codierung positiver und negativer Zahlen

Die Zuordnung der Zahlen 0 – 15 zu den Bitmustern 0000 – 1111 ist die übliche, aber man sollte sich darüber im Klaren sein, dass auch andere Zuordnungen möglich sind. Wenn man z.B. auch negative Zahlen benötigt, könnte man einfach das Bit ganz links als Symbol für das Minuszeichen verwenden und käme dann zu folgender Codierung.

Dual	Dez.
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Dual	Dez.
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

Diese Codierung ist von Rechnern aber nur mit viel Aufwand zu verarbeiten, darum ist die gebräuchliche Codierung von negative Zahlen die Bildung des 2er-Komplementes. Das bedeutet: zur Darstellung einer negativen Zahl werden alle Bits der positiven Zahl umgedreht (aus 0 wird 1 und aus

1 wird 0) und dann eine 1 addiert. Falls sich ein Übertrag auf eine Stelle ergibt, für die gar kein Bit vorhanden ist, wird dieser ignoriert. Die Regel ist nicht ganz einfach zu verstehen und soll am Beispiel der Codierung von –3 erläutert werden:

Die 3 wird dargestellt durch 0011
Dreht man alle Bits um, erhält man 1100
Addiert man dazu eine „1“, ergibt sich: 1101

Durch Anwendung dieser Regel ergibt sich dann eine solche Zuordnung:

Dual	Dez.
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Dual	Dez.
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Sie werden diese Umrechnungen nie selbst durchführen müssen, aber es ist wichtig, zumindest die Grundzüge zu verstehen. Für die praktische Arbeit ist ganz besonders wichtig, dass auch bei dieser Codierung das Bit ganz links immer ein Zeichen für eine negative Zahl ist. Diese Codierung wird (egal um wie viele Bits es sich handelt) auch „vorzeichenbehaftet“ genannt.

3.4. Darstellung und Codierung

Woran man beim Programmieren immer denken muss: Im Computer gibt es nur die Bit-Muster. Und der Computer kann diese zu Ihrer Bequemlichkeit in verschiedenen Darstellungen auf dem Monitor anzeigen. Damit ist aber noch nichts darüber gesagt, welche Bedeutung diese Muster haben, denn die ist dem Computer (wenn man das so sagen kann) vollkommen egal. Sie erst geben diesen Bitmustern eine Bedeutung, indem Sie ein Programm schreiben, das aus einzelnen Rechenbefehlen besteht. Bei jedem der möglichen Befehle (die Sie später in diesem Buch noch lernen werden) ist genau festgelegt, wie die Bitmuster der Operanden zu verstehen sind.

Ohne Codierung, d.h. ohne Interpretation der Zahlenfolge und der Vorgabe dieses Deutungsmusters ist einer Fehldeutung Tür und Tor geöffnet.
Beispiel: Was bedeutet die Ziffernfolge 1110?
Nach vorstehenden Tabellen könnte es dezimal "-2" bedeuten, aber auch "14"!

3.5. Das Byte

Mit Zahlen von 0 bis 15 oder –8 bis 7 kommt man nicht sehr weit. Die nächst größere Gruppe von Bits nennt man Byte und sie besteht aus 8 Bits. Hier gibt es 2^8 = 256 möglichen Bitkombinationen. Wie beim Nibble gibt es dafür verschiedene

- **Darstellungsmöglichkeiten (wie das Bitmuster vom Computer angezeigt wird)** und verschiedene
- **Codierungen (welche Bedeutung das Bitmuster haben soll).**

Die Darstellungsmöglichkeiten sind:
a.) als Bitmuster : 0000 0000 bis 1111 1111
b.) hexadezimal : 00 bis FF
c.) und gelegentlich werden die Muster auch einfach durchnummeriert: von 0 bis 255

Einige der üblichen Codierungen sind:

a.) als vorzeichenlose Zahl, dann geht der Bereich von 0 bis 255
b.) als vorzeichenbehaftete Zahl (in der SPS-Technik eher unüblich), dann geht der Bereich von -128 bis +127
c.) als Schriftzeichen (ASCII), das soll hier nicht betrachtet werden
d.) BCD-(Binary Coded Decimal) Codierung (00 – 99)

a.) und b.) sind genauso wie beim Nibble, nur der Zahlenbereich ist größer geworden. Bei BCD gibt es wieder etwas neues. Für diese Codierung ordnet man jeder Ziffer einer dezimalen (normalen) Zahl ein dualcodiertes Nibble zu. Das lässt sich am einfachsten an einem Bespiel zeigen:

Die Zahl 93 wird durch eine „9“ und eine „3“ codiert. Das dualcodierte Nibble-Bitmuster für „9“ ist 1001, für die „3“ ist es 0011, das resultierende Bitmuster ist also 1001 0011.

Nochmals die gleiche Warnung wie im vorherigen Abschnitt: achten Sie nicht darauf, wie ein Bit-Muster auf dem Monitor dargestellt wird, sondern achten Sie darauf, welche Bedeutung Sie ihm durch die Auswahl des richtigen Bearbeitungsbefehls zuweisen.

Darstellung und Wert einer 8-stelligen **Dualzahl**

1) Operand	E0.7	E0.6	E0.5	E0.4	E0.3	E0.2	E0.1	E0.0	
2) Potenz	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
3) Stellenwert	128	64	32	16	8	4	2	1	
4) Bitmuster dual	1	0	0	1	0	0	1	1	
5) Wert dezimal	128 + 0 + 0 + 16 + 0 + 0 + 2 + 1								$= 147_{dez}$

Darstellung und Wert der gleichen Bit-Kombination als **BCD-Tetraden**

1) Operand	E0.7	E0.6	E0.5	E0.4	E0.3	E0.2	E0.1	E0.0
6) Potenz	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
7) Bitmuster BCD	1	0	0	1	0	0	1	1
8) Wert dezimal	8 + 0 + 0 + 1				0 + 0 + 2 + 1			
9) Wert dezimal	9				3			
10) DEZIMALZAHL	9 3							

3.6. Das Wort

Mit einem Byte kann man immerhin z.B. einen Bereich von 0 bis 255 in Schritten darstellen, die kleiner als 0,5 % sind. Es wird aber heutzutage kaum noch zur Speicherung von Zahlen, sondern im wesentlichen für Buchstaben, eingesetzt. Eine Gruppe von zwei Bytes (also 16 Bits) bezeichnet man als Wort. Die Darstellungsmöglichkeiten sind analog zum Byte und auch die üblichen Codierungen sind ähnlich. Als vorzeichenlose Zahl kann mit einem Wort der Bereich von 0 – 65.535 abgedeckt werden. Dies ist aber eine in der SPS-Technik unübliche Verwendung. Viel wichtiger ist die Verwendung als vorzeichenbehaftete Zahl mit dem Wertebereich –32.768 bis +32.767. Wenn eine Gruppe von 16 Bits so verwendet wird, gibt es in der STEP7-Welt sogar einen eigenen Namen dafür: INT (integer, engl. Ganzzahl). Mit INTs können sie Additionen, Subtraktionen, Multiplikationen und Divisionen durchführen. Sie müssen aber darauf achten, dass das Ergebnis auf jeden Fall innerhalb des Wertebereiches bleibt. Sowie Sie da nicht ganz sicher sind, sollten Sie lieber die Doppel-INTs verwenden, die im nächsten Abschnitt besprochen werden. Auch beim Teilen von INTs durcheinander ist Vorsicht geboten: wenn der Dividend nicht garantiert deutlich größer als der Divisor ist, sollten Sie lieber die REAL-Zahlen verwenden, die im übernächsten Abschnitt besprochen werden.

Wörter werden auch BCD-codiert verwendet. Dabei gibt es in Step7 einige Besonderheiten. Regelmäßig werden nur drei Nibbels zur Darstellung der eigentlichen Zahl verwendet, man erhält also den Zahlenbereich von 000 – 999. Die verbleibenden vier Bits ganz links haben je nach Einsatzfall eine andere Bedeutung:

a.) Bei der Verwendung als Ablaufdauer eines Zeitgliedes ist dort das Zeitraster gespeichert, dies soll hier aber nicht weiter behandelt werden.
b.) Bei der Verwendung als Konstante zum Setzen eines Zählers sind diese Bits immer 0
c.) Wenn das BCD-codierte Wort einfach nur als Zahl betrachtet wird, dann gilt diese als negativ, wenn alle vier Bits „1“ sind. Dadurch erweitert sich dann der Wertebereich auf –999 bis +999.

3.7. Das Doppelwort

Als Fortsetzung der bisherigen Systematik wird hiermit eine Gruppe von 32 Bits bezeichnet. In Bezug auf Zahlen werden Doppelwörter auf zwei Arten verwendet:

Die erste Verwendung ergibt sich ganz zwanglos aus der Verwendung des Wortes als INT: zum Speichern von Zahlen im Bereich –2.147.483.648 bis +2.147.483.647. Wenn ein Doppelwort so verwendet wird, wird es als DINT bezeichnet.

Ein Wertebereich von über 4 Milliarden reicht sicherlich für fast jede Anwendung, wo etwas gezählt wird, aus. Nachteilig an dieser Codierung ist jedoch, dass nur ganze Zahlen codiert werden können, für 3,75 gibt es kein Bitmuster. Daher hat man sich eine weitere Codierung ausgedacht, die in STEP7 als REAL bezeichnet wird und mit der der Zahlenbereich von $– 10^{38}$ bis $– 10^{–38}$ und 0 und von $+10^{–38}$ bis $+10^{38}$ erfasst wird. Um die Zahl "0" ergibt sich eine winzige Lücke. Wie diese Codierung genau aussieht, braucht man als SPS-Programmierer nicht zu wissen. Ganz wichtig ist aber: sie ist vollkommen anders als die Codierung eines DINTs. Sowohl REALs als auch DINTs werden zwar als 32-Bitmuster abgespeichert, aber diese sehen selbst für die gleichen Zahlen absolut anders aus.
Ein Beispiel:
Sie speichern die Zahl 32 als DINT, das ergibt das Bitmuster

0000 0000 0000 0000 0000 0000 0010 0000.

Wenn Sie dieses Bitmuster jetzt einem Rechenbefehl geben, der davon ausgeht, es handele sich um eine REAL-Zahl, dann wird dieser Befehl die Zahl als 0.0000003453 verstehen. Das ist ziemlich weit weg von 32 und ein unerwünschtes Ergebnis ist vorherzusehen.
Beim Rechnen mit REAL-Zahlen brauchen Sie sich über den Wertebereich und die Auflösung meistens keine Gedanken zu machen. Seien Sie nur vorsichtig, nicht durch 0 zu teilen und vergleichen Sie REAL-Zahlen nie auf „gleich“ oder „ungleich“, in der TrySim-Hilfe unter <Index> | "+R", bzw. "==R" steht näheres dazu.

Lesen Sie bitte den Hinweis über die Mischung von INTs und DINTs, wenn was negatives dabei sein könnte, unter der TrySim-Hilfe, <Index> | "DINT"!

3.8. Konstanten (vgl. TrySim-Hilfe, <Index> | "Konstanten")

Konstanten können in verschieden Darstellungen eingegeben werden:

Dual-Konstante, z.B. **2#**1100, jede Stelle steht für ein Bit und darf nur 0 oder 1 sein. diese Konstante kann für Bytes, Words und DWords verwendet werden. Natürlich muss die maximale Anzahl der Bits berücksichtigt werden.

Byte-Konstante hex, z.B. **B#16#**F5, jede Stelle steht für ein Halb-Byte und darf nur 0..9, A..F sein. Der maximale Wert dieser Konstanten ist FF hex oder 255 dezimal.

Word-Kontante hex, z.B. **W#16#**6F34, jede Stelle steht für ein Halb-Byte und darf nur 0..9, A..F sein. Der maximale Wert dieser Konstanten ist FFFF hex oder 65535 dezimal.

Word-Konstante als **zwei dezimale Bytes,** z.B. **B#**(152,43), die beiden Zahlen geben die beiden Bytes des Words an und werden dezimal angegeben. Der maximale Wert dieser Konstanten ist B#(255,255).

DWord-Konstante hex, z.B. **DW#16#**6FA42322, jede Stelle steht für ein Halb-Byte und darf nur 0..9,A..F sein. Der maximale Wert dieser Konstanten ist FFFF FFFF hex oder ca. 4 Milliarden dezimal.

Integer-Konstante, z.B. -5. Diese Konstante entspricht unseren ganzen Zahlen. Der Wertebereich geht von -32.768 bis +32.767

Double Integer-Konstante, z.B. **L#**334124. Diese Konstante entspricht der Integer-Konstante mit einem erweitertem Wertebereich von -2.147.483.648 bis +2.147.483.647. Wenn Sie die Double Integer Operationen +D, -D, *D, /D, <D, >D usw. verwenden und eine negative Konstante laden, ist es wichtig, dass Sie das L# voranstellen.

Real-Konstanten, z.B. 3**.**5 oder 5**.**342124e+002. Wenn Sie mit Realzahlen rechnen oder diese vergleichen und dabei Konstanten verwenden ist es wichtig, dass Sie den '**.**' mit angeben. 5 und 5**.**0 sind etwas total verschiedenes für eine SPS!

S5-Time-Konstanten, z.B. **S5T#**1H2M. Die Syntax ist unter S5Time erklärt. Diese Konstante benötigen Sie, um Timer zu starten. Verwenden Sie auf keinen Fall eine Konstante vom Typ TIME, z.B. T#200MS !

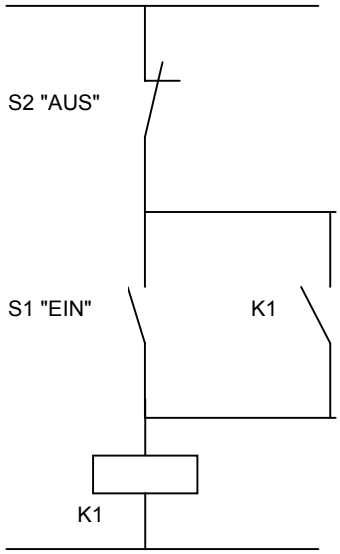
Zähler-Konstanten, z.B. **C#**59. Hiermit laden Sie die angegebene Zahl BCD -codiert. Der Wertebereich geht von C#0 bis C#999.

Time-Konstanten, z.B. **T#**5D2H5M. Hiermit laden Sie die angegebene Zeitdauer in ms als double integer (DINT).

Wenn Sie für eine bestimmtes Problem eine **Operation** suchen, aber den Code dafür nicht kennen, empfiehlt sich, in der TrySim-Hilfe unter <Index> | "Operationen" nachzuschauen. Dort finden Sie einen gut geordneten Überblick über alle Operationen.

Was kann eine SPS?

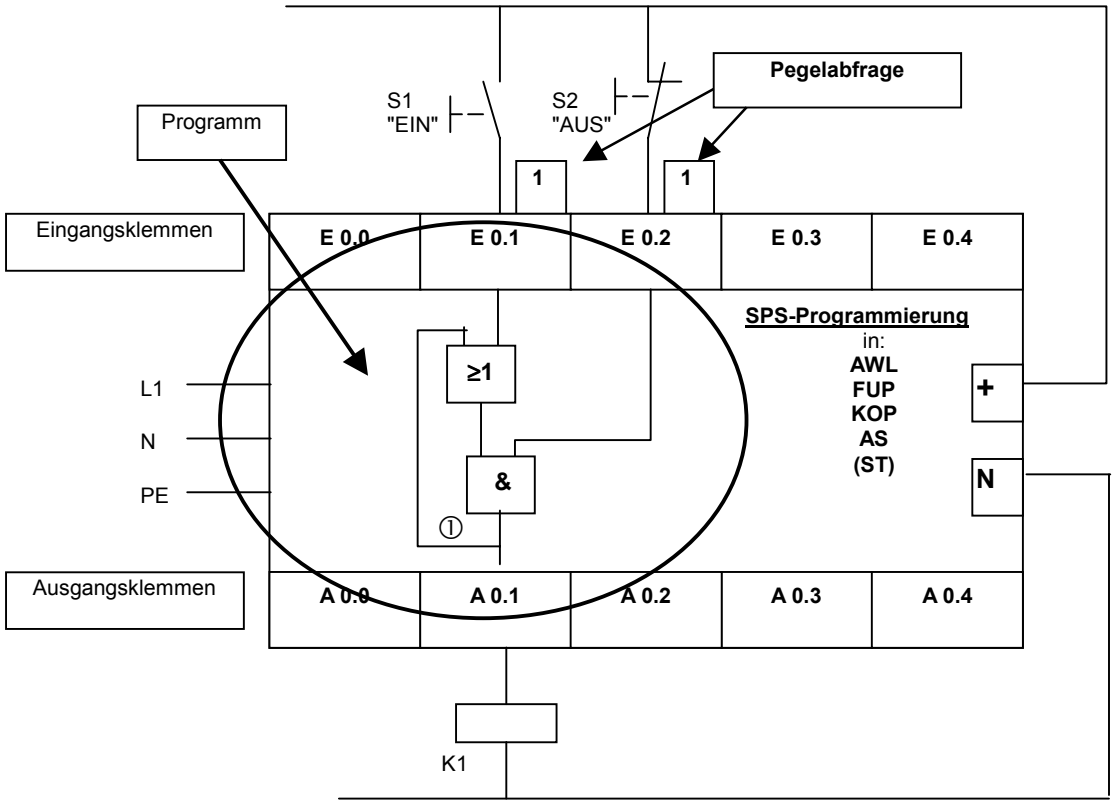
A) Logische Bit-Verknüpfungen



4.1. Prinzip der SPS-Programmierung:
Alle Befehlsorgane (z.B. Taster, Sensoren, Relais, Lastschütze, Ventilspulen, etc.) werden jeweils **einzeln** an Ein- und Ausgangsklemmen angeklemmt.
Die Verknüpfungslogik wird also nicht mehr durch die Verdrahtung erzeugt, sondern dafür wird das Programm der SPS geschrieben. Das Programm fragt die Spannungspegel an den entsprechenden Ein- und Ausgangsklemmen ab.
Der Programmierer muss entscheiden,

1) Welches Befehlsorgan an welche Klemme gelegt werden soll.
2) Welche Befehlsorgane bei der Abfrage betätigt werden sollen.
3) Welche nicht betätigten Befehlsorgane auch noch abgefragt werden müssen.
4) Welche Spannungspegel zum Einschalten des gewünschten Ausganges vorhanden sein sollen.

Die Verknüpfungslogik kann aus der Schützensteuerung übernommen werden. Dabei ist auf die **logisch richtige Reihenfolge** (siehe weiter unten) zu achten. Im obigen Stromlaufplan darf nur S1 betätigt werden, damit der Stromkreis für K1 geschlossen wird. Dem entsprechend muss an der SPS auch der Taster S1 gedrückt werden >> an der Klemme E 0.1 liegt dann eine logische 1 an. Würde oben S2 betätigt, wäre der Stromkreis unterbrochen. Sinngemäß darf S2 (Öffner) an der SPS auch nicht betätigt werden. Dann liegt im Ruhezustand an der Klemme E 0.2 ebenfalls eine logische 1 an. Auch Ausgangsklemmen können abgefragt werden ①.



4.2. Welche Programmiersprache ist zu wählen?

Die älteren und bekannteren Programmiersprachen für SPS sind

AWL **A**N**W**EISUNGS**L**ISTE

FUP **FU**NKTIONS**P**LAN

und KOP **KO**NTAKT**P**LAN

AWL ist die "eigentliche" Programmiersprache, die der Rechner versteht, d.h., das Programm wird in Textform zeilenweise geschrieben und gelesen.
FUP und KOP sind grafische Darstellungen, die im Hintergrund auch wieder in AWL-Befehle übersetzt werden. Sie erleichtern die Lesbarkeit des Programms und vereinfachen die Befehlseingabe.

Hilfreich ist, dass es die Software erlaubt, durch Mausklick von einer Sprache in die andere zu wechseln. Dabei ist schön zu sehen, wie z. B. ein Zählerbaustein im FUP von der Software in eine AWL-Anweisung umgeschrieben wird. In der AWL wird der vollständige Baustein mit allen möglichen Ein- und Ausgängen dargestellt, auch wenn man nicht alle Anschlussmöglichkeiten im FUP genutzt, bzw. benötigt hat. Die nicht belegten Ein- u. Ausgänge tauchen dann in der entsprechenden Zeile als NOP-Befehl (no operation) auf. Man kann diese Zeilen in AWL auch löschen, das Programm arbeitet trotzdem einwandfrei. Allerdings ist dann dieses Rumpfprogramm nicht nach FUP oder KOP zu übersetzen. Warum also sollte man sich die Mühe machen, und die korrekte zeilenweise Struktur in AWL schreiben, wenn der ganze Baustein auch mit Mausklick entstehen kann?

Das Ganze ist mehr eine "Geschmacks-" oder Erfahrungssache, je nachdem, welche Darstellungsart man gewöhnt ist. Der Nachrichtentechniker wird wohl auf FUP schwören und der Elektriker aus der Welt der Schütztechnik könnte sich wahrscheinlich eher mit KOP anfreunden.

Es gibt ein Problem, das je nach Sprache unterschiedlich groß wird:
Angenommen, eine etwas aufwändigere Schützkontaktverriegelung mit Reihen- u. Parallelschaltungen der Kontakte soll durch eine SPS ersetzt werden.

Die Kernfrage ist: **Bei welchen Kontakten beginne ich mit dem SPS-Netzwerk?**

Beginne ich wie bei der Schützensteuerung oben und arbeite mich Kontakt für Kontakt nach unten?

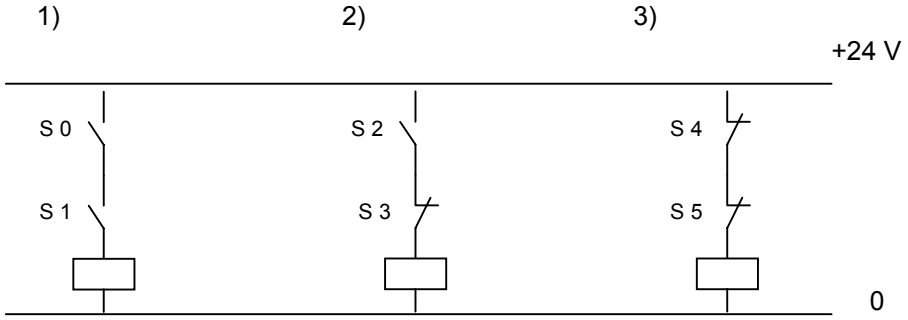
Der KOP heißt nicht von ungefähr so, sondern hier werden genau die Kontakte der Schützen-Steuerung in gleicher Anordnung nachgebildet, nur von links nach rechts und nicht von oben nach unten.
In AWL ist durch die Klammerbildung auch die gleiche Reihenfolge nachzubilden. Würde man das selbe aber im sonst so übersichtlichen FUP machen, führt dieser „beliebte“ Fehler zum Funktionschaos. Denn:

entscheidend ist die logisch richtige Reihenfolge.

Teile der Schaltung müssen vorrangig zusammengefasst werden und können erst dann mit anderen Abschnitten der Schaltung verknüpft werden.
Wer das beherrscht, findet am richtigen Schaltungsaufbau im FUP schnell seine Freude.

4.3. Welche Spannungspegel sind zu programmieren, „0“ oder „1“ ?

Beispiel: UND-Verknüpfungen (verdrahtungsprogrammiert)



Fragestellung: Wann ziehen die Relais an?
Das Relais zieht jeweils an, wenn der Stromkreis geschlossen ist. Dabei ist es egal, ob der Strom über unbetätigte Öffner oder über betätigte Schließer fließt. Der gleiche Sachverhalt (also wenn die gleiche Bediensituation gegeben ist) muss beim Anschluss der Taster an die SPS zum gleichen Ergebnis führen. Das SPS-Programm soll genau die Einschaltlogik der Schaltung ersetzen.
An der SPS wird jeder Sensor (z.B. Taster) an einen eigenen Eingang gelegt. Die logische Verknüpfung wird erst durch das SPS-Programm erreicht.

Ein Programm wird immer für die Einschaltsituation (und nicht für das Ausschalten !) programmiert. Das Ausschalten ergibt sich wegen der zyklischen Abfrage von allein, wenn nämlich die Einschalt-bedingung nicht (mehr) gegeben ist.

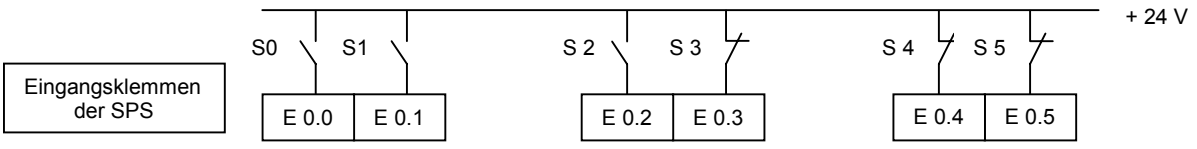
Im Fall 1) müssen in obiger Schaltung beide Taster betätigt werden.
Aber nicht das Betätigen ist für das SPS-Programm das Kriterium, sondern die Abfrage, welches Potenzial (z.B. 24 V = logisch "1" oder TRUE; 0 V = logisch "0" oder FALSE) an den Klemmen anliegt. Da beide Taster geschlossen sein müssen, sind an beiden SPS-Eingängen "1"-Abfragen zu programmieren.

Im Fall 2) müssen auch wieder beide Taster geschlossen sein. Natürlich dürfen Sie dafür nur Taster S2 betätigen. Wenn sie geschlossen sind, liegt wieder Spannung ("1") an den Eingängen an.

Im Fall 3) gelten genau die gleichen Bedingungen: Der Stromkreis ist nur geschlossen, wenn beide Taster geschlossen sind, also kein Taster betätigt wird.

Sie merken schon: Es ist völlig belanglos[1] für das Programm, ob Sie Schließer oder Öffner einsetzen: Wichtig ist der Spannungspegel, der anliegen **soll**, damit die Verknüpfungsbedingung erfüllt ist.

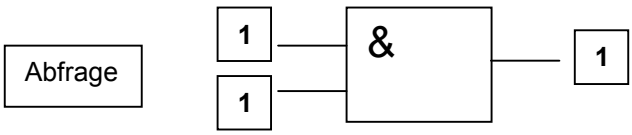
Alle 3 Schützensteuerung würden trotz ganz unterschiedlicher Tasterbestückung durch dasselbe SPS-Programm ersetzt werden! Es sollte ja jeweils ein Programm erstellt werden für den Fall, dass bei den **vorhandenen** Tastern das Relais anzieht. Dafür gilt praktisch nur eine Bedingung: Der Stromkreis muss geschlossen sein oder sinngemäß: Alle Kontakte müssen geschlossen sein.



erwarteter Signalzustand	1	1		1	1		1	1

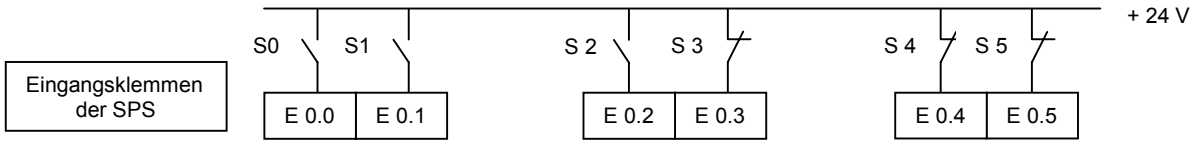
[1] Das bedeutet aber nicht, dass es völlig egal ist, ob Sie Schließer oder Öffner verwenden.
Sie müssen schon entscheiden, ob ein Taster und welcher betätigt werden soll oder darf. Ein Taster zum Ausschalten einer Anlage muss aus Sicherheitsgründen (Drahtbruchsicherheit) immer ein Öffner sein. Eingeschaltet wird immer mit einem Schließer.

In allen 3 Fällen gilt dieselbe Abfrage, bzw. Verknüpfung.



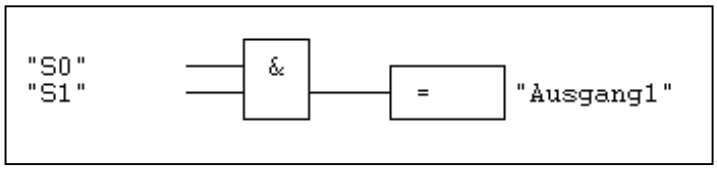
Wie sieht der Sachverhalt aber aus, wenn alle 3 Relais anziehen sollen und der Maschinenführer nur den Auftrag hat: „Drücke auf beide Taster gleichzeitig"?

Dann lautet nicht das Kriterium: „Der Stromkreis muss geschlossen sein", sondern von der Eingangsseite her ist zu fragen: „Auf welche Signalzustand soll das Programm reagieren"?
Daraus ergeben sich folgende Abfragen:

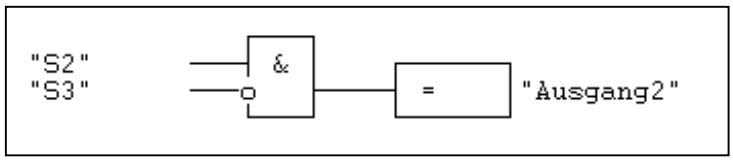


erwarteter Signalzustand:	1	1	1	0	0	0

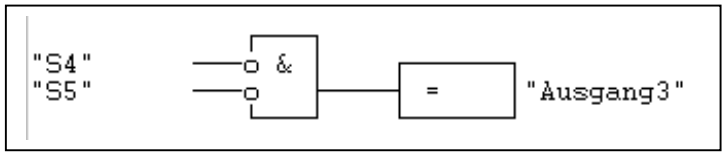
Fall 1)



Fall 2)



Fall 3)



Bemühen Sie sich bitte gar nicht erst, für Fall 2 und 3 gleichwertige Schützensteuerungen zu erstellen. Die sind als Vorlage für das Programm überhaupt nicht erforderlich, sondern es reicht völlig aus, die gewollten (!) Spannungszustände (Pegel) zu programmieren. Die am Eingang anliegende "0" wird vor der Verknüpfung in eine "1" invertiert.

4.4. Sicherheitsaspekte

Die Arbeit mit diesem Buch soll Sie befähigen, SPS-Programme zu schreiben. Im Mittelpunkt steht also die Software und deren Anwendung. Dennoch wäre es geradezu sträflich, wenn der Eindruck entstünde, die Projektierung eines funktionsfähigen Programms allein wäre ausreichend.
Ebenso wichtig ist es, die notwendige Sensibilität für Gefahren zu entwickeln, die von der Anlage im Betrieb, aber auch bei Störungen ausgehen können. Auch das Verhalten von elektrischen bzw. mechanischen Komponenten und die Auswirkungen von Fehlschaltungen sind zu berücksichtigen.

Allerdings würde es den Rahmen dieses Buches sprengen, diese Punkte ausreichend zu erläutern. Es folgen einige Anregungen, die Sie ggf. selbst vertiefen müssten.

4.4.1. Maschinenrichtlinie

Die Anwendung der „Maschinenrichtlinie“ ist seit dem 1. Januar 1995 bindend. Seitdem müssen neu auf den Markt gebrachte Maschinen eines Herstellers dieser Richtlinie entsprechen. Die Richtlinie 89/3921EG legt die wesentlichen allgemein gültigen Sicherheits- und Gesundheitsanforderungen fest. Die Maschinen müssen ihre Funktion gewährleisten sowie eingestellt und gewartet werden können, ohne dass dabei Risiken für Personen entstehen.
Insbesondere dürfen die Steuersysteme bei einem Fehler in der Schaltkreislogik keine Gefahrensituationen verursachen (Not-Aus, Schutzgitter-Überwachung, Zweihandsteuerung, . . .).

Die „Einsatzrichtlinie“ verlangt eine Anpassung der Maschinenparks an die Konformitätsbestimmungen bis spätestens 1. Januar 1997 (mit Ausnahme von Sonderregelungen). Von dem Zeitpunkt an müssen alle Arbeitsmittel (Maschinen), die den Beschäftigten in einem Unternehmen zur Verfügung stehen, den genormten Sicherheitsvorschriften entsprechen. Die „Einsatzrichtlinie“ legt die Mindestvorschriften für den Einsatz von Arbeitsmitteln in bezug auf Sicherheit und Gesundheit der Arbeiter fest
Alle Arbeitsmittel müssen für die auszuführende Arbeit geeignet und entsprechend angepasst sein, um so die Betriebsfähigkeit der Maschinen und die Sicherheit der Arbeiter gewährleisten zu können.

Beachtet werden muss die Gesamtheit der Aktivitäten an den Arbeitsmitteln, wie Inbetriebnahme, Abschalten, Betriebseinsatz, Transport, Reparatur, Umbau, Instandhaltung, Wartung und Reinigung.

Der Unternehmer sorgt dafür, dass die mit dem Einsatz der Arbeitsmittel betrauten Arbeiter eine entsprechende Ausbildung, auch in bezug auf die Risiken des jeweiligen Einsatzes, erhalten.

4.4.2. NOT-AUS

Jede Maschine muss mit einem oder mehreren klar erkennbaren und schnell zugänglichen Not-Aus-Einrichtungen versehen sein. Sie müssen unabhängig von der Betriebsart in möglichst kurzer Zeit das Anhalten des gefahrvollen Vorgangs bewirken. Ein Not-Aus kann möglicherweise einige Sicherheitsschritte auslösen oder deren Auslösung ermöglichen. Er darf allerdings auf keinen Fall durch eine trennende oder eine andere Schutzeinrichtung ersetzt werden.
Ausgenommen von dieser Verpflichtung sind:
- tragbare Maschinen,
- Maschinen, bei denen eine Not-Aus-Einrichtung das Risiko nicht verringert.

Stop-Funktion (nach EN 60 204-1 und EN 418)
Jede Maschine muss mit einer Stop-Funktion der Kategorie 0 ausgerüstet sein.
Der Not-Aus muss als ein Stop der Kategorie 0 oder 1 wirken.
Man unterscheidet drei Stopkategorien:
Stop-Kategorie 0:
Stillsetzen durch sofortiges Ausschalten der Energiezufuhr zu den Maschinenantrieben.
Stop-Kategorie 1:
Gesteuertes Stillsetzen, wobei die Energiezufuhr zu den Maschinenantrieben beibehalten wird, um das Stillsetzen zu erzielen und die Energiezufuhr erst dann unterbrochen wird, wenn der Stillstand erreicht ist.

Stop-Kategorie 2:
Gesteuertes Stillsetzen, bei dem die Energiezufuhr zu den Maschinenantrieben erhalten bleibt.

Für die Not-Aus-Funktion der Stop-Kategorie O dürfen nur festverdrahtete, elektromechanische Bauteile verwendet werden.
Die Auslösung darf nicht von einer Schaltlogik oder von der Übertragung von Befehlen über ein Kommunikationsnetzwerk oder eine Datenverbindung abhängen.
Bei der Stop-Kategorie 1 für die Not-Aus-Funktion muss die endgültige Abschaltung der Energieversorgung der Maschinenantriebe sichergestellt sein und muss durch Verwendung von elektromechanischen Bauteilen erfolgen.

Verlässlichkeit
Der Not-Aus besteht aus einer Steuereinrichtung (oder Sicherheitsschaltgerät) und einem Betätigungselement, das nach dem Prinzip der mechanischen Zwangsbetätigung arbeitet. Die Kontakte der Not-Aus-Einrichtungen müssen Kontakte mit Zwangsöffnung sein.

Sicherheit
Nach Betätigung bleibt der Not-Aus-Taster blockiert. Durch eine entsprechende Betätigung kann er wieder gelöst werden, ohne dass dabei jedoch die Maschine wieder anläuft.

4.4.3. Bewegliche Schutzeinrichtung

Definition (nach EN 1088)
Die Schutzeinrichtungen verhindern den Zugang zu Gefahrenbereichen. Zweck dieser Schutzeinrichtungen ist es, den Zugang zu sich bewegenden Arbeitsmitteln zu verhindern oder einzuschränken.
Diese Schutzvorkehrungen können mit Verriegelungsvorrichtungen installiert und ausgerüstet sein, im allgemeinen Endlagengeber, Mini-Grenztaster oder andere Sicherheitssensoren.
Wenn die Verriegelungseinrichtung durch Verschieben der Schutzeinrichtung betätigt wird, werden 2 Arten von Informationen erzeugt:

- Schutzeinrichtung geöffnet ⇨ Die Verriegelungseinrichtung gibt einen Stop-Befehl aus.
- Schutzeinrichtung geschlossen ⇨ Die Verriegelungseinrichtung ermöglicht das Anfahren der Maschine. Das Schließen der Schutzeinrichtung bewirkt nicht von sich aus das Anfahren der Maschine.

Höchstes Sicherheitsniveau Kategorie 3 oder 4
Hier fordert die Norm eine Verriegelungseinrichtung bestehend aus zwei mechanisch betätigten Positionsschaltern. die nach den beiden entgegengesetzten Betriebsarten funktionieren.
Einer mit einem normalerweise geschlossenen Kontakt (Öffnerkontakt), der von der Schutzeinrichtung zwangsläufig betätigt wird.
Der andere mit einem normalerweise geöffneten Kontakt (Schließerkontakt), der von der Schutzeinrichtung nicht zwangsläufig betätigt wird.

Sicherheitsniveau Kategorie 1 oder 2 (pr. EN 954-1)
In dem Fall genügt der Norm eine Verriegelungseinrichtung bestehend aus einem mechanisch betätigten Positionsschalter, der zwangsläufig betätigt wird und mit einem Kontakt mit Zwangsöffnung ausgestattet ist.

Ausschalten der Schutzeinrichtung
Die Schutzeinrichtungen und die Verriegelungseinrichtungen werden so konzipiert und installiert, dass ein Ausschalten vermieden wird.

4.4.4. Zweihandsteuerung

Definition (gemäß pr. EN 574)
Eine Zweihandsteuerung erfordert die gleichzeitige Betätigung mit beiden Händen, um den Betrieb einer Maschine zu starten und aufrechtzuerhalten, solange Risiken bestehen. Sie muss sich unbedingt außerhalb des Gefahrenbereichs befinden, damit der Bediener nicht in diesen Bereich eindringen kann, bevor die Maschine vollständig stillsteht.
Eine Zweihandsteuerung besteht aus zwei Teilen:
- Einem Bedienpult mit zwei Betätigungselementen, die so konzipiert sind, dass unbedingt beide Hände verwendet werden müssen.
- Einem Steuerkreis oder Sicherheitsschaltgerät (verbunden mit dem Bedienpult), das sich **unbedingt in einem Schaltschrank** befinden muss.

Synchrone Betätigung
Die Zeitspanne zwischen dem ersten und dem zweiten Signal der beiden Betätigungselemente muss **kleiner als 500 ms** sein.

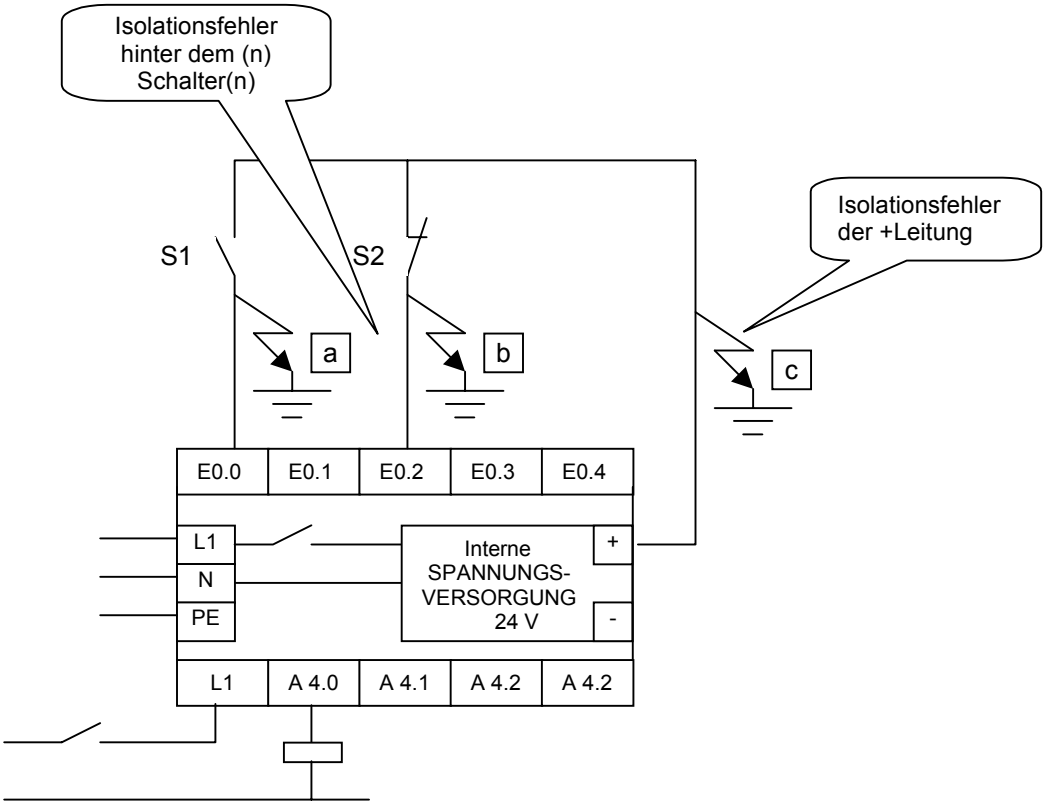
Re-Initialisierung des Ausgangssignals
Das Loslassen eines einzigen Betätigungselements bewirkt die Unterbrechung des Ausgangssignals, die Reinitialisierung ist nur möglich, wenn beide Betätigungselemente losgelassen wurden.

4.4.5. Wesentliche harmonisierte Normen in bezug auf die Sicherheit

EN 292-1/2	Sicherheit von Maschinen Grundbegriffe Allgem. Gestaltungsleitsätze	**EN 418**	Sicherheit von Maschinen NOT-AUS-Einrichtungen Funktionelle Aspekte
EN 60 204-1	Sicherheit von Maschinen Elektr. Ausrüstung von Maschinen	**pr. En 1088**	Sicherheit von Maschinen Verriegelungseinrichtungen Bewegliche Schutzeinrichtungen
pr. EN 954-1	Sicherheit von Maschinen Sicherheitsbezogene Teile von Steuerungen Tabelle der Risiken und Kategorien	**pr. EN 574**	Sicherheit von Maschinen Zweihand-Schalteinrichtungen

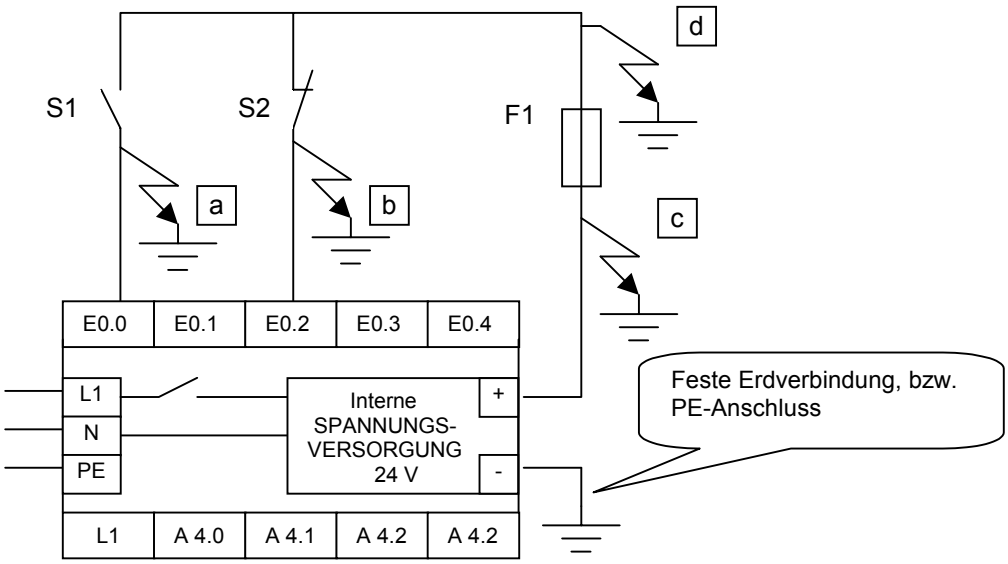
Auf den nächsten Seiten finden Sie einige Erläuterungen zur Erdschluss-Sicherheit einer SPS-Anlage. Ebenso sind Anschlussmöglichkeiten für die Stop-Kategorie 0 dargestellt, aus denen Sie entnehmen können, dass einige Verriegelungen nicht dem SPS-Programm überlassen werden dürfen.

4.4.6. Erdschluss im isolierten Netzsystem

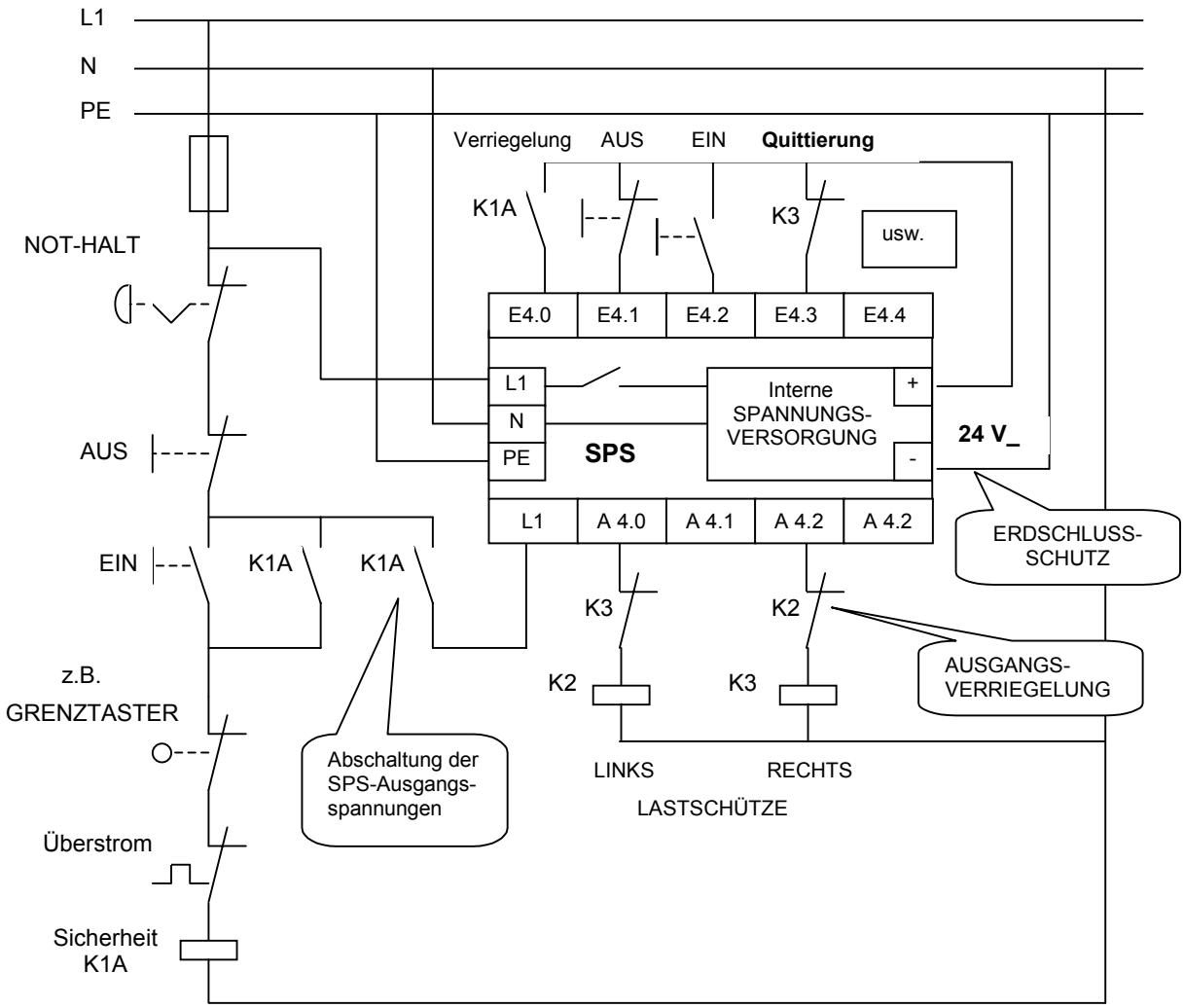


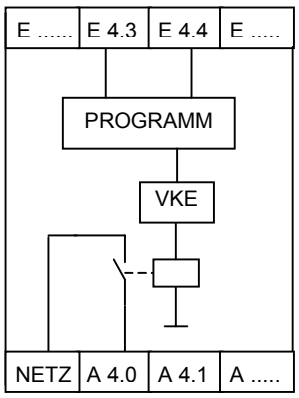
Ohne Isolationsüberwachung wird ein Isolationsfehler gegen Erde an keiner Stelle erkannt und ist praktisch noch unschädlich. Entsteht aber zusätzlich ein Isolationsfehler auf der jeweils anderen Tasterseite, wird der Taster durch die doppelte Erdberührung überbrückt.
Der überbrückte <EIN>-Taster S1 schaltet die Anlage ungewollt ein. Der überbrückte <AUS>-Taster S2 verhindert das Ausschalten der Anlage. **Beides sind evtl. tödlich wirkende Fehler!**

Abhilfe schafft nur die Erdung der -Leitung der Steuerspannung bzw. deren Anschluss an PE des Netzes, weil dann beim ersten Isolationsfehler der +Leitung - egal, ob vor oder hinter dem Schalter - aus dem Erdschluss ein Kurzschluss wird und die Sicherung des Netzgerätes die Steuerspannung abschaltet. Evtl. kann auch noch extern in die +Leitung eine angepasste Sicherung gesetzt werden, so dass die Spannungsversorgung der SPS erhalten bleibt und nur die Eingänge spannungslos werden.



4.4.7. Anschluss einer SPS (Lastspannung: 230 V)



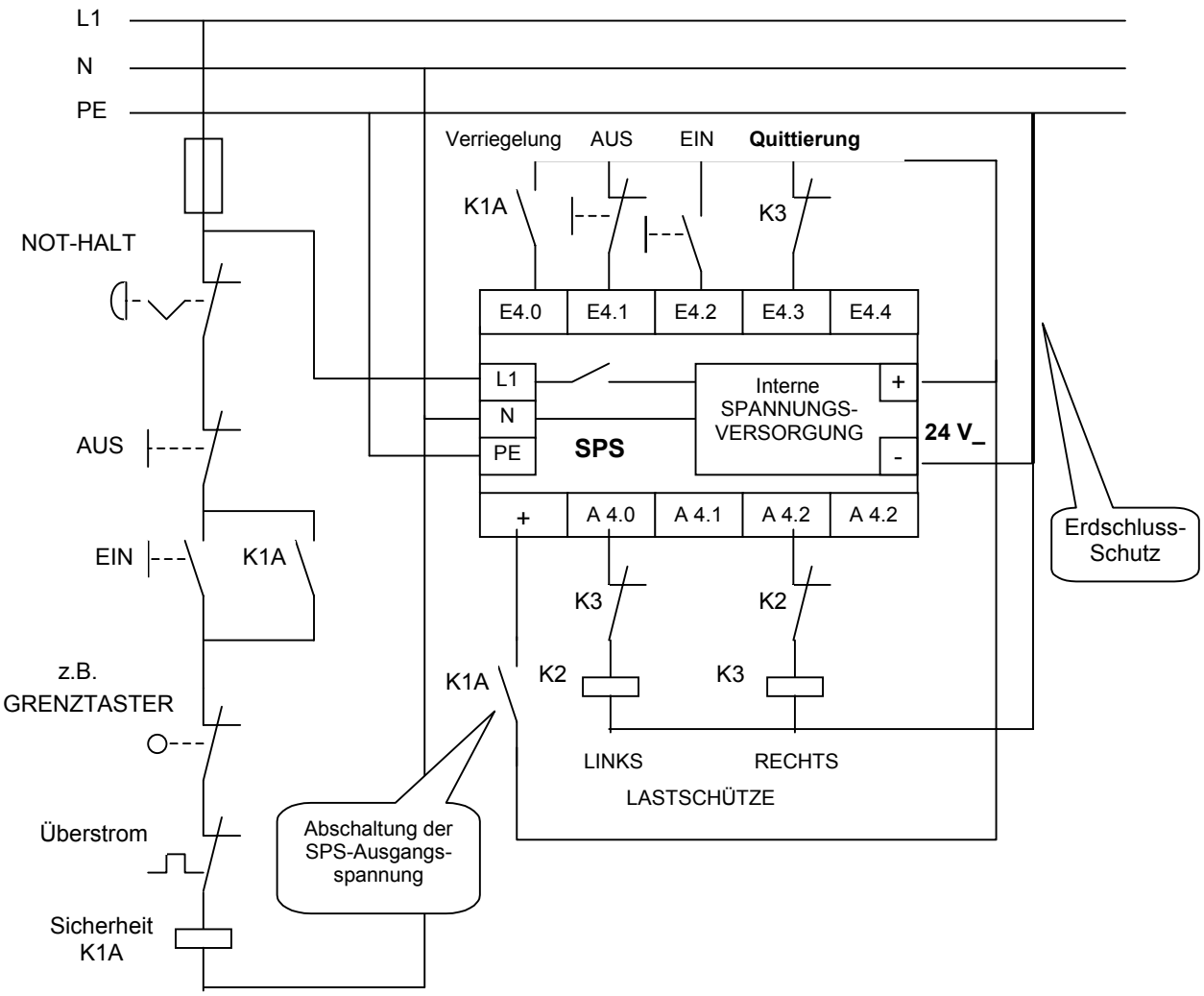


Prinzipdarstellung
Abschaltbare Ausgänge
(Ausschnitt)

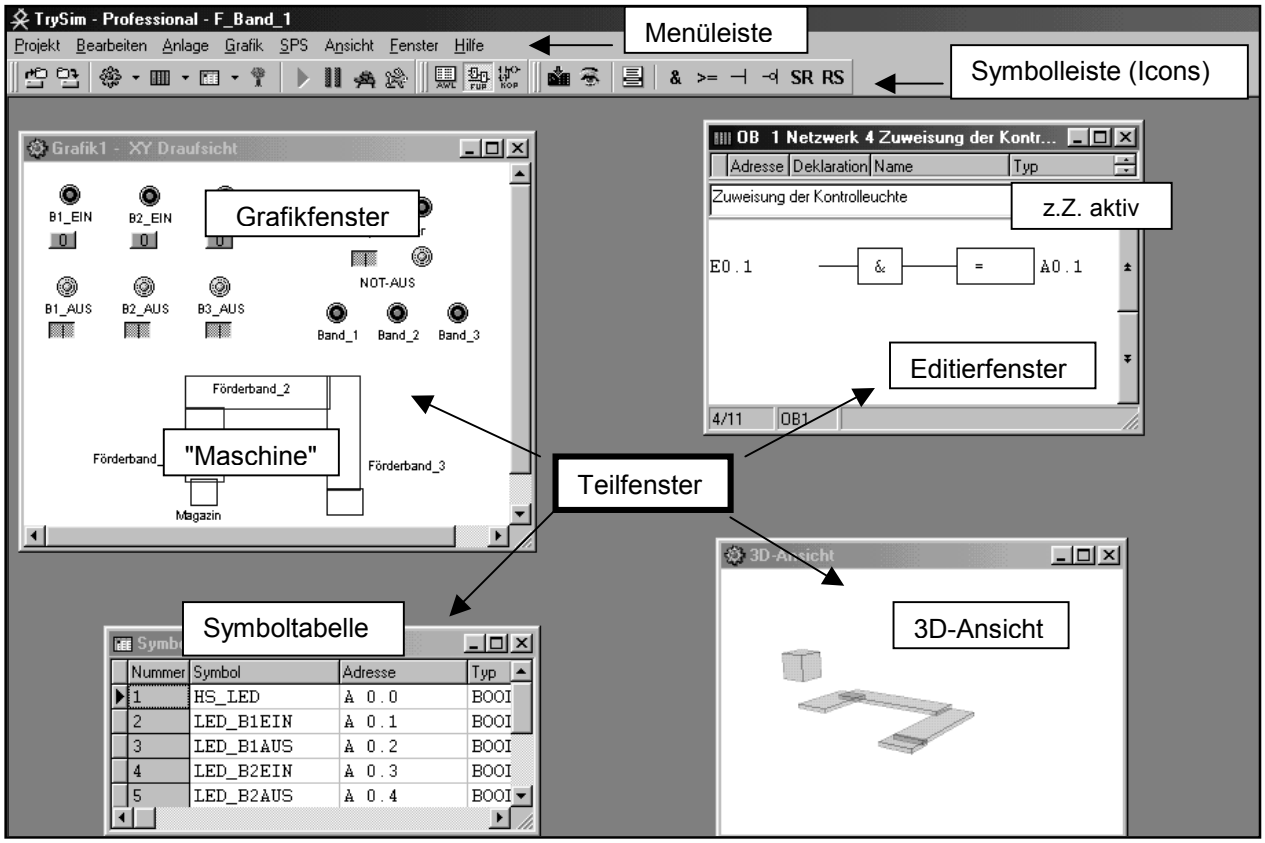
Aus Sicherheitsgründen reicht es nicht aus, einen Ausgang durch das Verknüpfungsergebnis (VKE) des Programms einzuschalten. Die Elektronik könnte z. B. durch Überspannung schadhaft werden, d.h. die Endstufen werden leitend und die Schütze würden beliebig einschalten.
Deshalb wirkt das VKE nicht direkt auf den Ausgang, sondern schaltet innerhalb der SPS einen Kontakt, der in Reihe zum äußeren Sicherheits-Schütz liegt.
Die Lastkreisspannung wird über den Kontakt des Sicherheitsschützes K1A an die entsprechende SPS-Klemme geschaltet. Wenn K1A abfällt, kann keine Schaltspannung an den Lastschützen liegen, auch wenn das Programm-VKE "1" wäre.

Aufgabe: Die Spulenspannung der **Motor**schütze soll 24 V_ betragen. Die Spannung wird der SPS entnommen. Ändern Sie die Schaltung (ohne Sicherheitsverlust!).

Anschluss einer SPS (Lastspannung: 24 V_)

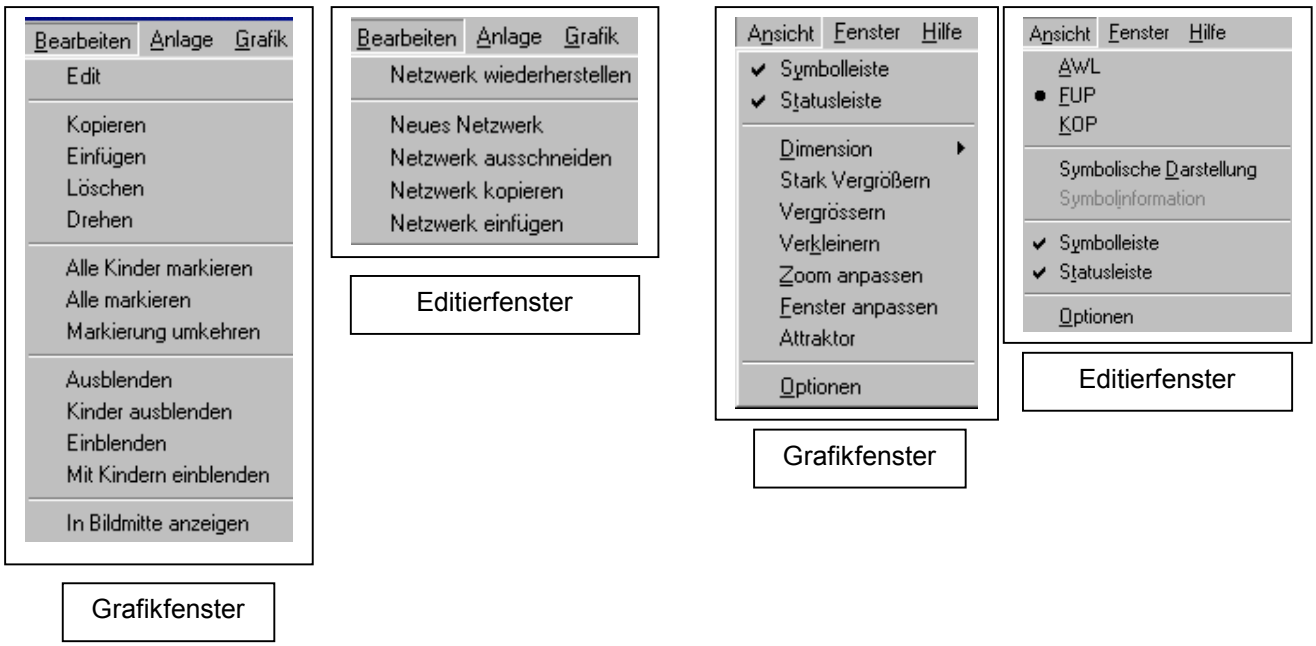


4.5. Praxis: Programmoberfläche von TrySim



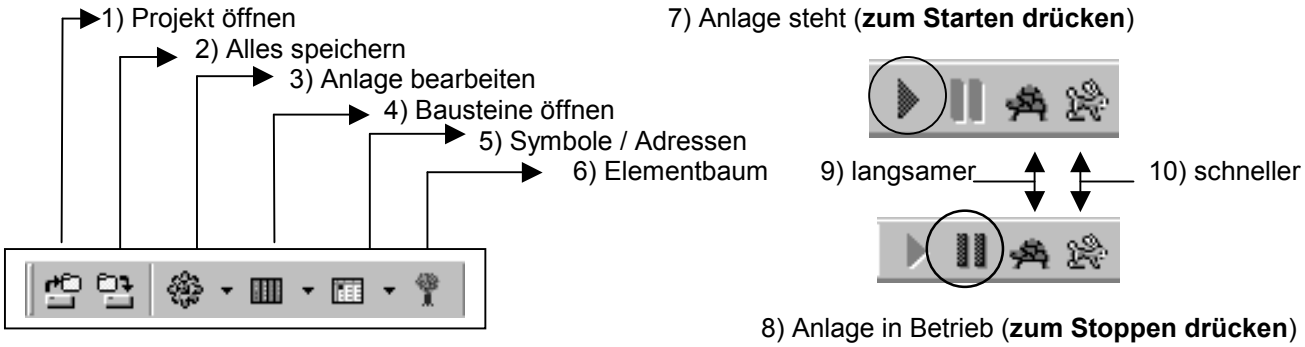
Sie können die verschiedensten Teilfenster gleichzeitig darstellen, aber immer nur in einem zur Zeit arbeiten.
Aktiviert wird ein Teilfenster, indem man mit der linken Maustaste irgendwo in das gewünschte Teilfenster klickt. Die Kopfzeile dieses Fensters ist dann blau dargestellt.
Je nachdem, ob das Grafik- oder das Editierfenster aktiv ist, ändern sich hinter einigen Menüpunkten die Untermenüs, bzw. die Icons der Symbolleiste ändern sich.

<Bearbeiten> **<Ansicht>**



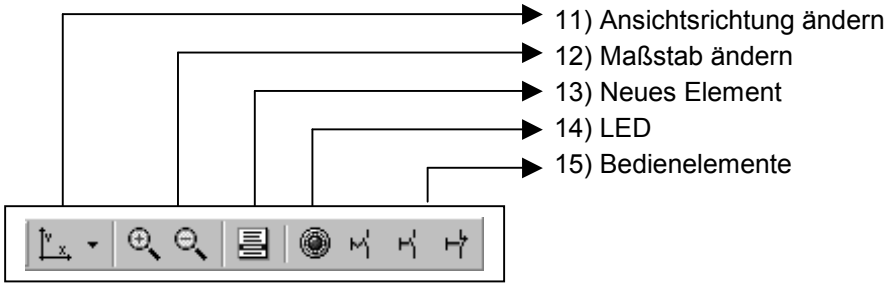
4.6. Übersicht über die Schaltflächen / Icons

Immer sichtbar

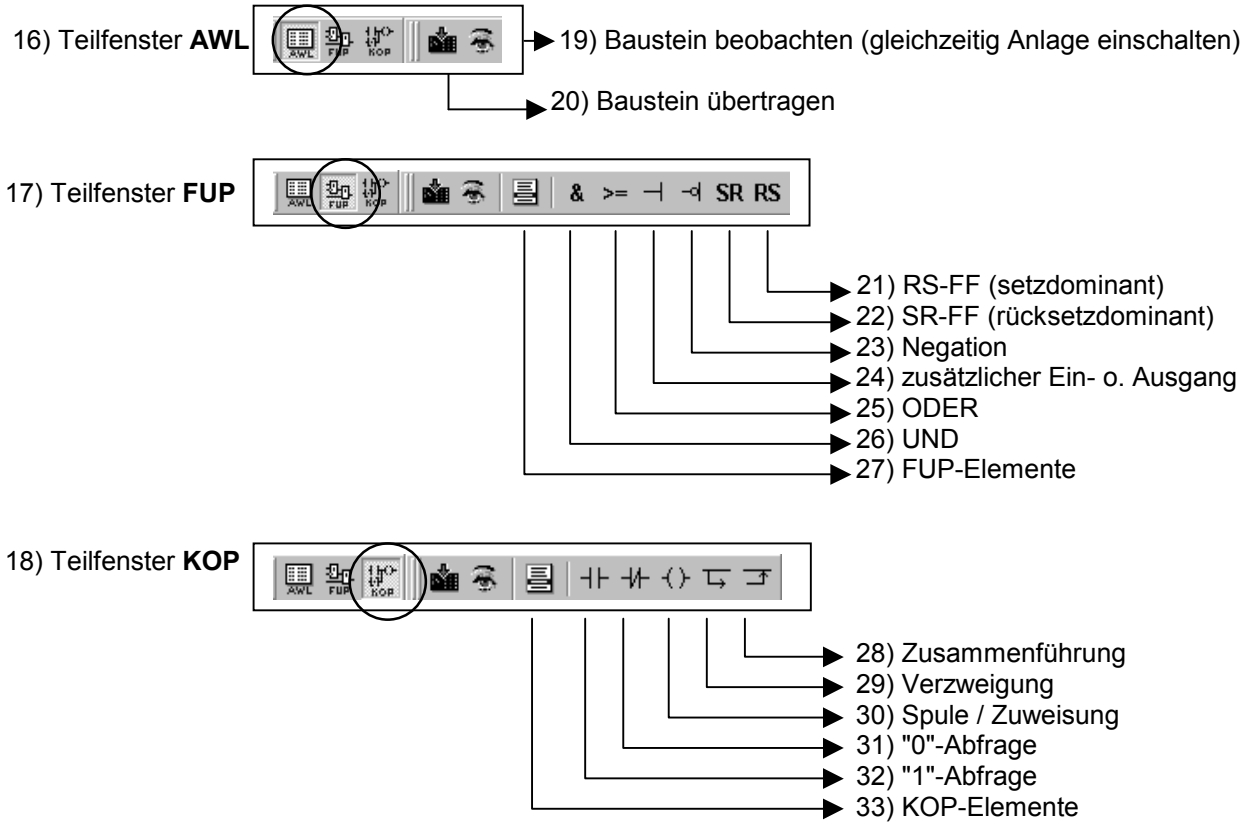


Je nach aktivem Fenster werden verschiedene Icons angezeigt

Grafikfenster aktiv



Editierfenster aktiv - Je nach gewählter Sprache werden verschiedene Icons angezeigt

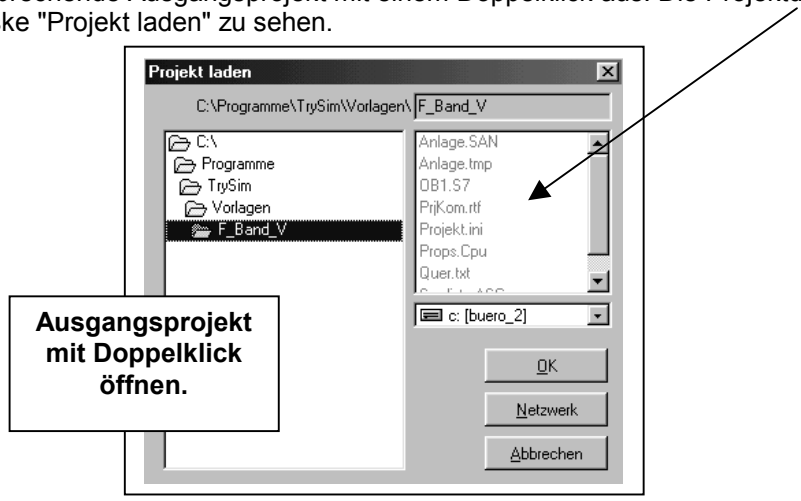


4.7. Anlegen eines neuen Ordners

Sie können sich mit dem Windows-Explorer an beliebiger Stelle einen neuen Ordner für Ihre eigenen Projekte anlegen. Sie können den neuen Ordner aber auch direkt aus TrySim heraus erstellen, wie gleich erläutert wird.

Laden eines Projektes

Wählen Sie das entsprechende Ausgangsprojekt mit einem Doppelklick aus. Die Projektdateien sind dann im rechten Feld der Maske "Projekt laden" zu sehen.



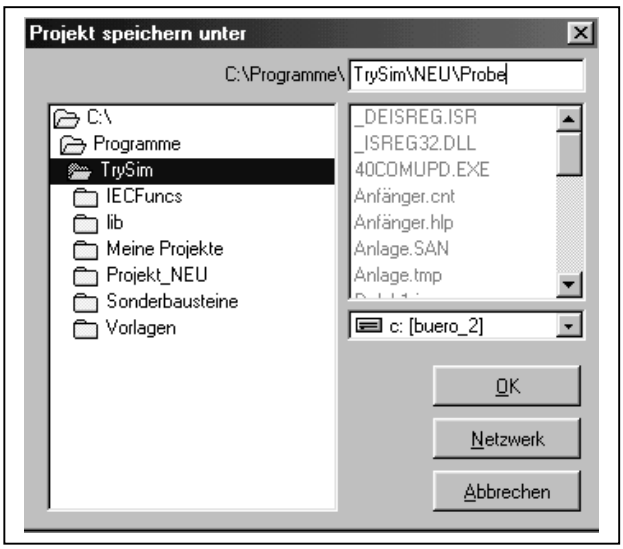
4.8. Speichern eines neuen Projektes

Falls Sie ein bestehendes Projekt oder eine Projektvorlage in ein anderes Verzeichnis kopieren wollen, können Sie das mit den bekannten Möglichkeiten im Windows-Explorer tun, oder den Vorgang unter TrySim direkt abwickeln.
Ein Projekt, das Sie unter TrySim speichern wollen, muss bereits geöffnet sein.

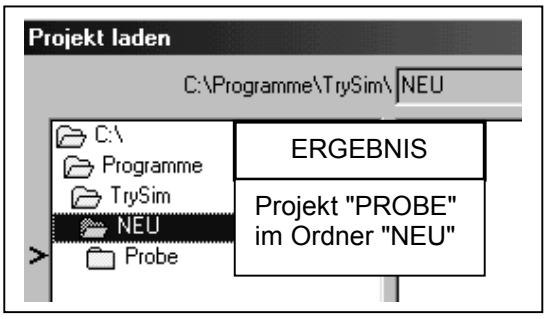
- Wählen Sie unter **<Projekt>|<Speichern unter...>,**
- schließen das geöffnete Verzeichnis mit Doppelklick auf <TrySim>,
- wählen dort ein gewünschtes Verzeichnis mit Doppelklick oder
- tragen oben rechts **hinter dem Fenstertext** ein: **\ <Dateiname>** und
- quittieren mit **<OK>**.

<Dateiname> steht stellvertretend für einen beliebigen Projektnamen.

- Falls Sie einen neuen Ordner für ein neues Projekt gleichzeitig anlegen wollen, tragen Sie hinter dem Fenstertext ein: **\<Ordnername>\<Dateiname>** und quittieren mit **<OK>.**



Vergessen Sie bitte nicht das erste "\"



4.9. Kurzanleitung zum Testen eines fertigen Projektes

Beispiel: Ordner: <Lösungen>, Projekt: <Für Anfänger>

Die ausführliche Beschreibung folgt auf der nächsten Seite.

Das Programm TrySim startet normalerweise mit dem zuletzt geöffneten Projekt.
Sie können wie folgt wechseln:

- Klicken Sie auf **<Projekt>** und dann auf **<Öffnen>**.
- Falls der Programmexplorer irgend ein Verzeichnis anzeigt, wählen Sie - falls Sie bei der Installation den Vorschlägen gefolgt sind - den Pfad über c:\Programme\TrySim.
- Dort öffnen Sie z.B. das Verzeichnis **<Lösungen>** mit <LM>-Doppelklick und wählen das Projekt **<Für Anfänger>** mit <LM>-Doppelklick aus.
- Öffnen Sie, falls nicht sichtbar, **<Grafik>** und dann **<Grafik 1 -XY-Draufsicht>.**
- Öffnen Sie, falls nicht sichtbar, **<SPS>** und dann **<Öffnen>** .
- Markieren Sie **OB1** (wird blau) und bestätigen mit **<OK>**.
- Klicken Sie in das **Editierfenster** (Kopf wird blau = aktives Fenster).
- Zum Start der Simulation und zur gleichzeitigen Kontrolle der Pegel im Editierfenster klicken Sie auf das **Auge-Icon.**

- Betätigen Sie im Grafikfenster die Schalter und beobachten Sie die Auswirkungen.
- Wählen Sie im Grafikfenster mit dem Scrollbalken weitere Netzwerke aus und stellen Sie im Editierfenster das dazugehörige Netzwerk mit den Pfeiltasten am rechten Fensterrand ein. (Notfalls verschieben Sie bitte das Editierfenster so, dass der rechte Fensterrand mit den Pfeiltasten zu sehen ist. Wenn Sie das Fenster aus Versehen zu hoch im Hauptfenster verschoben haben und nicht mehr die blaue Kopfzeile erreichen können, ist jedoch automatisch ganz rechts im Hauptfenster ein neuer Scrollbalken erschienen, mit dem Sie Ihr Teilfenster wieder "in Position" bringen können).
- Sie können jedes Teilfenster beliebig vergrößern oder verschieben. Der Inhalt des Grafikfensters lässt sich mit der Lupe vergrößern oder verkleinern.

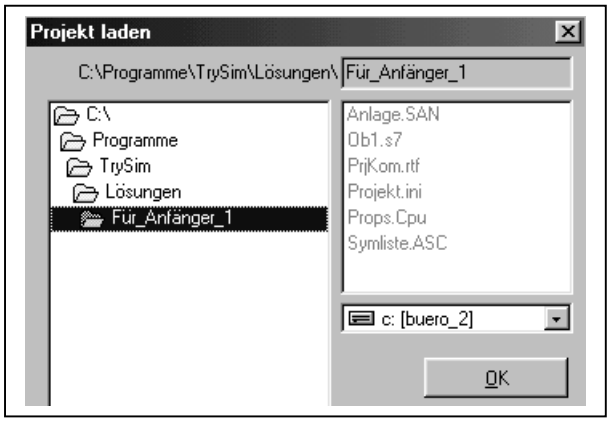
Sie können jedes Projekt jederzeit in der Windows-üblichen Weise verlassen.

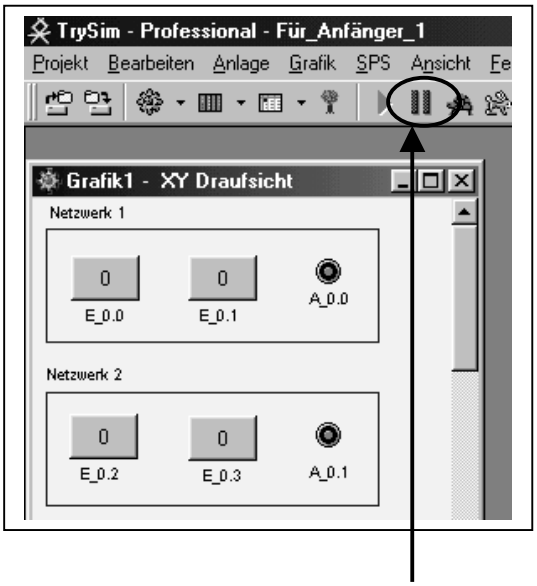
5. Projektarbeit

Projekt 0: Anwendungsübung mit TrySim

Schwerpunkte: Grundverknüpfungen

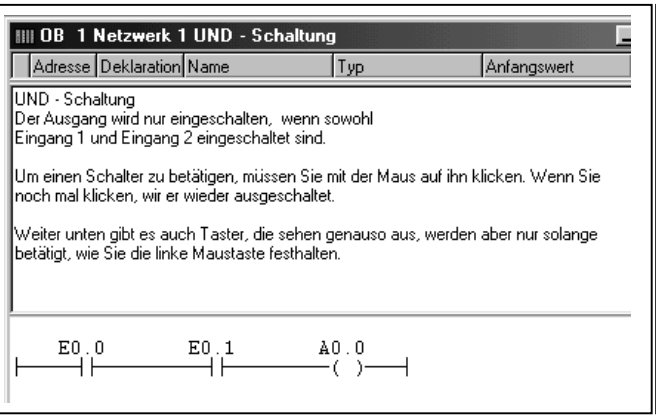
Zu Beginn dieses Kurses werden Sie mit einem bereits fertig erstellten Projekt - einer Auflistung vieler Grundverknüpfungen - an die praktische Arbeit und an die Verknüpfungstheorie herangeführt.
Dazu laden Sie bitte unter **<Projekt>|<Öffnen>** aus dem Verzeichnis **<Lösungen>** die Datei **<Für_Anfänger_1>,** indem Sie die Datei **mit Doppelklick** markieren und **<OK>** betätigen.





Es öffnet sich das rechts dargestellte Fenster, das als Ausschnitt abgebildet ist. Wenn das Stop-Icon blau markiert ist, können Sie sofort das erste Netzwerk testen. Sonst müssen Sie das Start-Icon links daneben (das Dreieck) betätigen. Sie sehen 2 Taster und eine LED, die einem Verknüpfungsausgang zugeordnet wurde.

Wie die beiden Taster verknüpft sind, sehen Sie im Editierfenster.

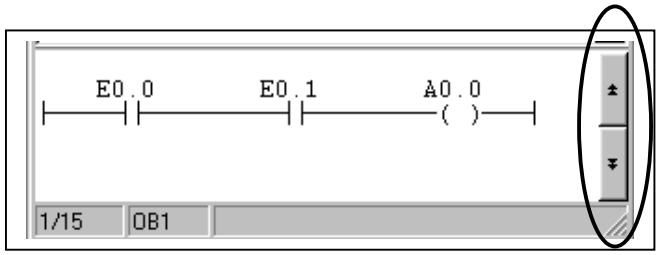


In jedem Netzwerk finden Sie einen zugehörigen Kommentar, so dass Sie ohne weitere Anleitung jedes Netzwerk erklärt bekommen und ausprobieren können.
Die Programmiersprache ist über die entsprechenden Icons umzuschalten.

Durch rote Leitungsführung in FUP und KOP können Sie die "1"-Pegel verfolgen, wenn Sie das Auge-Icon einschalten.

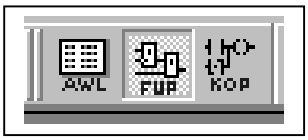
Falls das Fenster nicht zu sehen ist, können Sie es öffnen mit <SPS>|<Öffnen>. Dort markieren Sie bitte <OB 1> und drücken <OK>.

Mit dem Scrollbalken am Grafik1- Fenster können Sie die weiteren "Anlagen" erreichen. Die zugehörigen Netzwerke sind mit den Buttons rechts im Editierfenster zu wählen.



Damit Sie die Darstellung in den Sprachen AWL, FUP und KOP auch ohne PC besser vergleichen können, sind alle Netzwerke auf den folgenden Seiten noch einmal zusammenhängend und in allen Sprachen aufgeführt.

Vergleichen Sie das Projekt: <Für_Anfänger>
im Verzeichnis <Lösungen>



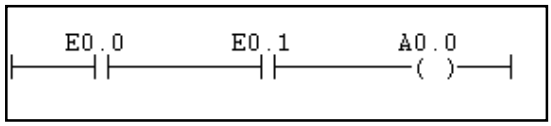
Sie können die Sprachen umschalten.

Sie können mit <Für_Anfänger_V> im Verzeichnis <Vorlagen> die Schaltungen selbst nachbauen. Die Anleitung erfolgt allerdings erst in den nächsten Projekten, in denen das Editieren erläutert wird.

OB 1; Netzwerk 1: UND – Schaltung (vgl. TrySim-Hilfe |<Index> „Und“)

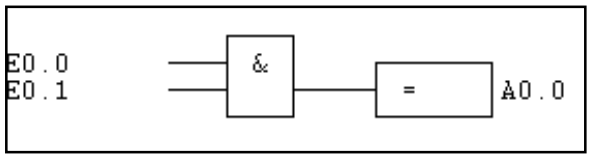
Der Ausgang wird nur eingeschalten, wenn sowohl Eingang 1 und Eingang 2 eingeschaltet sind.

Programmier-Sprache **KOP**



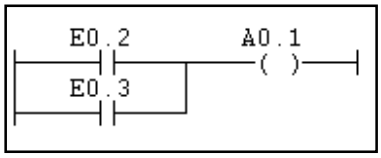

```
AWL:  U  E  0.0
      U  E  0.1
      =  A  0.0
```

Programmier-Sprache **FUP**

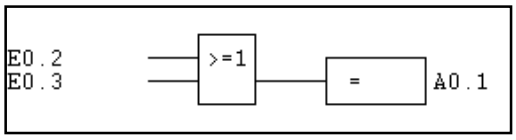


OB 1; Netzwerk 2: ODER – Schaltung (vgl. TrySim-Hilfe |<Index> „Oder“)

Der Ausgang wird eingeschaltet, wenn entweder einer oder beide Eingänge eingeschaltet sind.
Natürlich können auch mehr als 2 Eingänge ODER verknüpft werden.

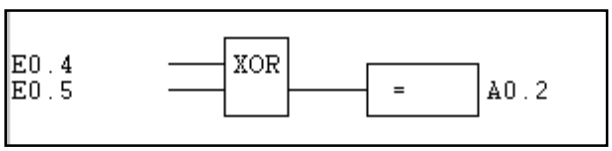



```
O  E  0.2
O  E  0.3
=  A  0.1
```



OB 1; Netzwerk 3: Exclusives ODER (XOR) (vgl. TrySim-Hilfe |<Index> „X“)

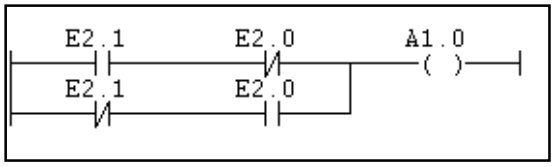
Der Ausgang wird eingeschaltet, wenn einer der Eingänge eingeschaltet ist, aber nicht wenn beide Eingänge geschaltet sind.
Dieses Netzwerk lässt sich nicht in KOP darstellen. Im nächsten NW ist gezeigt, wie diese Funktion dennoch in KOP programmiert werden kann.
Wenn die XOR-Box mehr als 2 Anschlüsse hat, wird der Ausgang dann eingeschaltet, wenn eine ungerade Anzahl von Eingängen auf EIN ist.



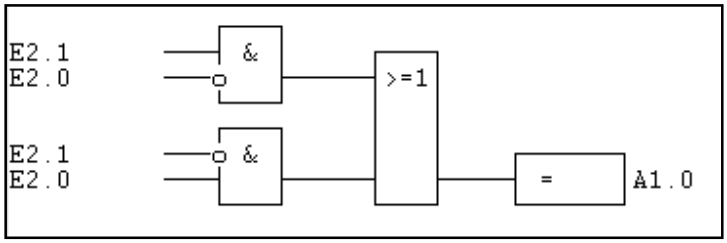

```
X  E  0.4
X  E  0.5
=  A  0.2
```

OB 1; Netzwerk 4: EXCLUSIVES ODER, in KOP darstellbar

Hier ist zum ersten mal von Öffnern Gebrauch gemacht worden. Durch einen Öffner-Kontakt fließt der Strom, wenn der Eingang NICHT eingeschaltet ist.

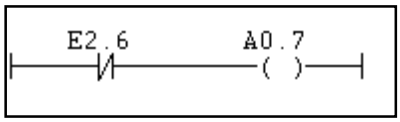



```
U  E  2.1
UN E  2.0
O
UN E  2.1
U  E  2.0
=  A  1.0
```

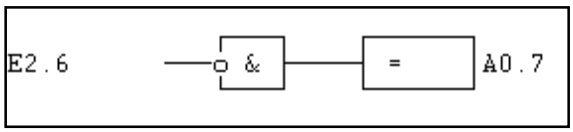


OB 1; Netzwerk 5: Negation

Durch eine Negation wird aus dem Signal "0" eine "1" und umgekehrt.
In KOP entspricht dies einem Öffner, der durch das Symbol --|/|-- dargestellt wird.
In FUP wird sie durch einen kleinen Kreis gekennzeichnet.

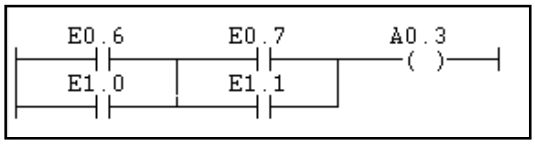


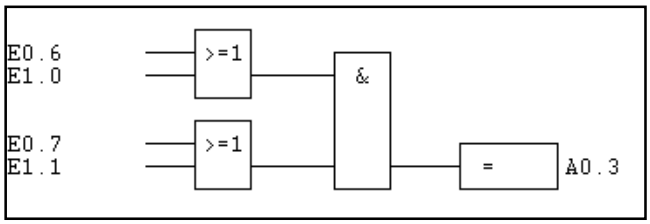

```
UN E  2.6
=  A  0.7
```



OB 1; Netzwerk 6: ODER vor UND

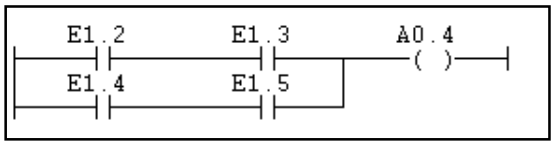
Beachten Sie, dass in KOP der Strom anders als in tatsächlichen Schaltungen immer nach rechts fließt, nie nach links. In diesem speziellen Fall macht es keinen Unterschied, in manchen größeren Netzwerken schon.



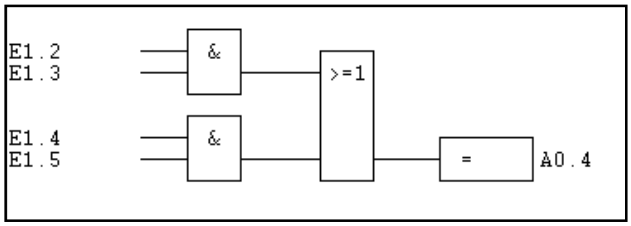



```
U(
O  E  0.6
O  E  1.0
)
U(
O  E  0.7
O  E  1.1
)
=  A  0.3
```

OB 1; Netzwerk 7: UND vor ODER

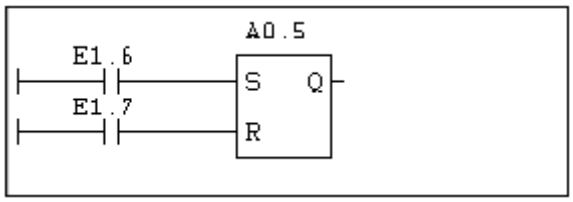



```
U E 1.2
U E 1.3
O
U E 1.4
U E 1.5
= A 0.4
```

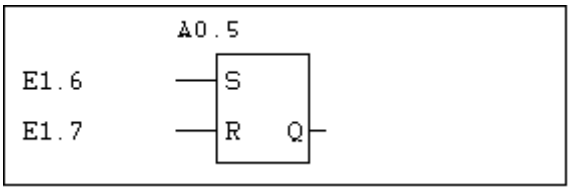


OB 1; Netzwerk 8: FlipFlop mit vorrangigem Rücksetzen (TrySim-Hilfe |<Index> „Flip-Flop“)

Der Setz-Eingang muss nur einmal kurz eingeschaltet werden, das FlipFlop
bleibt dann auf "1", bis der Rücksetzeingang eingeschaltet wird.
Wenn beide Eingänge eingeschaltet sind, dominiert der untere (hier R).
Warum das so ist, erkennt man, wenn man sich dieses NW in AWL anzeigen lässt.
Der Ausgang A 0.5 wird zwar zuerst gesetzt, aber dann wieder zurückgesetzt.
Wenn in einem SPS - Programm ein Ausgang gesetzt wird, bedeutet das NICHT,
dass das daran angeschlossene Gerät (in diesem Fall der Leuchtmelder) sofort
eingeschaltet wird. Sondern es wird zunächst nur ein Bit im Speicher, dem Prozessabbild
Ausgänge (PAA), gesetzt. Erst am Ende eines Programmzyklusses wird der Zustand
des Bits dann auf die Klemme übertragen, die mit A 0.5 beschriftet ist.

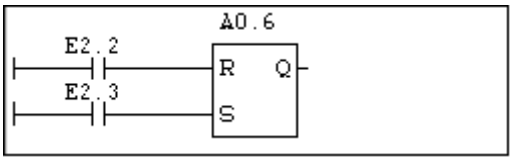


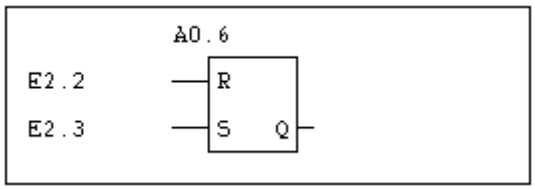

```
U E 1.6
S A 0.5
U E 1.7
R A 0.5
NOP 0
```



OB 1; Netzwerk 9: FlipFlop mit vorrangigem Setzen

Dieser Typ wird häufig verwendet, um Störzustände zu speichern. An den Setz-Eingang kommt das Signal, an dem die Störung erkannt wird, z.B. eine Motorschutzschalterabfrage, und an den Rücksetz-Eingang der Taster "Störung quittieren". Da das Setzen vorrangig ist, ist sichergestellt, dass die Störung erst dann quittiert werden kann, wenn sie behoben ist. Verwendet man hierfür das FlipFlop aus dem vorherigen NW, kommen die Bediener von Maschinen tatsächlich auf die Idee, den Taster "Störung quittieren" mit einem Streichholz festzuklemmen, damit es irgendwie weitergeht. Die Folgen kann man sich ausmalen.



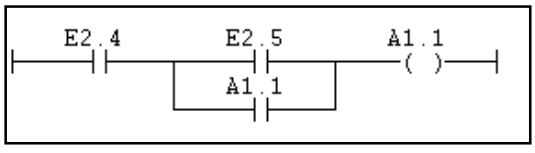



```
U  E   2.2
R  A   0.6
U  E   2.3
S  A   0.6
NOP 0
```

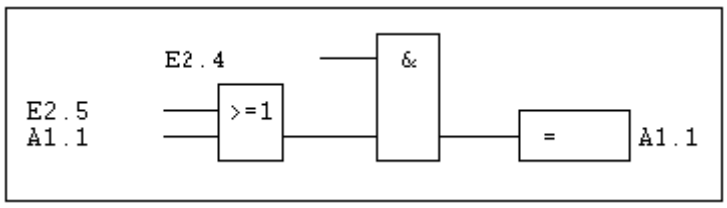
Hier erkennen Sie gut, dass der Setz-Befehl nach dem "R"-Befehl kommt und somit dominiert. Es wird immer die letzte gültige Verknüpfung an den Ausgang geschickt.

OB 1; Netzwerk 10: Selbsthaltung

Diese exakte Nachbildung der Signalspeicherung mittels eines Schützes, das sich über einen seiner Kontakte selbst hält, wird i.A. nicht so häufig wie die FlipFlops verwendet. Wir haben sie hier trotzdem aufgeführt, weil sie sehr schön zeigt, dass man beim Programmieren sehr genau darüber nachdenken muss, ob ein Eingangssignal durch "Spannung ein" oder durch "Spannung aus" dargestellt wird.
Wer einmal eine Selbsthaltung mit einem Schütz aufgebaut hat, wird zu diesem NW zunächst sagen: das ist falsch, zum Ausschalten muss ein Öffner verwendet werden! Es ist auch ein Öffner verwendet worden, aber in der Anlage, nicht im Programm.

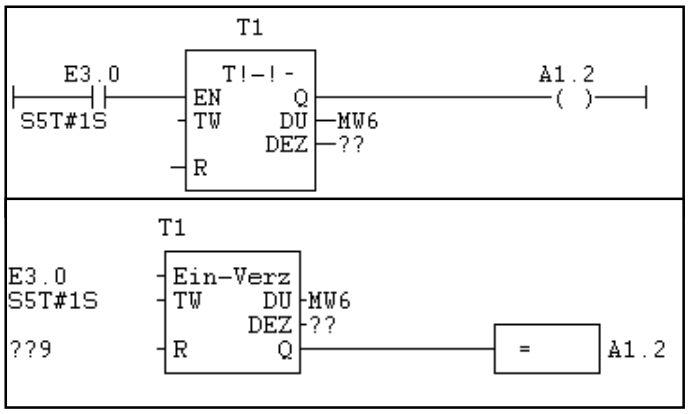



```
U  E   2.4
U(
O  E   2.5
O  A   1.1
)
=  A   1.1
```



OB 1; Netzwerk 11: Einschaltverzögerung (vgl. TrySim-Hilfe |<Index> „SE“)

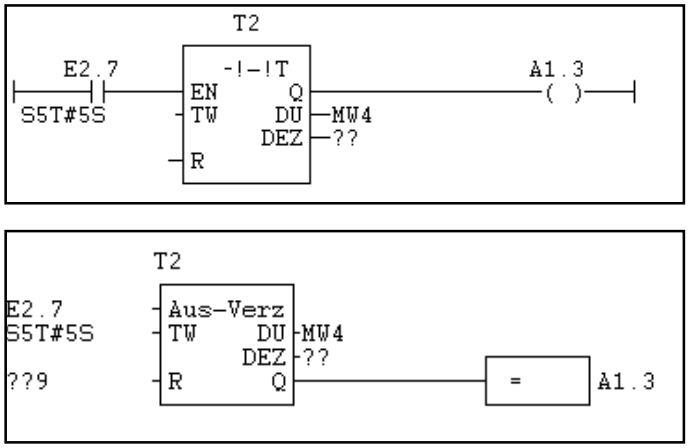
In der Automatisierung kommt es sehr häufig vor, dass ein Eingangssignal erst ausgewertet werden soll, wenn es eine bestimmte Zeitlang ansteht.
Damit der Melder leuchtet, müssen Sie den Taster E 3.0 anklicken und festhalten.
Sowie Sie den Taster loslassen, erlischt der Melder.
Auf dem Oszillographen sehen Sie die ablaufende Zeit.




```
U  E   3.0
L   S5T#1S
SE T   1
NOP 0
L   T   1
T   MW  6
NOP 0
U  T   1
=  A   1.2
```

OB 1; Netzwerk 12: Ausschaltverzögerung (vgl. TrySim-Hilfe |<Index> „SA“)

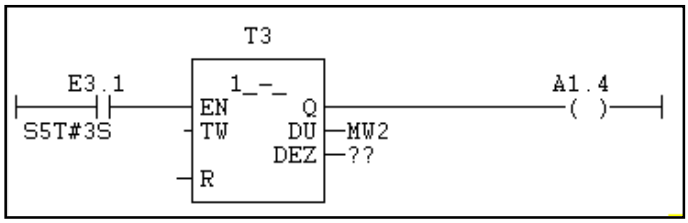
Diese kennt jeder aus dem täglichen Leben: Treppenhausbeleuchtung.



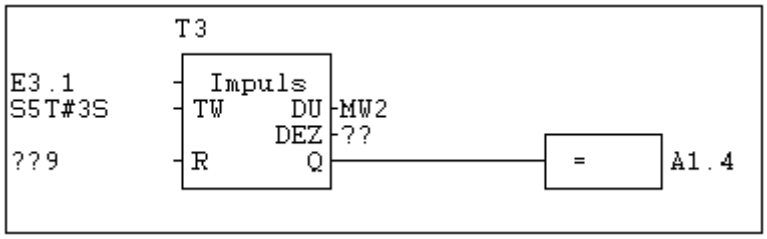

```
U  E   2.7
L   S5T#5S
SA T   2
NOP 0
L   T   2
T   MW  4
NOP 0
U  T   2
=  A   1.3
```

OB 1; Netzwerk 13: Zeit als Impuls (vgl. TrySim-Hilfe |<Index> „SI“)

Hier bleibt der Leuchter eingeschaltet, solange Sie auf den Taster klicken, aber höchstens 3 Sekunden.

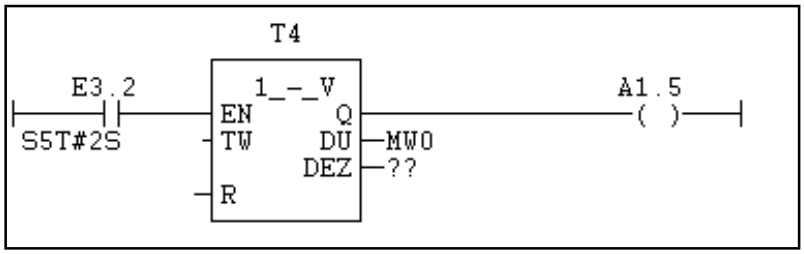



```
U  E   3.1
L  S5T#3S
SI T   3
NOP 0
L  T   3
T  MW  2
NOP 0
U  T   3
=  A   1.4
```

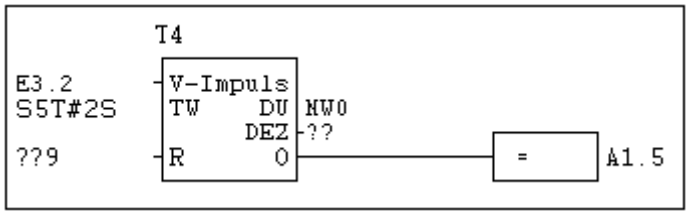


OB 1; Netzwerk 14: Zeit als verlängerter Impuls (vgl. TrySim-Hilfe |<Index> „SV“)

Hier wird der Leuchter bei Betätigung des Tasters immer für 2 Sekunden lang eingeschaltet, unabhängig davon, ob der Taster kürzer oder länger betätigt wird.
Sie können den Leuchter trotzdem länger leuchten lassen, indem Sie den Taster während dieser 2 Sekunden kurz loslassen und dann wieder betätigen.

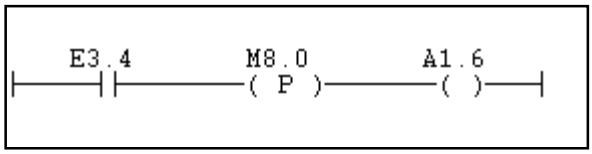



```
U  E   3.2
L  S5T#2S
SV T   4
NOP 0
L  T   4
T  MW  0
NOP 0
U  T   4
=  A   1.5
```

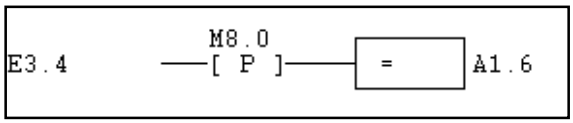


OB 1; Netzwerk 15: Flanken-Erkennung (vgl. TrySim-Hilfe |<Index> „FP“ bzw. „FN“)

Gelegentlich ist es nützlich ein Signal zu haben, das nur ganz kurz genau in dem Moment ansteht, wenn ein Eingang gesetzt wird. Dazu gibt es die Flankenerkennung. Sie verwendet einen zusätzlichen Speicher (Merker genannt) in dem der Zustand des Eingangs beim letzten Zyklus gespeichert wird. Nur dann, wenn der Eingang im letzten Zyklus "0" war und er in diesem Zyklus "1" ist, ist der Ausgang der Flankenerkennung "1". Dann wird der Merker auf "1" gesetzt. Im nächsten Zyklus sind dann Eingang UND Merker auf "1", also ist der Ausgang der Flankenerkennung wieder "0".
Um die Funktionsweise der Flankenerkennung gut zu sehen, sollten Sie die Simulationsgeschwindigkeit mit der Schildkröte verringern.




```
U  E  3.4
FP M  8.0
=  A  1.6
```



Projekt 1: 3 Förderbänder
Schwerpunkte: Verknüpfung mit UND, ODER; Selbsthaltung.

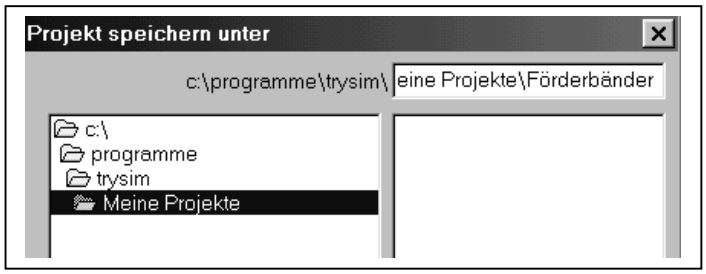
Das folgende Projekt soll Sie schrittweise an die Lösung einer praktischen Aufgabe heranführen. Dafür ist bereits ein Simulationsmodell erstellt worden. Hier soll es also nicht um den Entwurf einer Modellanlage gehen, sondern um die Programmierung der erforderlichen Verknüpfungen.

Aufgabe: Pakete sollen über eine größere Distanz befördert werden. (Man könnte sich auch Schüttgut vorstellen; dieses ist für die Simulation aber nicht geeignet).

Theoretische Erörterung der Aufgabe:
Rein praktische Gründe sprechen dafür, die Gesamtstrecke in 3 Teilstrecken zu unterteilen: Aus der Sicht des Elektrikers würde 1 großer Drehstrommotor beim Volllast-Start das Versorgungsnetz erheblich stärker belasten, als 3 kleinere Motoren, die auch noch einzeln nacheinander eingeschaltet werden können. Im Normalfall könnte man vielleicht dafür sorgen, das Band leer anzufahren, aber was passiert nach einem Not-Aus-Fall? Dann können die Pakete (das Schüttgut) nicht von Hand entsorgt werden. Also: die Pakete sollen über 3 Teilbänder transportiert werden. Dabei muss der Projekteur die Aufgabe so umfassend verstehen und Verknüpfungen vorsehen, dass unter keinen Umständen durch falsche Bedienung am Steuerpult unerwünschte Zustände in der Anlage entstehen können:
Die Bänder müssen so verriegelt sein, dass kein Schüttgut auf ein stehendes Band gefördert wird. Also muss das letzte Band zuerst anlaufen, dann folgt das mittlere Band und zum Schluss das erste. Um den erneuten Sanftanlauf (Leerlauf) zu ermöglichen, muss beim Ausschalten dafür gesorgt werden, dass die Motoren in umgekehrter Reihenfolge stoppen: zuerst das vordere Band, dann das mittlere und zum Schluss das hintere Band. Im Störungsfall (Überlast oder Not-Aus) soll die gesamte Anlage sofort stehen.
Um aus dieser Aufgabe ein SPS-Projekt mit Grundverknüpfungen aus UND- und ODER-Gattern zu machen, wird von einem Steuerpult ausgegangen, in dem für jedes Band ein separater Ein- und Austaster vorhanden ist. Eine nicht unterwiesene Person könnte zwar beliebig auf den Taste(r)n "klimpern", die Anlage dürfte aber trotzdem nur in der vorstehend beschriebenen Reihenfolge reagieren.

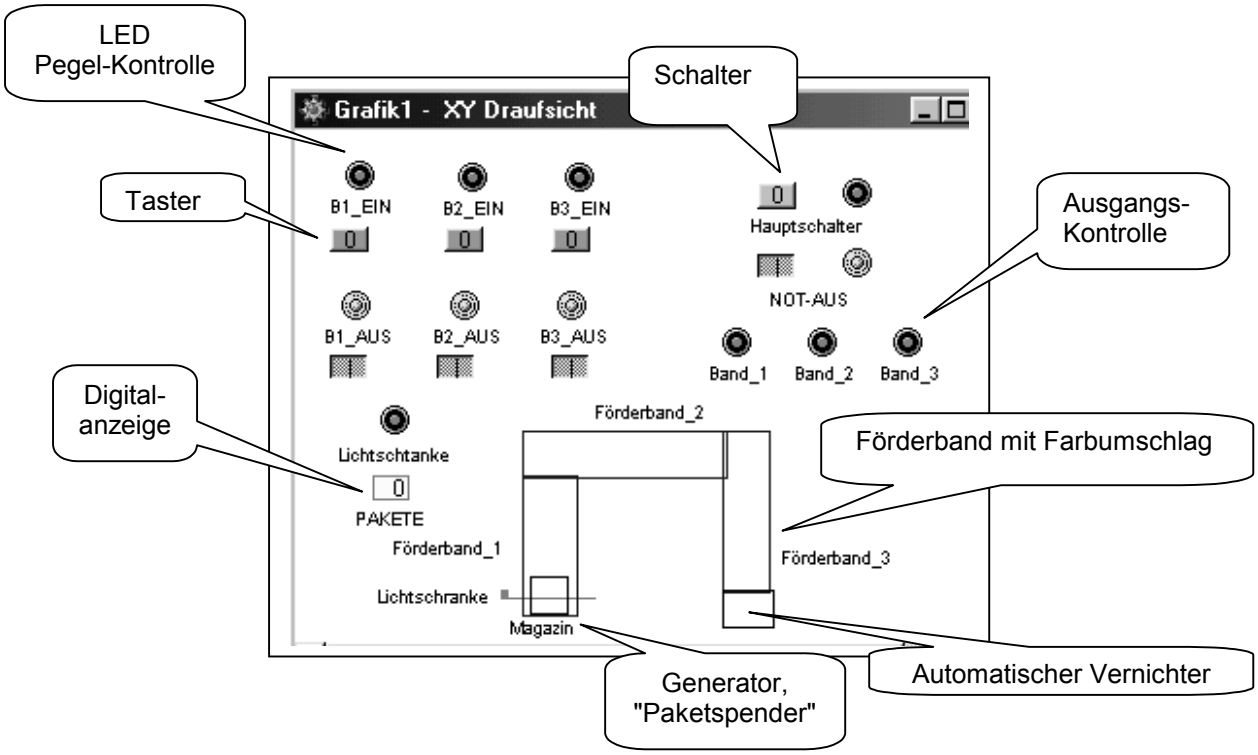
Zur Praxis mit TrySim:
Starten Sie bitte das Programm TrySim.
Öffnen Sie für die Projektierung im Ordner **<Vorlagen>** das Projekt **<F_Band_V>** durch Doppelklick auf die gleichnamige Zeile im Auswahlfenster. (Haben Sie auch das richtige Verzeichnis gewählt?).

Im rechten Fenster erscheint grau eine Reihe von Dateien, um die Sie sich nicht kümmern müssen. Quittieren Sie statt dessen mit **<OK>**.
Um das jetzt geöffnete Projekt als weitere Vorlage unbenutzt zu erhalten, sollten Sie hiervon eine Kopie mit anderem Namen anlegen:
Wählen Sie bitte unter **<Projekt>** den Punkt **<Speichern unter...>.**



Jetzt haben Sie ein "eigenes Projekt".

In aller Regel erscheint zumindest das oben erwähnte "Steuerpult" und die Bandanlage. Falls nicht, öffnen Sie bitte das Menü **<Grafik>** und wählen dort **<XY-Draufsicht>**. Sie können das Fenster zwar mit der bekannten WINDOWS-Technik vergrößern, aber der Inhalt wird dadurch nicht im Maßstab angepasst. Besser geht es, wenn Sie aus der Icon-Leiste eine der Lupen anklicken. Dann vergrößert / verkleinert sich die Anlage maßstäblich.

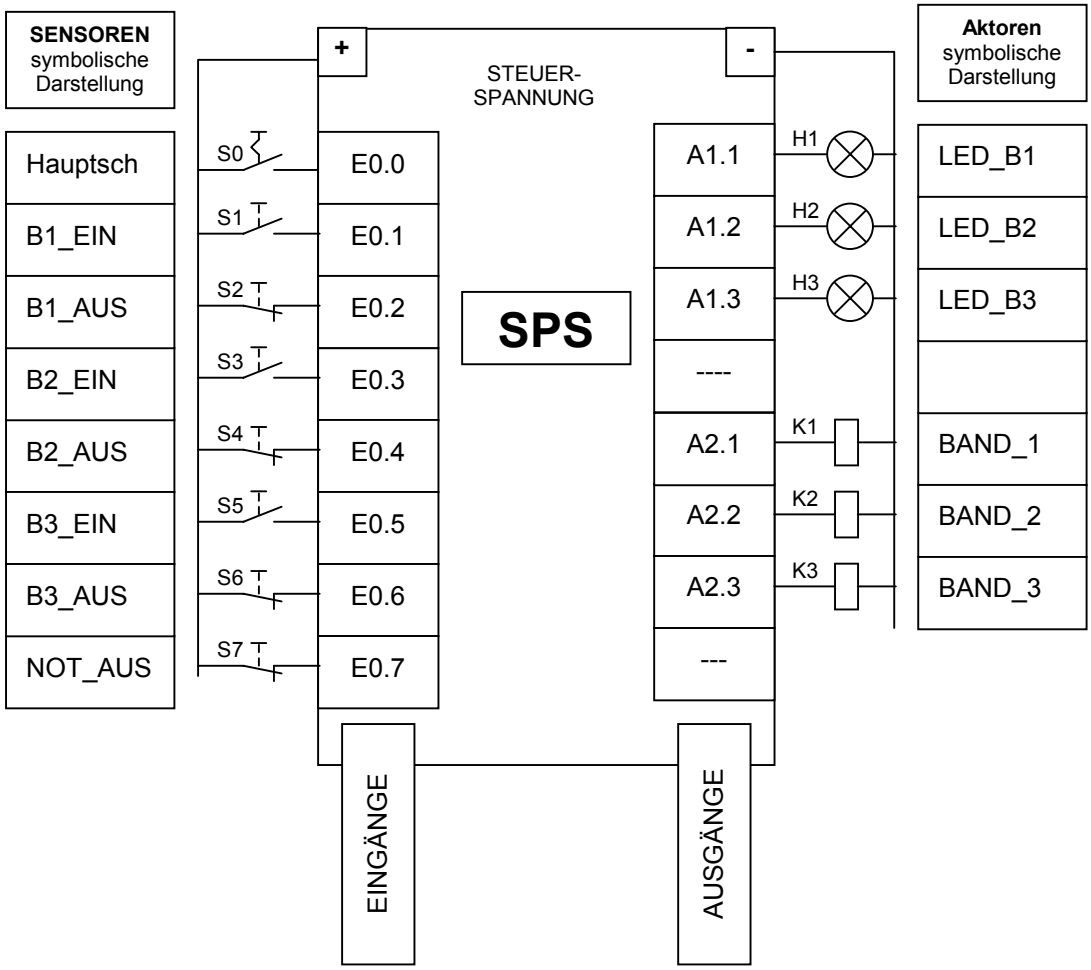


Sie sehen die Ein- und Austaster, den Not-Aus-Taster und einen Hauptschalter (Schließer), der stellvertretend für die Überstromauslösung der Motoren steht. Dieser Überstromauslöser muss für den Betriebsfall von Hand ein- und im Störungsfall ausgeschaltet werden.
Die Bandanlage ist in der Draufsicht dargestellt. In der 3D-Ansicht unter <Grafik> sehen Sie, dass das "Magazin", aus die Pakete entnommen werden, über dem Band 1 angeordnet ist. Von dort fallen die Pakete auf das Band. Am Ende von Band 3 ist ein sogenannter Vernichter angebracht, der die Pakete wieder verschwinden lässt, damit es keinen Stau gibt. Diesen Vernichter müssen Sie nicht programmieren, d.h. nicht über einen Operanden ansprechen. Der Kontakt des Paketes mit dem Vernichter reicht zum Löschen aus.
Unter <SPS>|<Symboltabelle> können Sie die folgende Tabelle öffnen. Sie wurde schon vorbereitet, damit Sie 1) mit einer fertig verdrahteten Anlage arbeiten können und 2) die Funktionen der Operanden im editierten Programm direkt verfolgen können.

Symbolische Adresse	Operand	Datentyp	Kommentar
HS_LED	A 0.0	BOOL	Hauptschalter-Pegel
LED_B1EIN	A 0.1	BOOL	Pegel Taster B1_EIN
LED_B1AUS	A 0.2	BOOL	Pegel Taster B1_AUS
LED_B2EIN	A 0.3	BOOL	Pegel Taster B2_EIN
LED_B2AUS	A 0.4	BOOL	Pegel Taster B2_AUS
LED_B3EIN	A 0.5	BOOL	Pegel Taster B3_EIN
LED_B3AUS	A 0.6	BOOL	Pegel Taster B3_AUS
LEDNOTAUS	A 0.7	BOOL	Pegel NOT-AUS-Taster
LED_B1	A 1.1	BOOL	LED BAND 1
LED_B2	A 1.2	BOOL	LED BAND 2
LED_B3	A 1.3	BOOL	LED BAND 3
Band_1	A 2.1	BOOL	Motor Förderband 1
Band_2	A 2.2	BOOL	Motor Förderband 2
Band_3	A 2.3	BOOL	Motor Förderband 3
PAKET	A 2.5	BOOL	Paketgenerator
Hauptsch	E 0.0	BOOL	Hauptschalter
B1_EIN	E 0.1	BOOL	Band 1 Ein (Schließer)
B1_AUS	E 0.2	BOOL	Band 1 Aus (Öffner)
B2_EIN	E 0.3	BOOL	Band 2 Ein (S)
B2_AUS	E 0.4	BOOL	Band 2 AUS (Ö)
B3_EIN	E 0.5	BOOL	Band 3 Ein (S)
B3_AUS	E 0.6	BOOL	Band 3 AUS (Ö)
NOT-AUS	E 0.7	BOOL	Taster NOT-AUS (Ö)

Anschlussplan

Der Netzanschluss und die externe Sicherheitsabschaltung der Ausgänge wurde wegen der Übersichtlichkeit nicht dargestellt. Hier soll es nur um das Verständnis für die relevanten Ein- und Ausgangsbeschaltungen gehen, die in der realen Anlage für das Programm wichtig sind.

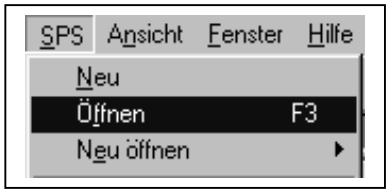


Die zusätzlich im Steuerpult und in der Symboltabelle aufgeführten Anschlüsse A 0.0 bis A 0.7 werden nur für die Simulation programmiert und "angeschlossen". Sie sollen nur verdeutlichen, welche Spannungspegel an den Tastern im betätigten und unbetätigten Zustand anliegen.

Der Ausgang A 2.5 kommt in der Praxis auch nicht vor. In der Simulation muss allerdings von irgendwo her der Befehl kommen, dass ein Paket angeliefert werden soll. Der Ausgang A 2.5 ist für den Paketgenerator gewählt worden, d.h., wenn dort eine logische "1" anliegt, wird ein Paket erzeugt.

Die den Tastern zugeordneten Kontroll-Lampen (oder LEDs) sollen nicht den Betriebszustand der Bänder anzeigen, sondern die Spannungspegel der Eingangsklemmen der SPS, an denen die Taster angeschlossen sind. Dadurch soll verdeutlicht werden, welche Spannungspegel abgefragt bzw. programmiert werden sollen. Die LEDs der <EIN>-Taster (Schließer) leuchten, wenn die Taster betätigt werden, die der AUS-Taster (Öffner) im unbetätigten Ruhezustand.

Sollte kein weiteres Teilfenster zu sehen sein (oder wenn es plötzlich verschwunden zu sein scheint), können Sie die gewünschten Fenster über Befehle der Menüleiste öffnen.
Im Augenblick interessiert natürlich das Fenster, in dem Sie programmieren wollen. Dazu klicken Sie bitte in der Menüzeile auf **<SPS>** und wählen dort **<Öffnen>**. Sie finden dort immer den OB1. Hier steht das Hauptprogramm. Dies ist der Baustein, der jedes Mal aufgerufen wird, wenn die Simulation der Maschine abgeschlossen ist. Wenn der OB 1 bearbeitet worden ist, wird wieder die Simulation der Maschine gestartet. Üblicherweise werden im OB 1 hauptsächlich Funktionen und Funktionsbausteine aufgerufen, Sie können aber im OB 1 auch ganz normal programmieren. Kleine Programme, die keine Strukturierung benötigen, werden vollständig im OB 1 programmiert, der dann der einzige Baustein des Programms ist. (Vertiefen Sie dieses Thema in TrySim unter <Hilfe>|<Index> "OB").



Klicken Sie bitte auf **<OB1>** im Fenster (OB1 sollte dann blau unterlegt sein) und bestätigen mit **<OK>**. Es wird Ihnen das 1. Netzwerk angezeigt.


START

Warum leuchten die LEDs z.T. schon, wenn Sie das Start-Icon betätigen? Es wurden schon einige Netzwerke für die Pegelanzeige angelegt. Ein Netzwerk besteht immer aus der Abfrage mindestens einer Adresse (eines Operanden) und der Zuordnung zu einer anderen Adresse.

In einem Netzwerk kann immer nur eine Ausgangsadresse zugewiesen werden, es sein denn, es liegen mehrere Ausgänge direkt parallel.

Dazu später.

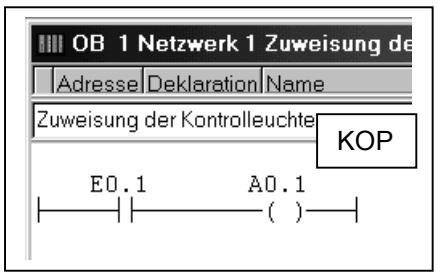
In der Projektvorlage startet dieses Netzwerk in der Programmiersprache KOP. Sofern Sie irgendwo in dieses Programmierfenster geklickt haben (und der **Fensterkopf blau** ist [**= aktives Fenster**, auf das sich alle Befehle beziehen sollen]), sind in der Icon-Zeile die Icons für die Sprachenwahl (AWL, FUP und KOP) zu sehen. Sie können hier jetzt ganz einfach auf ein anderes Sprach-Icon drücken und die Übersetzung im Editierfenster beobachten.

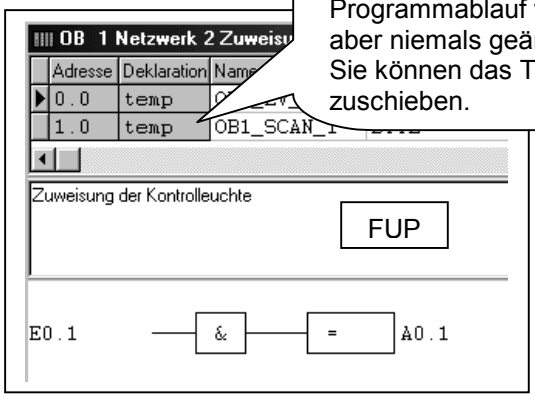
Im Netzwerk 1 wird nur ein Eingang abgefragt. Hier handelt es sich also nicht um eine logische Verknüpfung, sondern nur um eine Zuweisung unter einer bestimmten Bedingung, nämlich wenn am Eingang eine logische "1" anliegt.

Zuweisung der Kontrollleuchte

U E 0.1
= A 0.1

AWL



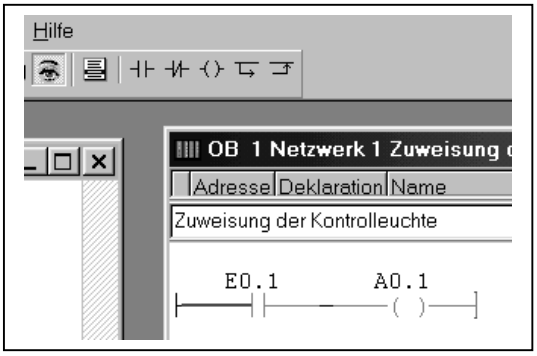


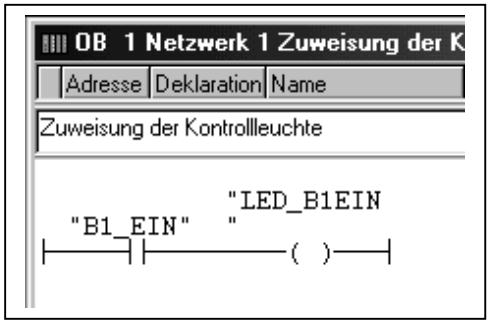
Dieser Bereich enthält nur "Routinedaten", die für den Programmablauf wichtig sind, aber niemals geändert werden. Sie können das Teilfenster zuschieben.

Im FUP fällt auf, dass der Eingang auf ein UND mit nur einem Eingang geführt wird. Das Symbol wurde so gewählt, weil es kein Symbol für nur eine Eingangszuweisung gibt. Ein Eingang kann aber in einer SPS prinzipiell nicht ohne Verknüpfung auf einen Ausgang geführt werden. Deshalb hilft man sich, indem man ein normales UND oder ODER in das Netzwerk legt und einen markierten Eingang mit der <Entf>-Taste löscht.
Dass die "Steuerung" schon arbeitet, lässt sich bereits testen: Wenn das **Editierfenster aktiv** ist (blauer Kopf), können Sie oben **das Auge-Icon anklicken.** Falls vorher die Anlage ausgeschaltet war, werden Sie gefragt, ob Sie jetzt starten wollen. Kicken Sie getrost auf <OK>. Dadurch springt das Start-Icon um und gleichzeitig wird - bei FUP und KOP - im Editierfenster das "1"-Potenzial durch rote Linien dargestellt.


Auge-Icon


START





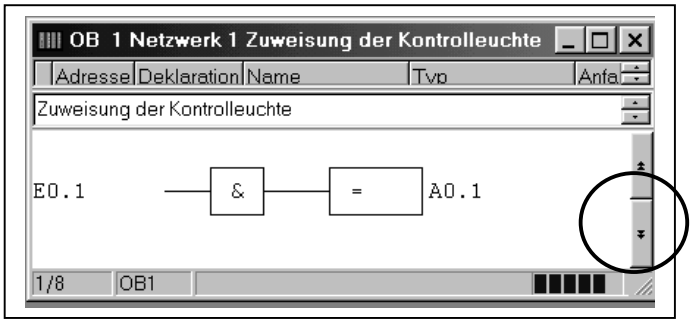
Symbolische Darstellung

In obiger Liste der symbolischen Adressen (Symboltabelle) sind die tatsächlichen Klemmen den Bezeichnungen im Pult / in der Anlage zugeordnet. Diese Klemmenbezeichnungen (Operanden) sind in den Netzwerken wiedergegeben.
Wenn eine entsprechende Symboltabelle erstellt wurde (vgl. TrySim <Hilfe>|<Index>| <Symboltabelle>), können (nur wenn das Editierfenster aktiv ist !) die Operanden unter **<Ansicht>|<Symbolische Darstellung>** auch symbolisch dargestellt werden, so dass die Funktionen der Operanden zu erkennen sind.

Betätigen Sie bitte den Taster <B1_EIN> im Pult (Grafikfenster) mit der linken Maustaste (in Zukunft <LM> abgekürzt) und betrachten Sie dort die darüber liegende LED, bzw. im Editierfenster den roten geschlossenen Linienzug.
Das bedeutet: wenn die Verknüpfungsbedingung erfüllt ist, wird der Ausgang auf "1" gelegt.

In AWL werden die Pegel durch nachstehende "0"- bzw. "1"- Angaben dargestellt (wenn vorher das Auge-Icon aktiviert wurde).

Die nächsten Netzwerke können Sie sich anzeigen lassen, indem Sie auf den unteren Pfeil ganz rechts im Editierfenster klicken. Das letzte Fenster ist immer ein freies Fenster.



Wählen Sie bitte das Icon <FUP>, um die folgenden Schritte vergleichen zu können.

Sie werden feststellen, dass bereits 8 Netzwerke für die Kontrolllampen der Taster eingerichtet wurden. Das letzte Netzwerk ist immer leer. Dort beginnt das neue Programm. Wählen Sie bitte das leere Netzwerk 9 aus.

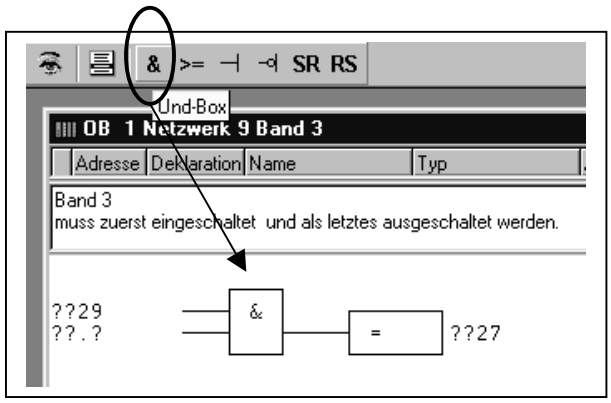
Das letzte Band 3 soll erst einmal allein laufen. Es soll mit einem Impuls von Taster <B3_EIN> eingeschaltet und mit einem Druck auf Taster <B3_AUS> ausgetastet werden. Dazu ist eine programmierte Selbsthaltung nötig. Diese wurde schon in der Einführung behandelt.

Wenn irgend ein Stoffel den Taster <EIN> und <AUS> zur selben Zeit drücken würde, dürfte das Band nicht laufen. Folglich müssen sowohl die Pegel der Eingangsklemme für <EIN> und für <AUS> abgefragt und UND-verknüpft werden.

Zum Programmieren schalten Sie ggf. das Auge-Icon durch einen weiteren Klick aus.

Klicken Sie bitte bei aktivem Editierfenster auf das &-Icon. Dadurch wird folgendes Rumpf-Netzwerk angelegt:

UND

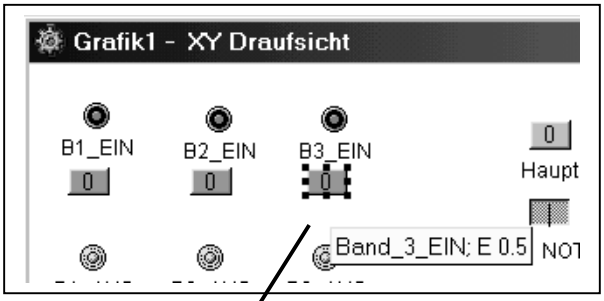


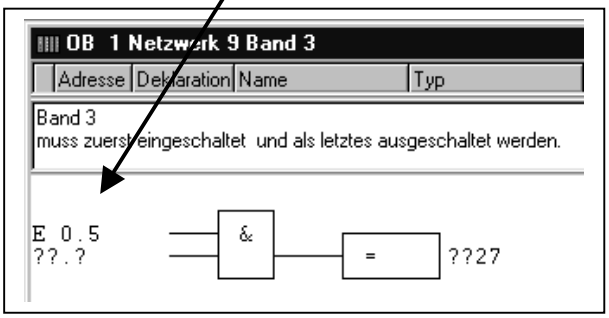
Jetzt kommt der Clou: Wenn Sie die Anlage mit Icon Nr. 8 oder der Taste <F5> ausschalten, können Sie mit <LM> im Steuerpult (Grafikfenster) auf jedes beliebige Element klicken, so dass dieses Element mit einem Rahmen umgeben ist. Anschließend wechseln Sie mit der Maus in das Editierfenster und klicken an die Stelle, an der das ausgewählte Element bzw. dessen Operand abgefragt werden soll.


Nr. 8

Für die Zuweisung der Operanden "per Klick" muss die Anlage ausgeschaltet sein!

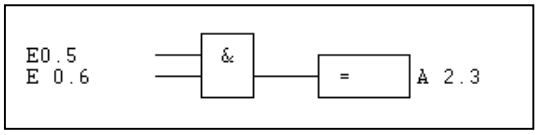

drücken



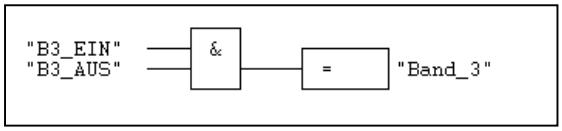


Zuweisung des Operanden

An den zweiten Eingang klicken Sie bitte den Operanden des eckigen Tasters <B3_AUS> (nicht die runde LED). Jetzt muss noch der Ausgang zugeordnet werden. Dieses Klick-Verfahren gilt für Ein- und Ausgänge. Um das gewünschte Förderband zu markieren, müssen Sie den blauen Rand des Bandes an irgendeiner Stelle mit dem Mauszeiger genau treffen (der Pfeil wird zur Hand) und können anschließend im Editierfenster rechts hinter den Zuordnungsblock klicken.

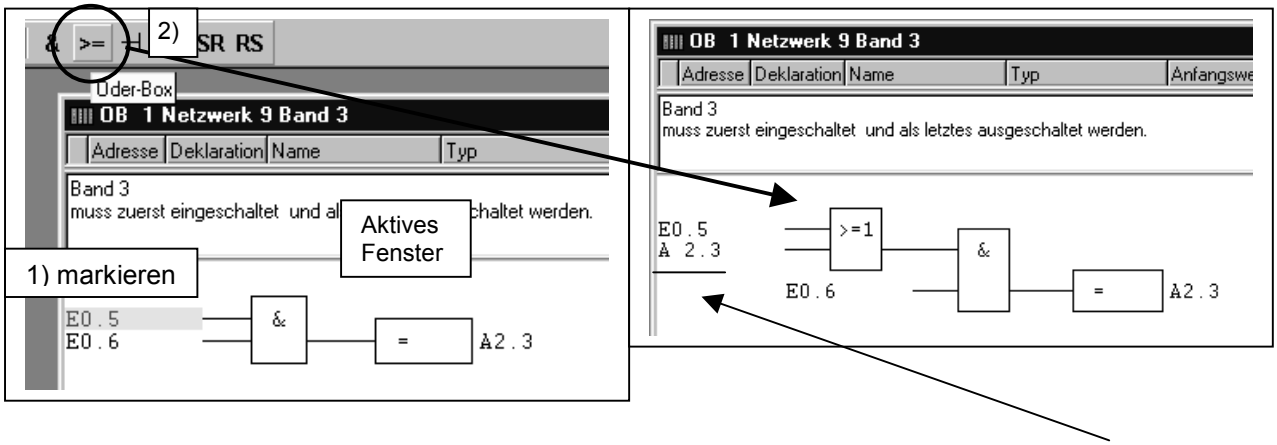


Operanden (Ein- u. Ausgangsklemmen)



symbolische Darstellung

Sie kommen jetzt zum ersten Probelauf:
Schalten Sie bitte das Auge ein, quittieren mit <OK> und drücken im Pult <B3_EIN>. Das letzte Band müsste rot werden, solange Sie den Taster drücken, ebenso das zugehörige Netzwerk. Wenn Sie den Taster wieder loslassen, "steht" das Band wieder. Es fehlt ja noch die Selbsthaltung. Die Abfrage vom "EIN"-Befehl ODER vom eigenen Ausgang (BAND 3 = Klemme A 2.3) tritt an die Stelle der bisherigen "EIN"-Abfrage. Setzen Sie bitte bei ausgeschaltetem Auge (!) den Cursor **vor** das entsprechende Eingangsbein (also auf den Text) und klicken Sie oben auf das ODER-Icon. Sie sehen, wie sich das ODER dazwischenschiebt und der bisherige Eingangsoperand nach vorn wandert.

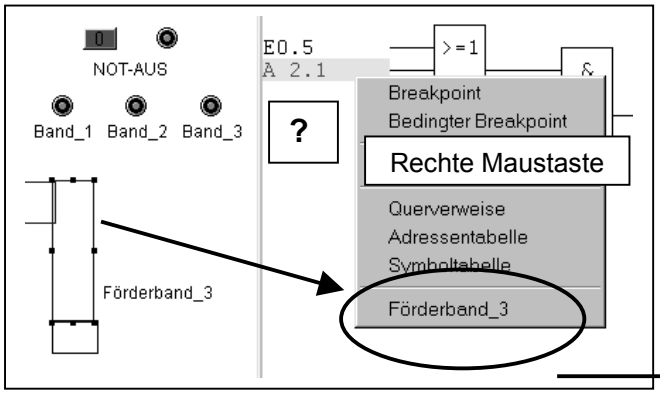


Der zweite ODER-Eingang wird wieder (bei ausgeschalteter Anlage) aus dem Grafikfenster per Klick zugeordnet.
(Wenn Sie den Cursor **auf** das **obere** Eingangsbein setzen, überdeckt der gelbe Balken gleichzeitig das "&"-Symbol. In dieser Lage könnten Sie auf das ODER-Icon klicken und aus dem UND wird ein ODER. Das ist hier aber nicht gewollt. Nur sollten Sie darauf achten, damit sich nicht aus Versehen die Logik ändert).
Es sollte jetzt dort am Eingang "A 2.3" stehen.

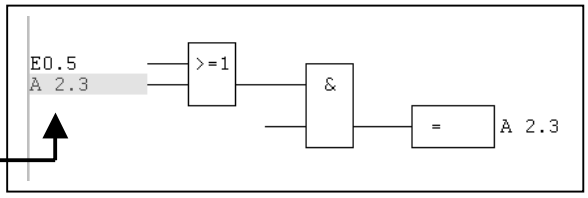
Falls nicht, können Sie

1) bei ausgeschalteter Anlage mit dem Mauszeiger über die Anlage fahren und (ohne Klick) bekommen Sie die Operanden angezeigt.
2) von Hand den Eintrag im Editierfenster ändern.
   Markieren Sie dazu die entsprechende **Textstelle**, quittieren mit **<Enter>** und überschreiben den Operanden. Schließen Sie wieder mit **<Enter>** ab.

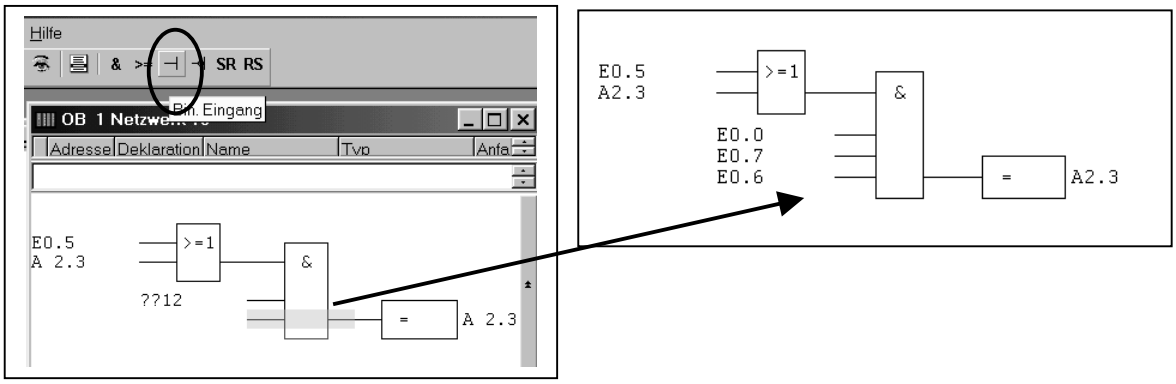
Falls Sie einen schon vorhandenen Operanden ändern wollen, können Sie diesen leider nicht - wie oben beschrieben - mit einem Klick zuweisen. Das geht nur beim ersten Mal. Danach geht es aber auch ganz einfach: Nachdem Sie das Bauteil im Grafikfenster markiert haben, wechseln Sie zu dem gewünschten Ein- /Ausgang im Editierfenster und klicken dort mit <rM>. Es öffnet sich ein Kontext-fenster, in dem der letzte Eintrag dem gewünschten Operanden entspricht. Klicken Sie bitte darauf.



Beispiel:
A 2.1 soll geändert werden.
Förderband_3 wurde am Rand angeklickt.
Das Kontextmenü weist
"Förderband_3" (=A 2.3) aus. Mit Klick wird der Eingang überschrieben.



Bei Not-Aus und ausgelöstem Motorschutz muss das Band auch sofort und vorrangig vor der EIN-Funktion stehen. Also müssen diese Eingänge auch noch eingebunden werden. Dazu benötigen wir noch 2 zusätzliche Eingänge am UND. Klicken Sie (bei ausgeschaltetem Auge !) auf das **Icon <Bin. Eingang>** und belegen die entsprechenden Eingänge, bzw. Operanden.



Testen Sie bitte auch diesen Schritt:

- Auge an
- <OK>
- <B3_EIN> tasten
- Band 3 beobachten: funktioniert die Selbsthaltung?
- Mit <B3_AUS> abschalten. Alternativ auch mit Not-Aus abschalten.
- Auge aus (damit sie weiter programmieren können).

Denken Sie daran, dass der "Hauptschalter" eingeschaltet sein muss.

Beherzigen Sie bitte den folgenden Rat: Schreiben Sie grundsätzlich über das Editierfeld einen sinnvollen Kommentar. Am besten, bevor Sie ein Netzwerk programmieren, gewissermaßen als Arbeitsauftrag, damit Sie genau planen, was gerade passieren soll.

Vergessen Sie bitte nicht, auch Teilfortschritte immer wieder zu sichern mit dem Menüpunkt **<Projekt>|<alles speichern>** oder mit Icon Nr. 2.

Nr. 2

Nun sind Sie ja schon gewissermaßen Experte für Selbsthaltungen. Deshalb nutzen Sie bitte Ihre Kenntnisse für zwei weitere, neue Netzwerke: Band 2 und Band 1 können in gleicher Weise völlig unabhängig voneinander programmiert werden. Vergessen Sie nicht den Kommentar! Testen Sie auch diese Netzwerke.

Nachdem Sie sich davon überzeugt haben, dass bisher alles wunschgemäß läuft, kommen wir zu den **Verriegelungen der einzelnen Bänder**.

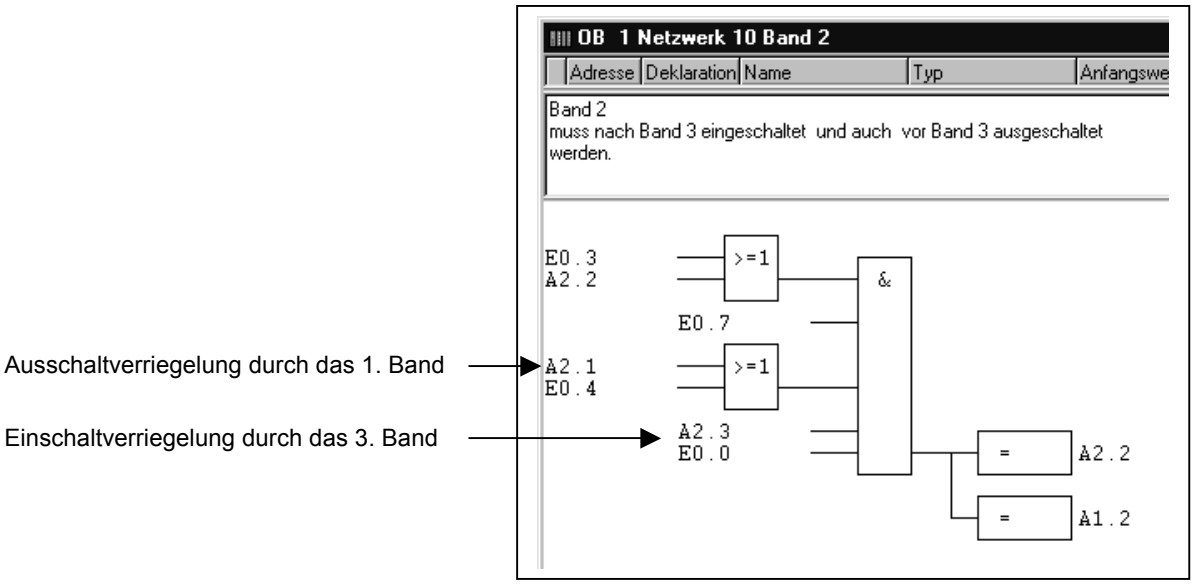
Band 2 darf erst laufen, wenn Band 3 schon läuft. Folglich reicht der Startimpuls von <B2_EIN> allein nicht aus, sondern dieser Impuls darf nur unter einer Bedingung wirksam werden. Eine Bedingung ist logisch immer eine UND-Verknüpfung. Ein weiterer Eingang wird am UND eingefügt und mit Klick wird der Operand von Band 3 (A 2.3) zugewiesen.

Und schon testen Sie wieder:

- Auge an
- <OK>
- <B2_EIN> tasten: passiert nichts? Gut!
- <B3_EIN> tasten
- Band 3 beobachten: funktioniert die Selbsthaltung? Gut!
- <B2_EIN> tasten: Läuft Band 2? Gut!
- Mit <B2_AUS> und <B3_AUS> abschalten. Alternativ auch mit Not-Aus abschalten.
- Auge aus (damit sie weiter programmieren können).

Für das Band 1 gilt sinngemäß die gleiche Verriegelung mit Band 2.

Das Ausschalten muss auch verriegelt werden, damit es keinen Stau gibt. Band 2 darf erst abzuschalten sein, wenn Band 1 schon steht. Also muss der Austaster für Band 2 (Öffner) so lange überbrückt werden, bis Band 1 steht. Eine "Brücke" oder eine parallele Leitung ist logisch immer ein ODER. An den Eingang zum Ausschalten (E 0.4) wird (bei... na, das wissen Sie jetzt bestimmt schon: bei ausgeschaltetem Auge) ein ODER geklickt. An den freien Eingang kommt die Abfrage des Ausgangs von Band 1 (A 2.1). Jetzt sollte die Verriegelung beim Ein- und Ausschalten (in umgekehrter Reihenfolge) funktionieren.

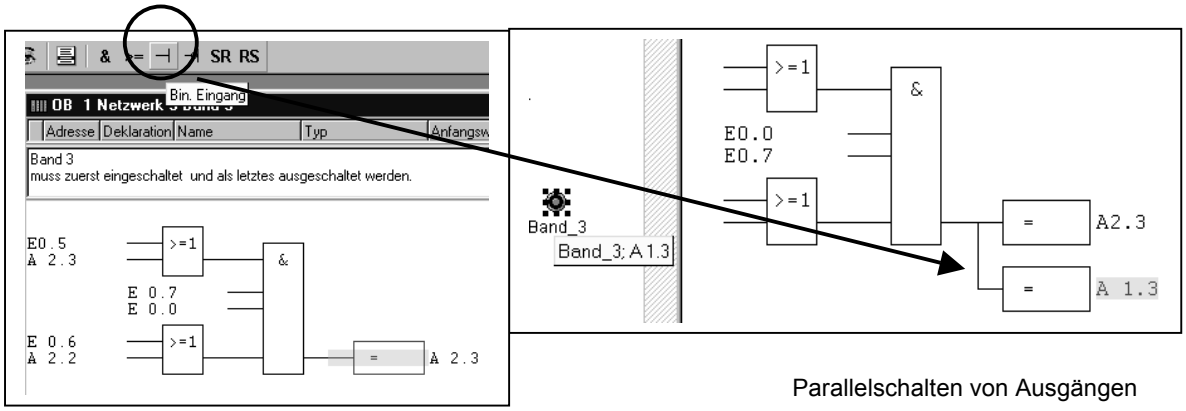


Solange Band 1 läuft, liegt an A 2.1 eine „1“ an. Deshalb wäre es bedeutungslos, wenn der Öffner <B2-AUS> betätigt würde und dann an E 0.4 eine „0“ anläge. Erst nach dem Abschalten von Band 1 wird die Unterbrechung durch <B2_AUS> wirksam.

Wenn dieser Teil funktioniert, sollten Sie die sinngemäße Verknüpfung zwischen dem dritten und dem zweiten Band vornehmen.

Jetzt sollte die gesamte Anlage wunschgemäß verriegelt sein. Testen Sie bitte alle Varianten am Pult und verfolgen Sie die Spannungspegel bei eingeschaltetem Auge im Editierfenster.

Es bleiben noch die Kontroll-Lampen für die Bandantriebe im Pult übrig. Sie können einfach parallel zu den entsprechenden Ausgängen gelegt werden. Das "=" im Ausgangsblock wird markiert und anschließend Icon Nr.24 (zusätzlicher Ein- / Ausgang) angeklickt.



Parallelschalten von Ausgängen

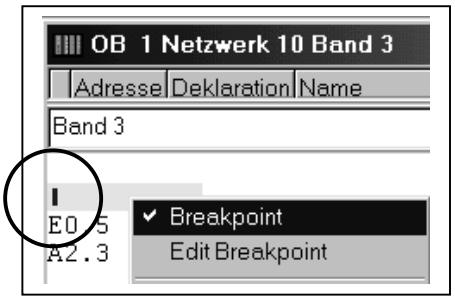
Wenn Sie den Cursor auf den Rand des "Magazins" halten, finden Sie den Eintrag in der Symboltabelle bestätigt: der zugeordnete Operand ist A 2.5. Das ist die Adresse des "Generators", der von TrySim bereitgestellt wird. Die Funktion dieses Bausteins ist nicht zu programmieren, sondern Sie können ihn einfach zur Erzeugung neuer Pakete benutzen, wenn an seiner Adresse eine logische "1" anliegt. Diese Funktion können Sie jetzt einbauen, da die Bänder bereits logisch richtig laufen, (denn sonst würden sich ja die Pakete stapeln, oder das gedachte Schüttgut würde einen großen Haufen schmeißen).
Wenn das Förderband 1 läuft, kann gleichzeitig der TrySim-Generator (A 2.5) als Magazin wirken und die Pakete spenden. Im Netzwerk für Band 1 wird A 2.5 ebenfalls parallel zum Ausgang gelegt.

Es wird Ihnen sicherlich auffallen, dass beim Ausschalten die Pakete immer auf dem ersten Band liegen bleiben. Wir werden später eine Lösung diskutieren.

Nützliche Tipps:

- Sollte es wider Erwarten doch einen Paketstau geben, können Sie ganz einfach "aufräumen": Wählen Sie **<Anlage>|<Dynamiks>|<Normale löschen>.**

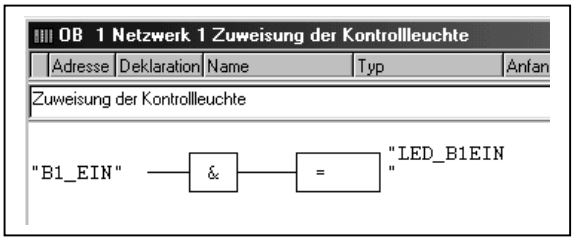
- Sollte sich die Anlage "wie eingefroren" verhalten, könnte es sein, dass Sie aus Versehen mit der rechten Maustaste das Kontextfenster geöffnet haben und dort die obere Zeile <Breakpoint> betätigt haben. Das erkennt man am **Häkchen vor <Breakpoint>** oder an dem kleinen blauen Strich oben im Netzwerk. Entfernen Sie dann das Häkchen durch einen weiteren Klick. Sinn des Breakpoints ist es, das Programm an gewollter Stelle für Testzwecke anzuhalten.



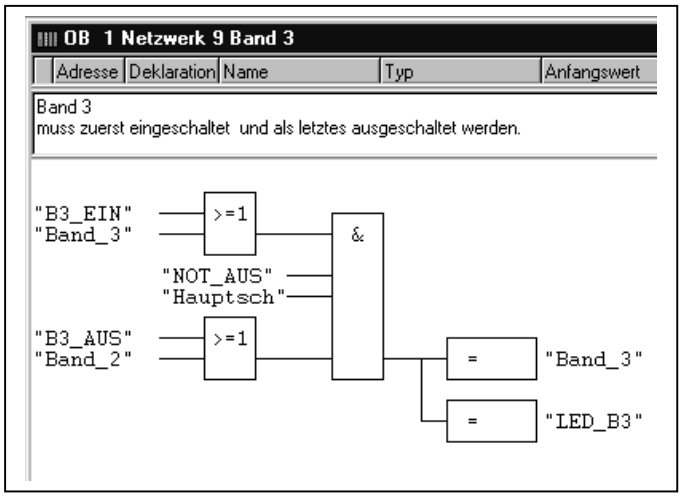
Breakpoint setzen / entfernen

Zusammenstellung der vollständigen Netzwerke mit symbolischen Adressen.
vgl. TrySim-Verzeichnis <Lösungen>|<F_Band_1>

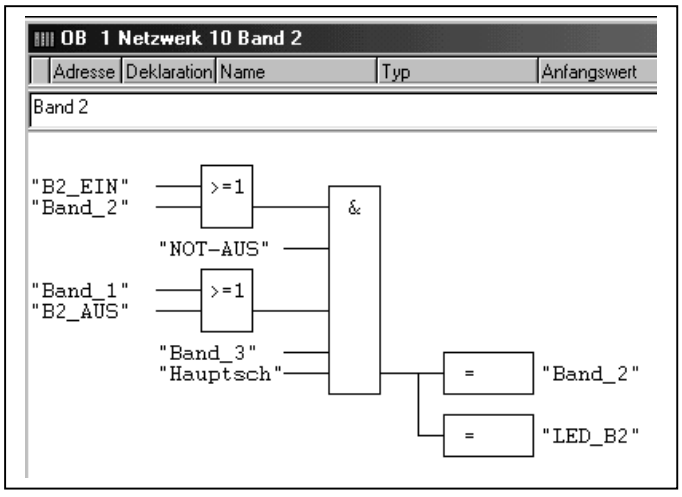
Netzwerk 1: Zuweisung der Kontrollleuchte (exemplarisch für alle LEDs)



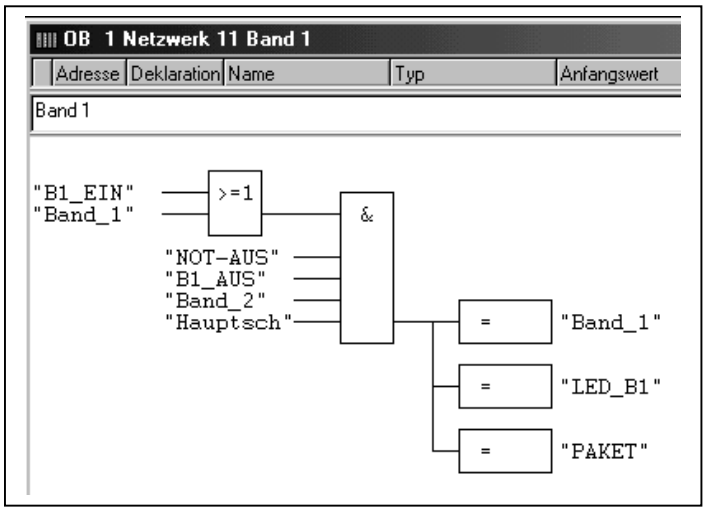
Netzwerk 9: Band 3 muss zuerst eingeschaltet und als letztes ausgeschaltet werden.



Netzwerk 10: Band 2 muss nach Band 3 eingeschaltet und auch vor Band 3 ausgeschaltet werden.



Netzwerk 11:
Band 1 kann erst nach Band 3 und Band 2 eingeschaltet werden und wird als erstes ausgeschaltet.



Sie haben gelernt:

- Auswahl eines bestimmten Projektes
- Erstellen eines neuen Ordners
- Kopieren eines Projektes
- Speichern eines Projektes in einem anderen Ordner
- Öffnen des Grafikfensters in TrySim
- Verändern der Größe des Grafikfensters
- Öffnen der Bausteine (Netzwerke)
- Anlegen neuer Netzwerke
- Arbeiten in verschiedenen Teilfenstern, Wahl des aktiven Fensters
- Umschalten der Anzeige von direkten und symbolischen Operanden
- Verknüpfung von UND- und ODER-Bausteinen
- Wahl der Sprachen AWL, FUP und KOP
- Parallele Ausgänge erstellen
- Zusätzliche Eingänge schaffen
- Zuordnung der Operanden zu Maschinenelementen
- Ändern der Operanden
- Kommentare schreiben
- Test der Anlage / des Programms
- Beobachten der Signalpegel
- "Dynamiks" löschen
- Breakpoint setzen / löschen

Projekt 2: 3 Förderbänder, projektiert mit SR-Flip-Flops

Schwerpunkte: Verknüpfung mit SR-FlipFlops; Selbsthaltung.

Wir knüpfen an die vorstehende, uns schon vertraute Förderband-Aufgabe an. Bedienpult und Maschinenanlage sind gleich geblieben. Das Programm soll allerdings in anderer Weise erstellt werden, nämlich mit Speichergliedern vom Typ <SR>.

Aufgabe: Pakete sollen über eine größere Distanz befördert werden. Die Pakete sollen über 3 Teilbänder transportiert werden. Dabei muss der Projekteur die Aufgabe so umfassend verstehen und Verknüpfungen vorsehen, dass unter keinen Umständen durch falsche Bedienung am Steuerpult unerwünschte Zustände in der Anlage entstehen können: Die Bänder müssen so verriegelt sein, dass kein Schüttgut auf ein stehendes Band gefördert wird. Also muss das letzte Band zuerst anlaufen, dann folgt das mittlere Band und zum Schluss das erste.

Theorie:
Ein SR-FlipFlop kann die Selbsthaltung ersetzen, die im vorstehenden Beispiel aus UND und ODER konstruiert wurde. Wir hatten den Motor eingeschaltet, indem wir kurz auf den Starttaster getippt hatten. Genauso reagiert das SR-FF, wenn kurzfristig, dauernd oder in Intervallen am "S"-Eingang (Setzen) eine "1" anliegt. Der Unterschied zu der bisher konstruierten Selbsthaltung liegt im Ausschaltpegel: Haben wir bisher zum Ausschalten den AUS-Öffner betätigt und damit eine "0" an die entsprechende Eingangsklemme gelegt, wird ein FlipFlop mit einer "1" am "R"-Eingang (Rücksetzen) zurückgesetzt, d.h. am Ausgang liegt dann "0". Wegen der Drahtbruchsicherheit muss aber im Pult weiterhin ein Öffner die Funktion <AUS> übernehmen. (Wir verwenden das bisherige Pult!). Somit liegt in unbetätigter Ruhestellung des Öffners an der Eingangsklemme "1" an, die sofort zu Rücksetzen des FFs führen würde. Aus dem Zwang der Drahtbruchsicherheit heraus muss also die "1" zwar an der SPS ankommen, vor dem SR-FF jedoch invertiert (negiert) werden.

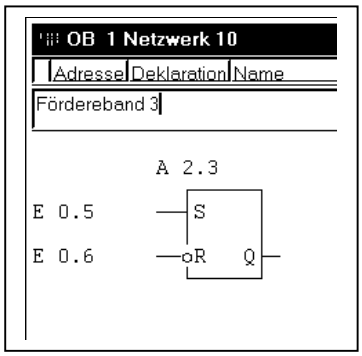
Zur Praxis mit TrySim:

- Starten Sie bitte das Programm TrySim.
- Öffnen Sie für die Projektierung im Ordner **<Vorlagen>** das Projekt **<F_Band_V>** durch Doppelklick auf die gleichnamige Zeile im Auswahlfenster.
- Speichern Sie das Projekt unter einem beliebigen (aber doch sinnvollen) Namen in Ihrem Übungsordner. (Falls erforderlich, sehen Sie sich bitte noch einmal die entsprechende Anleitung im Projekt <F_Band> an).
- Wählen Sie das letzte - freie - Netzwerk aus. Klicken Sie das **<FUP>**-Icon an.
- Klicken Sie auf das Icon  (Bitte aufpassen: Nicht mit RS verwechseln)

Sie werden feststellen, dass über dem Baustein auch <??>-Zeichen stehen. Derartigen Bausteine müssen **"instanziiert"** werden, d.h. einem Operanden zugewiesen werden. Das kann ein Merker sein (z.B. M 2.3) oder direkt ein Ausgangsoperand (z.B. A 2.3), (aber natürlich kein Eingangsoperand). Wenn der **Baustein direkt als Ausgang gekennzeichnet** wird, kann der dahinterliegende Ausgangsblock nach dem Markieren mit der <Entf>-Taste gelöscht (und mit dem Icon  auch wieder gesetzt) werden.

- Belegen Sie die Eingänge mit den entsprechenden Taster-Operanden.
- Markieren Sie den "R"-Eingang und klicken Sie auf das Icon  (Negation).
- Testen Sie das "Rumpfprogramm".
- Beobachten Sie die Signalpegel im Editierfenster.

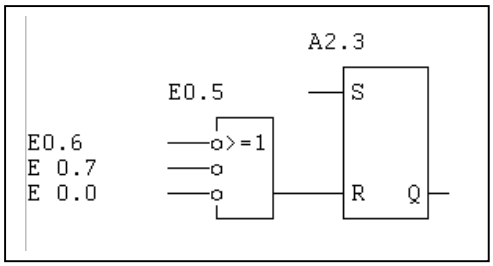
Hinweis: eine falsch gesetzte Negation wird durch einen weiteren Klick an gleicher Stelle wieder gelöscht.



Netzwerk 10

- Verfahren Sie für die Bänder 2 und 1 in neuen Netzwerken (!) ebenso.

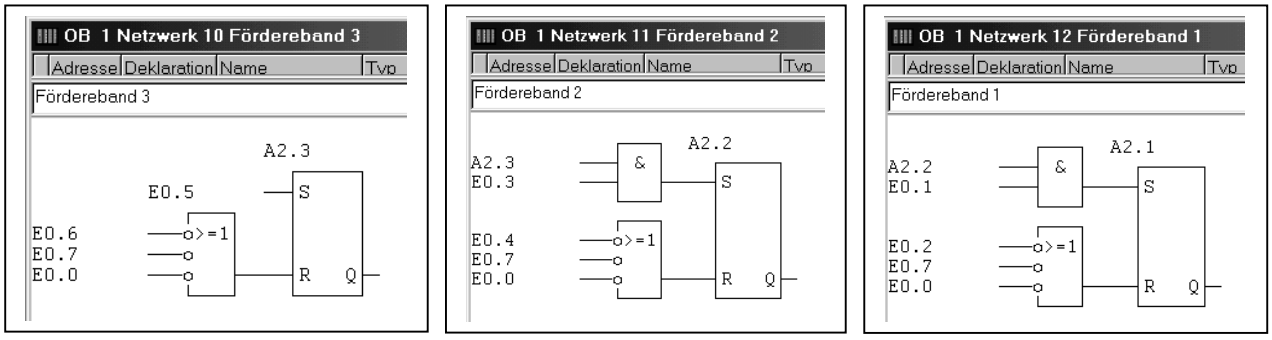
Nachdem Sie die einzelnen, unabhängig laufenden Netzwerke getestet haben, müssen in alle Antriebs-Netzwerke wieder die Not-Aus- und Überstrom- /(Hauptschalter-)-Abfragen integriert werden. Sie erinnern sich: In Betriebsstellung liegen an all diesen Klemmen "1"-Signale an. Wird ein Taster mit der Ausschaltfunktion betätigt, liegt an der entsprechenden Klemme (und nur dort!) eine "0" an. Durch die Negation aller vor dem "R"-Eingang liegenden Signale erhalten wir die logisch richtige Zuordnung. Da jeder Taster mit der Funktion <AUS> unabhängig von den anderen AUS-Tastern das FlipFlop zurücksetzen muss, sind alle Abfragen durch ein ODER zu verknüpfen.



Erweitertes Netzwerk 10

Testen Sie bitte diese Änderungen. **Vergessen Sie nicht, den Hauptschalter einzuschalten.**

Nun kommen wir zur Verriegelung der Bänder untereinander. Band 2 darf erst einzuschalten sein, wenn Band 3 schon läuft. Folglich ist die "EIN"-Taster-Abfrage mit dem Ausgang von Band 3 UND-verknüpft. Das gleiche gilt sinngemäß für Band 1.

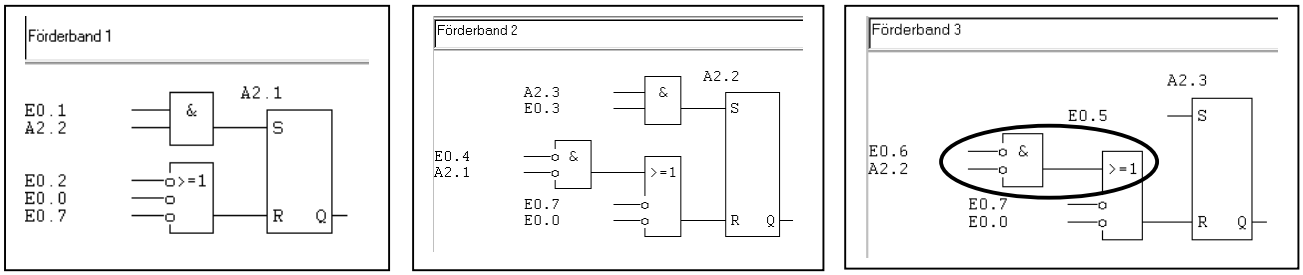


Einschaltverriegelung

Testen Sie bitte diese Änderungen.

Übrig bleibt noch die Verriegelung beim Ausschalten: Band 1 darf ohne Bedingung ausgeschaltet werden. Band 2 ist mit Band 1 zu verriegeln. In der Anlage aus UND- und ODER-Verknüpfungen wurde das Ausschalten über ein ODER verriegelt: E 0.4 konnte erst dann unterbrechen, wenn A 2.1 keine "1" mehr lieferte. Bei dieser <SR>-Version wird zum Ausschalten an "R" eine "1" benötigt. Deshalb könnte man die Ausschaltbedingung auch so verstehen: Wenn Band 1 ausgeschaltet ist (A 2.1 "0") UND "B2_AUS" betätigt wird (E 0.4 liefert auch "0"), liegt an "R" die "1". Sinngemäß sind Band 3 und Band 2 zu verriegeln.

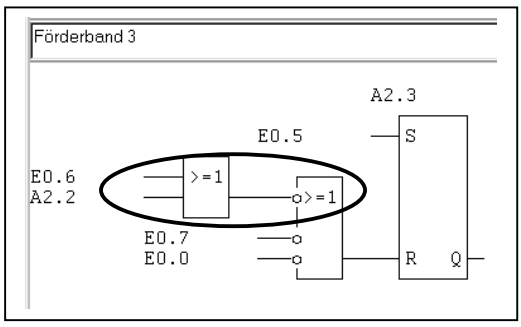
Damit wäre die Programmierung beendet.



Ausschaltverriegelung (vollständige Programmierung)

Vertiefung:

Es kommt in der Praxis jedoch auch darauf an, fertige, lauffähige Programme zu verstehen. Deshalb bauen wir noch eine kleine Schaltungsvariante in das Netzwerk für Band 3 ein.



Nach de Morgan (siehe Fachbücher der Elektrotechnik) kann eine UND-Verknüpfung grundsätzlich auch durch eine ODER-Verknüpfung ersetzt werden (und umgekehrt).

Für die zuletzt beschriebene **UND**-Verknüpfung des <AUS>-Tasters mit dem entsprechenden Band (die an den **EINGÄNGEN negiert** wurde) kann auch eine **ODER**-Verknüpfung **mit negiertem AUSGANG** programmiert werden. Ändern Sie bitte die Schaltung in einem Netzwerk, damit Sie die Richtigkeit der Behauptung überprüfen können.

Um die gesamte Funktion des SR-FF zu untersuchen, müssen wir für Testzwecke die <EIN>- und <AUS>-**Taster in Schließer umbauen.** Dazu markieren Sie bitte bei ausgeschalteter Anlage einen Taster mit der linken Maustaste und öffnen anschließen mit der rechten Maustaste das Editierfenster für die Anlagen-Elemente. Dort finden Sie im Fenster <Typ> die Möglichkeit, die Funktion des Bedienelements zu ändern. Wählen Sie bitte <Schalter> und bestätigen mit <OK>. Dann können Sie den Fall simulieren, dass beide "Taster" betätigt werden. Denken Sie bitte daran: Der <EIN>-Taster wird geschlossen und liefert eine "1", der <AUS>-Taster wird geöffnet und liefert eine "0".
In dieser Stellung darf die Anlage trotz "EIN"-Befehl nicht laufen. Das erreichen wir durch das **SR**-FF.

Deutlich wird die Reihenfolge der Abfrage, wenn das Netzwerk im <AWL>-Modus läuft. Dort erkennen wir, dass die Befehlszeile für das Rücksetzen nach der Setzzeile kommt. (Eselsbrücke: R kommt nach S in SR).
Es gilt grundsätzlich:

Wenn sich mehrere Befehle auf den gleichen Ausgangs-Operanden beziehen und sich diese widersprechen, wird der letzte Befehl ausgeführt. Das gilt auch für ganze Netzwerke. Daraus folgt, dass es für jeden Ausgangsoperanden nur ein Netzwerk geben darf. Alle Verknüpfungen, die sich - auch in verschiedenen Programmphasen - auf den selben Ausgang beziehen, müssen in diesem Netzwerk zusammengefasst werden.

Wie Sie noch sehen werden, gibt es lediglich bei der SR-Programmierung eine Ausnahme, nämlich dann, wenn in einem Netzwerk ein Ausgang gesetzt wird und im folgenden zurückgesetzt wird.

Auf der Icon-Leiste ist auch ein RS-FF abgebildet. Wie man schon aus der Reihenfolge RS erkennen kann, ist in AWL der letzte Befehl die "SETZ"-Zuweisung.

Achtung: Passen Sie gut auf, dass Sie diese beiden Typen nicht verwechseln. Abgesehen davon, dass durch ein Austauschen durch Markieren und Klick auf das jeweils andere Symbol die Ein- und Ausgänge vertauscht werden, bekämen Sie dieses FF nicht aus, falls Sie irgendwo an der Maschine verzweifelt den Not-Aus-Taster drücken und woanders ein naiver "Scherzbold" oder unwissender Kollege auf <EIN> drückt! Dieser Fehler wird erst im Schadens- oder Testfall wahrgenommen!!

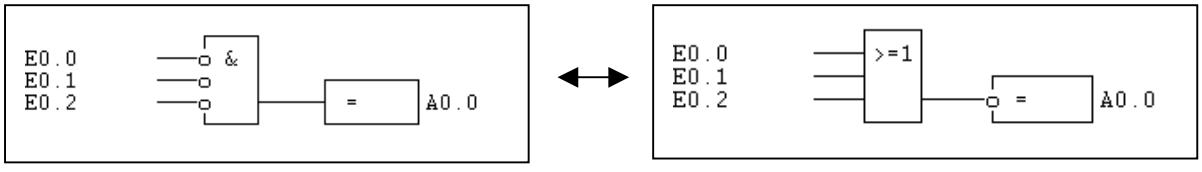
Wozu benötigt man dann diesen "gefährlichen" Baustein?
Zum Beispiel in Sicherheitskreisen, wenn verhindert werden muss, dass jemand einen Alarm abschaltet, weil die laute Hupe nervt. Erst wenn die Ursache der Störungsmeldung beseitig ist, liegt keine "1" mehr am Setzeingang an. Erst dann wird der Befehl am "R"-Eingang wirksam.

Sie können die fertige Version auch unter Verzeichnis <Lösungen>|<F_Band_SR> vergleichen.

Sie haben gelernt:

Flipflops werden zum Speichern von Zuständen verwendet. Wenn der S-Eingang eine „1“ führt, wird der Ausgang eingeschaltet, wenn der R-Eingang eine „1“ führt wird der Ausgang wieder ausgeschaltet. Wenn beide Eingänge eine „1“ führen, dominiert der untere Eingang.

- Unterschied von SR und RS
- Sicherheitsbetrachtungen
- Instanziierung
- Wirksame Signalzustände an S und R
- Rücksetzen mit "1"
- Invertierung der Rücksetzpegel in "AUS"-Funktionen
- SR in AWL
- Jeder Ausgangsoperand darf nur in einem Netzwerk zugewiesen werden (Ausnahme: „S“- oder „R“-Zuweisungen in getrennten Blöcken. Dieser Fall wird später erörtert).
- Anwendung der de Morgan´schen Regeln, z. B.



Projekt 3: Verknüpfungsübungen

Schwerpunkte: Umwandlung von verknüpfungsprogrammierten Steuerungen in SPS-Programme. Reihenfolge der Anweisungen, Abfrage von Schließern und Öffnern.

Häufig steht man vor der Aufgabe, eine funktionierende Kontaktverdrahtung durch ein SPS-Programm zu ersetzten. Dabei wird man sich vom Stromlaufplan leiten lassen und diesen "übersetzen". Dafür eignet sich besonders der Kontaktplan KOP, weil man dabei den Stromlaufplan nur um 90 º drehen muss, um die "Verdrahtung" zu übernehmen. In AWL und FUP ist besonders auf die Reihenfolge der Verknüpfungen zu achten.

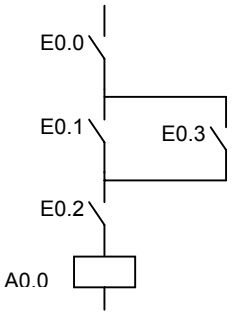
Aufgabe: Die folgende Seite zeigt eine Zusammenstellung von 10 Verknüpfungsaufgaben. Diese sollen durch SPS-Programme realisiert werden. Dazu sollten Sie zuerst - gedanklich oder besser auf Papier - den jeweiligen Anschlussplan erstellen, d.h. die Taster sind einzeln auf Klemmen zu legen und die Verriegelung entsteht durch das Programm.

Damit die Übung nicht zu trocken ausfällt und Sie auch eine Rückmeldung über die Richtigkeit Ihres Programms erhalten, werden Sie die Programme sicherlich am PC erstellen wollen.
Dazu ist eine "Anlage" bereits vorbereitet. Wählen Sie im Verzeichnis **<Vorlagen>** die Datei **<Übungen_1a_V>** bzw. **<Übungen_1b_V>** mit **Doppelklick** und **<OK>** aus.
Speichern Sie die Vorlagen in Ihrem separaten Programmordner unter einem geeigneten Namen, damit Ihnen die Vorlage „wie neu“ erhalten bleibt.
Erstellen Sie dort die Programme in der Sprache Ihrer Wahl und testen Sie die Verknüpfungen.

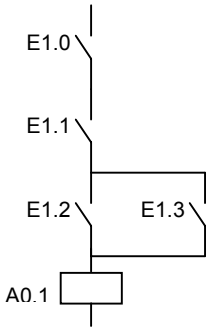
Auf den weiteren Seiten dieses Projektes finden Sie die kommentierten Lösungen, die Sie sich aber erst ansehen sollten, nachdem Sie selber programmiert haben. Die fertigen Lösungen finden Sie auch als Projekt unter <Lösungen>|<Übungen_1a> bzw. <Übungen_1b>.

Erstellen Sie die entsprechenden Programme in AWL, FUP und KOP

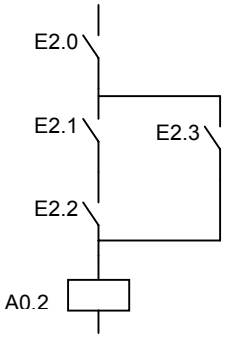
Die Tasterbezeichnungen werden für die SPS-Anschlüsse übernommen.



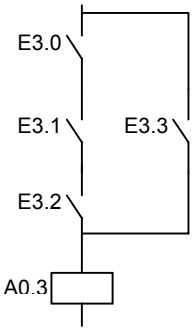
Fall... 1)



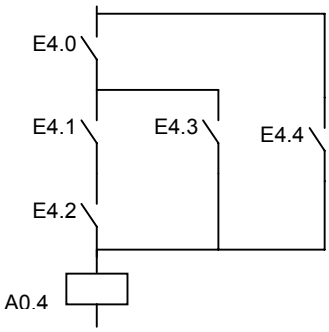
2)



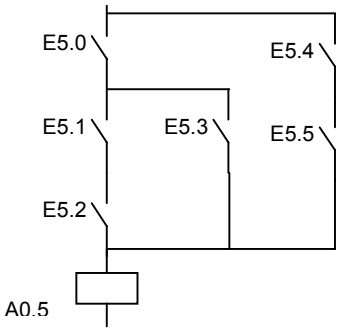
3)



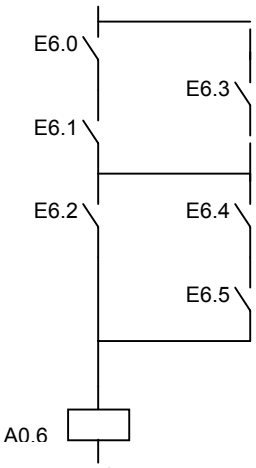
4)



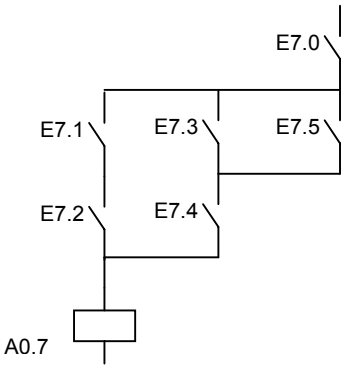
5)



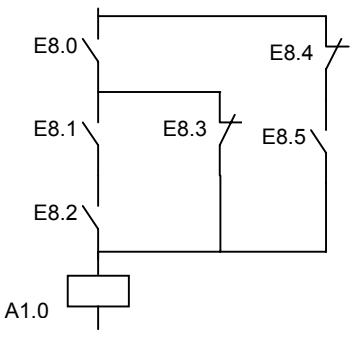
6)



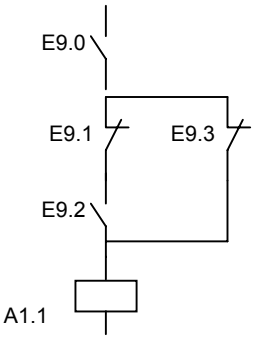
7)



8)



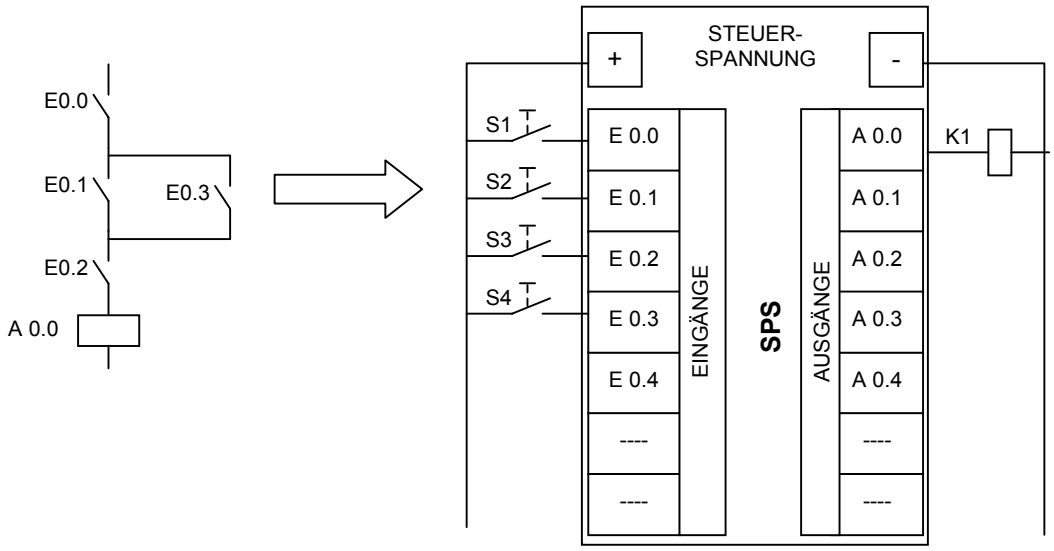
9)



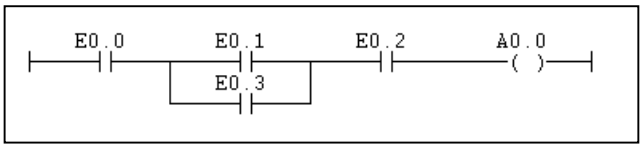
10)

In allen Fällen ist zu fragen: "In welcher Kombination der Schalter ist der Stromkreis geschlossen?" Für diese Situationen ist das Programm zu schreiben. Jeder Schalter wird an eine eigene Eingangsklemme der SPS angeklemmt. Zur besseren Lesbar- und Übertragbarkeit sind die Klemmenbezeichnungen in der verdrahteten Steuerung den Schaltern direkt zugeordnet.
Denken Sie bitte daran, dass das Programm die Spannungspegel "0" bzw. "1" abfragt. Ob und welche Schalter Sie jeweils drücken müssen, ergibt sich aus der Aufgabenstellung, d.h. aus der Schützensteuerung.

FALL 1 **Anschlussplan**

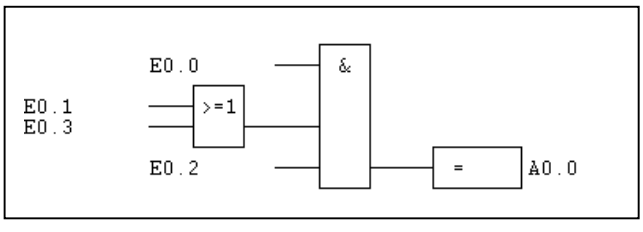


Kontaktplan KOP:



Der KOP lässt sich sehr einfach aus der verdrahtungsprogrammierten Steuerung entwickeln: Die Steuerung wird einfach um 90 ° gedreht. Die Reihenfolge der Verknüpfungen ist identisch mit der Kontaktverdrahtung. Daher stammt auch die Bezeichnung "Kontaktplan". Denken Sie jedoch bitte daran, dass Sie keine Öffner bzw. Schließer programmieren. Die Symbole bedeuten hingegen, ob die SPS an den Klemmen (Operanden) eine logische "1" oder "0" abfragen soll.

Funktionsplan FUP:



Im FUP fühlen sich wohl alle zu Hause, die sich schon mit der Digitaltechnik beschäftigt haben. Hier stellt sich jedoch für Anfänger häufig die Frage, mit welcher Verknüpfung begonnen werden muss. Das ist im Prinzip egal. Man kann die Schaltung in beide Richtungen ergänzen - vom Ausgang nach vorn und umgekehrt.

Variante 1: Vom Anfang (links) zum Ausgang (rechts).
Dieser Weg empfiehlt sich vielleicht für Elektriker, die schon lernen mussten, wie man aus einer Gemischtschaltung den Ersatzwiderstand ermittelt. Dabei denkt man sich alle Kontakte - egal, ob Öffner oder Schließer - durch Widerstände ersetzt. Das geübte Elektrikerauge erkennt dann, dass zuerst der Widerstand der Parallelschaltung (logisches ODER) berechnet werden müsste. Somit beginnt man zuerst (links) auch im FUP mit dem ODER, welches die entsprechenden Schalter bzw. Klemmen abfragt. Der Ersatzwiderstand der Parallelschaltung ersetzt also die Parallelschaltung, so dass nur noch eine Reihenschaltung von 3 Widerständen als nächster Schritt übrig bliebe. Dementsprechend werden diese 3 Eingänge mit einem UND verknüpft.

Variante 2: Vom Ausgang nach vorn (links).
Sie können auch wie im KOP die Schaltung "von oben nach unten" übertragen. Sie erkennen wahrscheinlich, dass der gesamte Aufbau übergeordnet eine Reihenschaltung (also ein logisches UND) ist. Das ist immer dann der Fall, wenn man an nur einer Stelle den Stromfluss unterbrechen könnte, in der Aufgabe z. B. bei E 0.0 bzw. E 0.2.
Die letzte (rechte) Verknüpfung ist also ein UND. Die Parallelschaltung (das logische ODER) wird an einen dieser Eingänge links davor geschaltet. Das Ergebnis beider "Strategien" ist gleich.

Variante 3: "Spaghetti-Programm"
Wie bereits oben erwähnt, ersetzt das SPS-Programm alle Möglichkeiten der verbindungs-programmierten Steuerung (VPS), um den Stromkreis zu schließen. Man könnte also auch ohne Rücksicht auf Programmzeilen oder Übersichtlichkeit wahllos alle Möglichkeiten von geschlossenen Stromwegen ODER-verknüpfen. Wenn man den Überblick behält, mag das wohl technisch funktionieren, lässt aber die Programmlogik bei größeren Verschachtelungen schlecht nachvollziehen und zeugt eher von Unsicherheit. Diese Variante sollten Sie gar nicht erst erwägen, sondern sich um eine saubere Strukturierung bemühen.

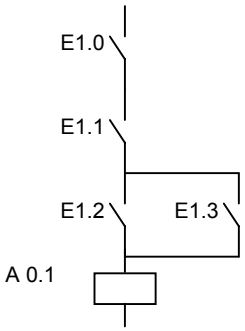
Anweisungsliste AWL:

```
U  E   0.0
U(
O  E   0.1
O  E   0.3
)
U  E   0.2
=  A   0.0
```

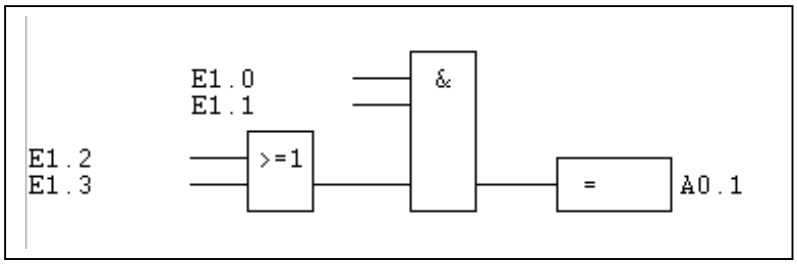
Durch die Klammerbildung U(.....) ist es ebenfalls möglich, "von oben nach unten" die Schützensteuerung zu übertragen. Diese Sprache ist im Programm <Für Anfänger> näher erläutert.

Die nächsten Fälle sind etwas kürzer gefasst und erklären sich - hoffentlich - von selbst.

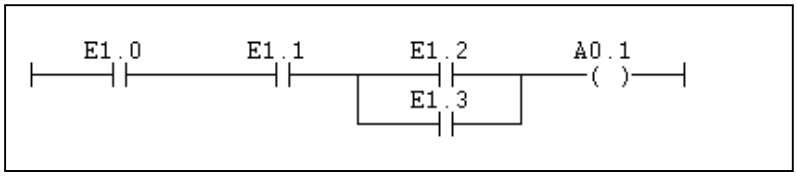
FALL 2



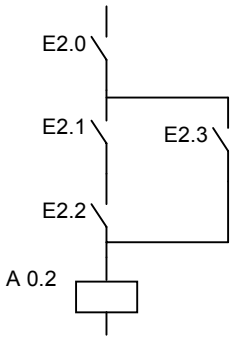
Diese Variante von Fall 1 zeigt, dass nicht die Reihenfolge der Kontakte für das Programm entscheidend ist, sondern die **logische Reihenfolge** (hier: "ODER vor UND").

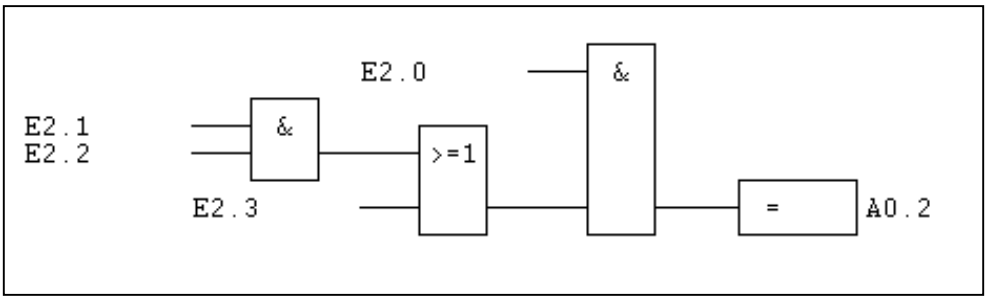



```
U  E   1.0
U  E   1.1
U(
O  E   1.2
O  E   1.3
)
=  A   0.1
```

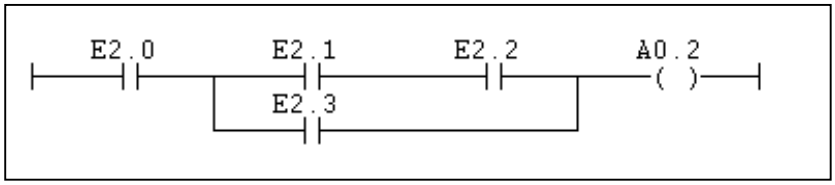


FALL 3

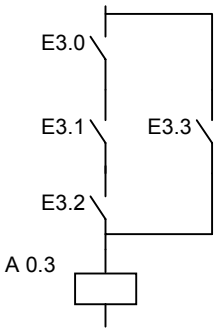


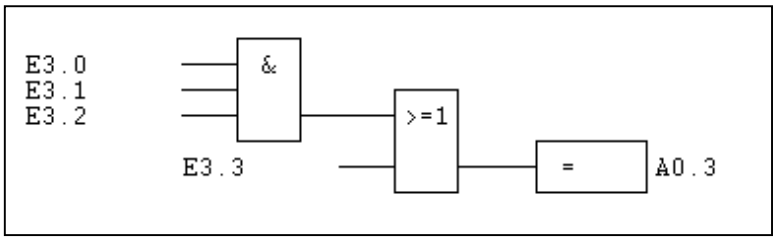


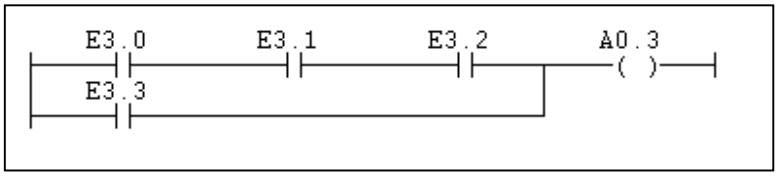

```
U  E   2.0
U(
U  E   2.1
U  E   2.2
O  E   2.3
)
=  A   0.2
```



FALL 4

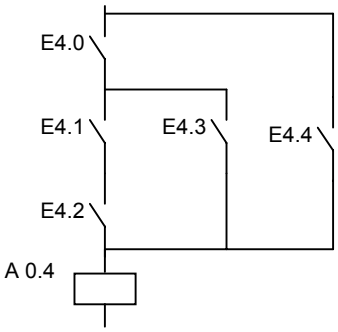


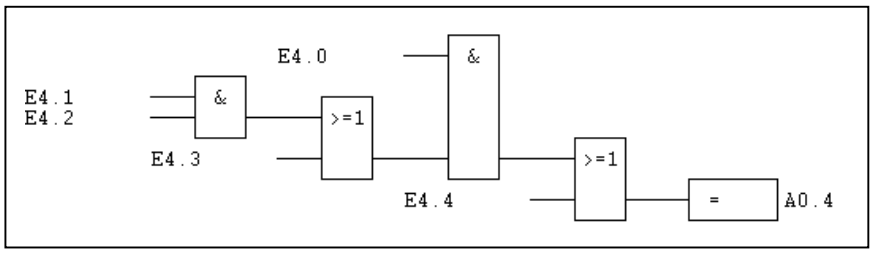




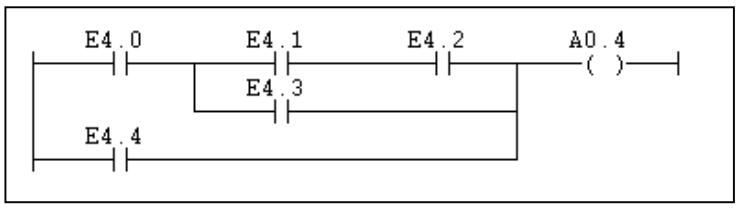

```
U  E   3.0
U  E   3.1
U  E   3.2
O  E   3.3
=  A   0.3
```

FALL 5

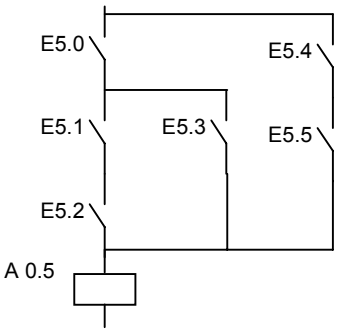


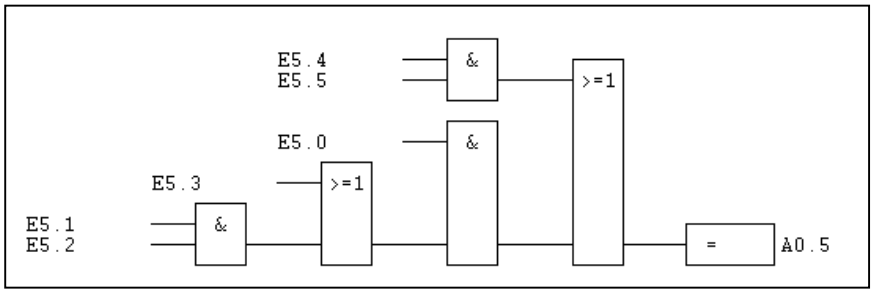



```
U  E   4.0
U(
U  E   4.1
U  E   4.2
O  E   4.3
)
O  E   4.4
=  A   0.4
```

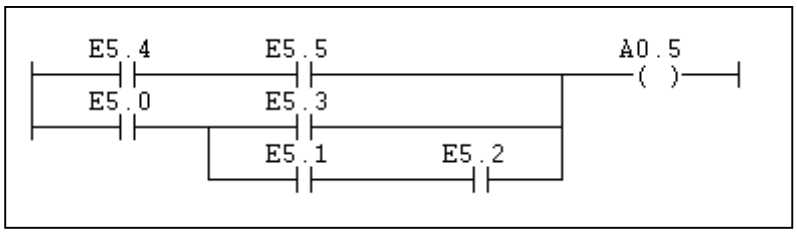


FALL 6

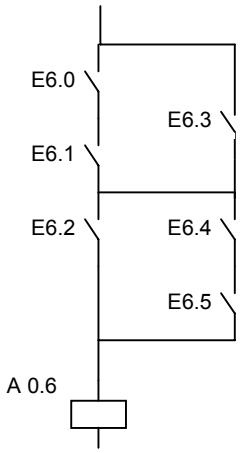


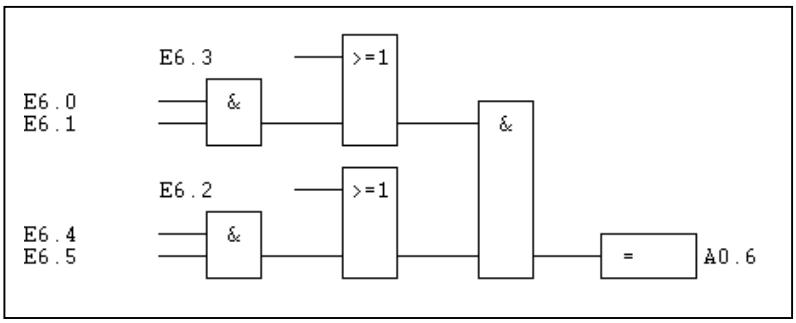


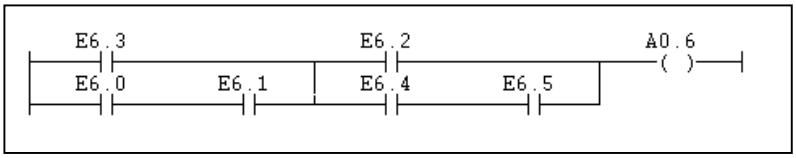

```
U  E   5.4
U  E   5.5
O
U  E   5.0
U(
O  E   5.3
O
U  E   5.1
U  E   5.2
)
=  A   0.5
```



FALL 7



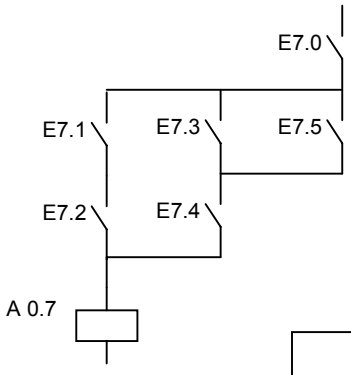


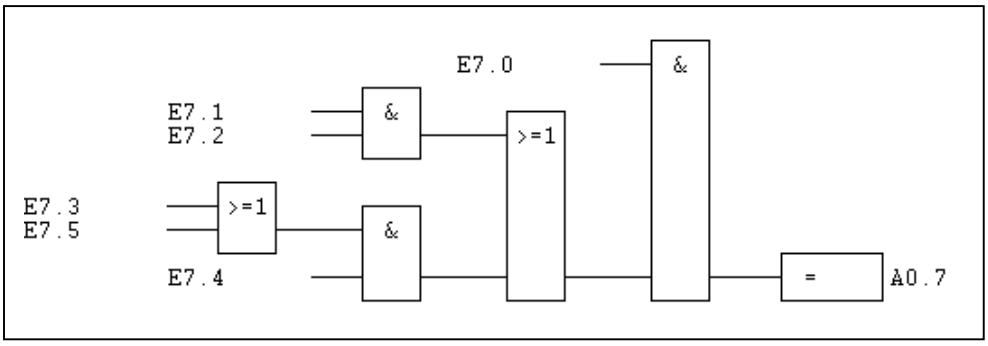



```
U(
O   E    6.3
O
U   E    6.0
U   E    6.1
)
U(
O   E    6.2
O
U   E    6.4
U   E    6.5
)
=   A    0.6
```

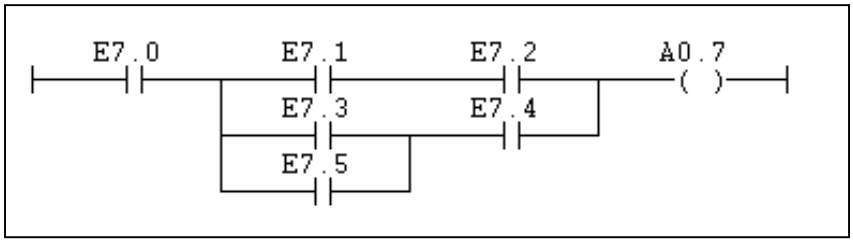
Achten Sie bitte in der AWL auf die **beiden** Klammerungen:
2 ODER-Klammern werden UND-verknüpft.
Für komplexe boolesche Verknüpfung ist diese Sprache unübersichtlich und schlicht ungeeignet.

FALL 8



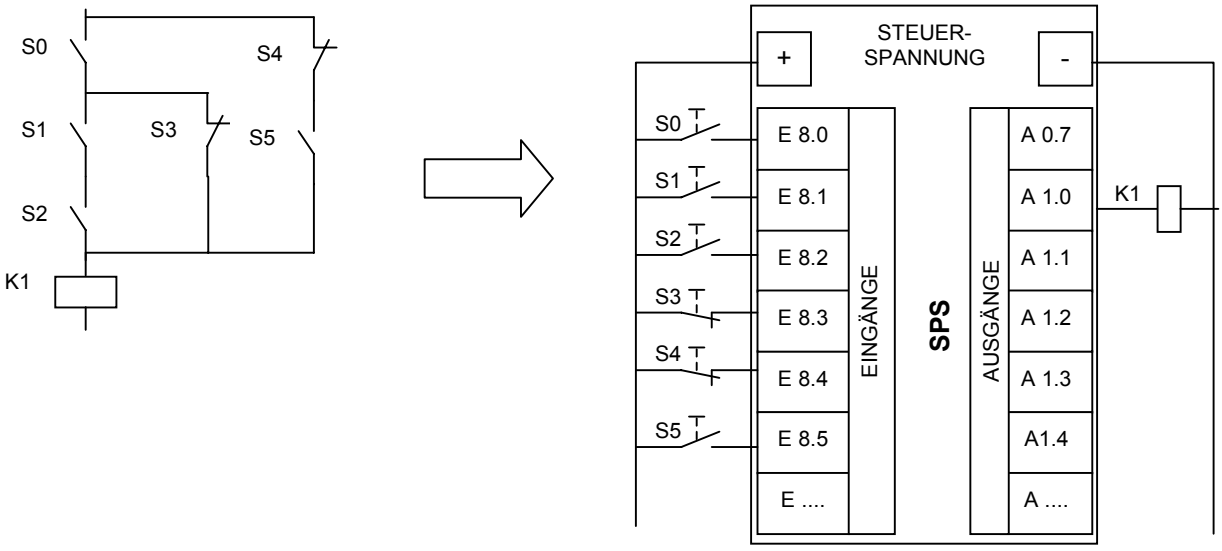


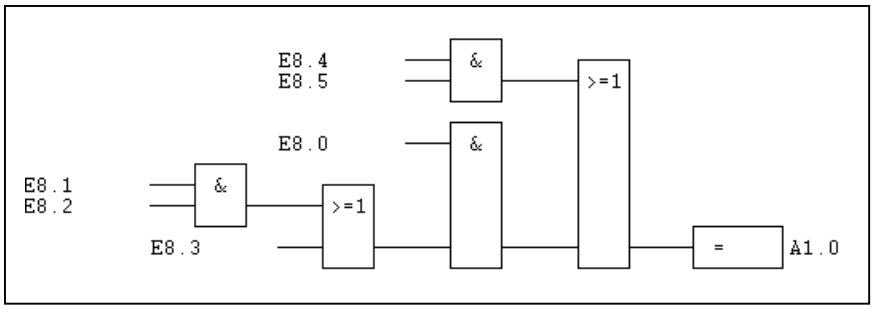
```
U  E   7.0
U(
U  E   7.1
U  E   7.2
O
U(
O  E   7.3
O  E   7.5
)
U  E   7.4
)
=  A   0.7
```



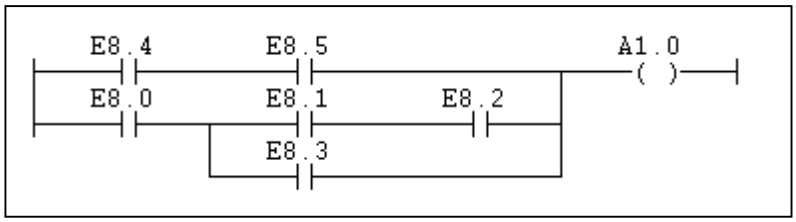
FALL 9

Dieser Fall weicht von den vorstehenden Übungen etwas ab, weil hier auch Öffner vorhanden sind. Denken Sie daran:
Der Schützspule (dem Ausgang) ist es völlig egal, ob der Stromkreis über einen betätigten Schließer oder einen unbetätigten Öffner fließt! Nur für die geschlossenen Stromkreise der Schützensteuerung wird das Programm geschrieben. Der Programmierer muss die Schalter und Taster genau so einplanen, wie sie in der Schützensteuerung bedient würden.
Aus der VPS ist ersichtlich, dass z. B. ein Strompfad geschlossen wäre, wenn S0 betätigt ("1") und S3 nicht betätigt (auch "1"!) wäre. E 8.3 wird also nicht negiert abgefragt! Beide Eingänge müssen UND-verknüpft abgefragt werden. Parallel (ODER-verknüpft) zu S3 liegt noch die Reihenschaltung von S1 und S2 (UND-verknüpft). S4 und S5 schließen einen eigenen Stromkreis, wenn nur S5 betätigt wird. Auch hier liefert S4 im unbetätigten Zustand eine "1", deswegen wird E 8.4 nicht negiert.





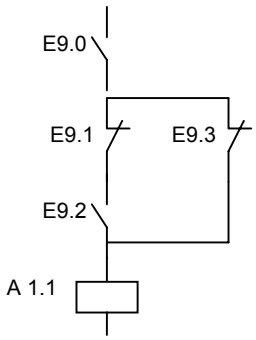
```
U  E   8.4
U  E   8.5
O
U  E   8.0
U(
U  E   8.1
U  E   8.2
O  E   8.3
)
=  A   1.0
```

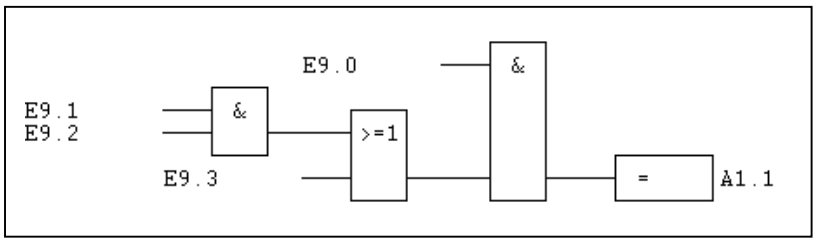


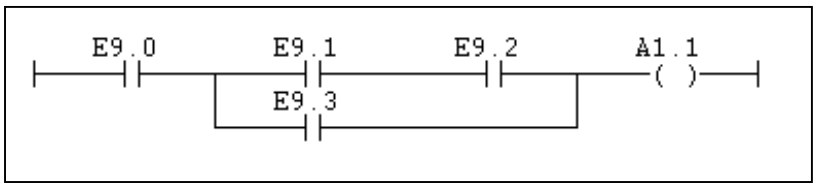
FALL 10

Auch in diesem Fall werden keine Öffner programmiert (dafür gibt es gar kein Symbol! Falls man dennoch behauptet, das "--| / |--" wäre das Öffner-Symbol, so ist das einfach irreführend).

Damit Strom fließen kann, müssten E 9.1 bzw. E 9.3 u.a. nicht betätigt sein. Dann liegt an diesen Klemmen jeweils "1". Dieser Zustand ist abzufragen - also keine Negation!



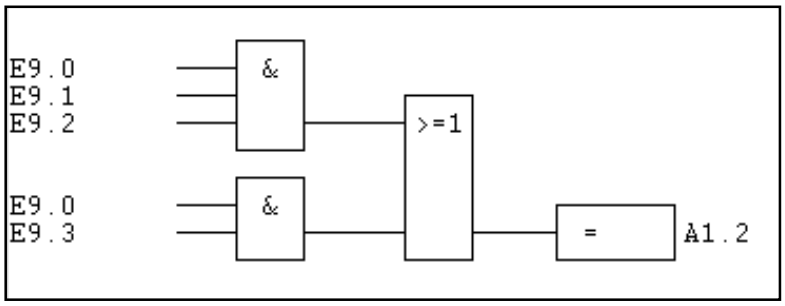


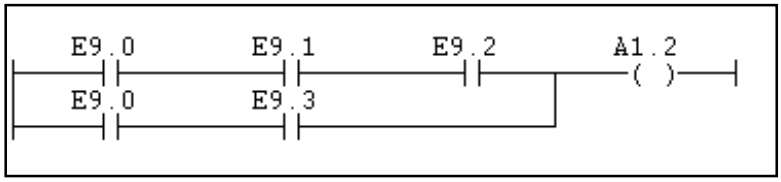



```
U  E   9.0
U(
U  E   9.1
U  E   9.2
O  E   9.3
)
=  A   1.1
```

Exemplarisch für eine andere "Sichtweise" der Schaltung können auch alle Pfade aufgezeigt werden, bei denen ein Stromkreis geschlossen wird, ohne sonstige Verknüpfungen zu berücksichtigen. Bei kleinen Schaltungen ist das vertretbar, bei größeren wird diese Art der Programmierung sehr schnell unübersichtlich, bzw. es besteht die Gefahr, dass nicht alle Pfade erfasst werden.
Es werden also alle möglichen Pfade parallel (als ODER) dargestellt. Für diesen Fall 10 sehe das wie folgt aus:

FALL 10 (Variante)



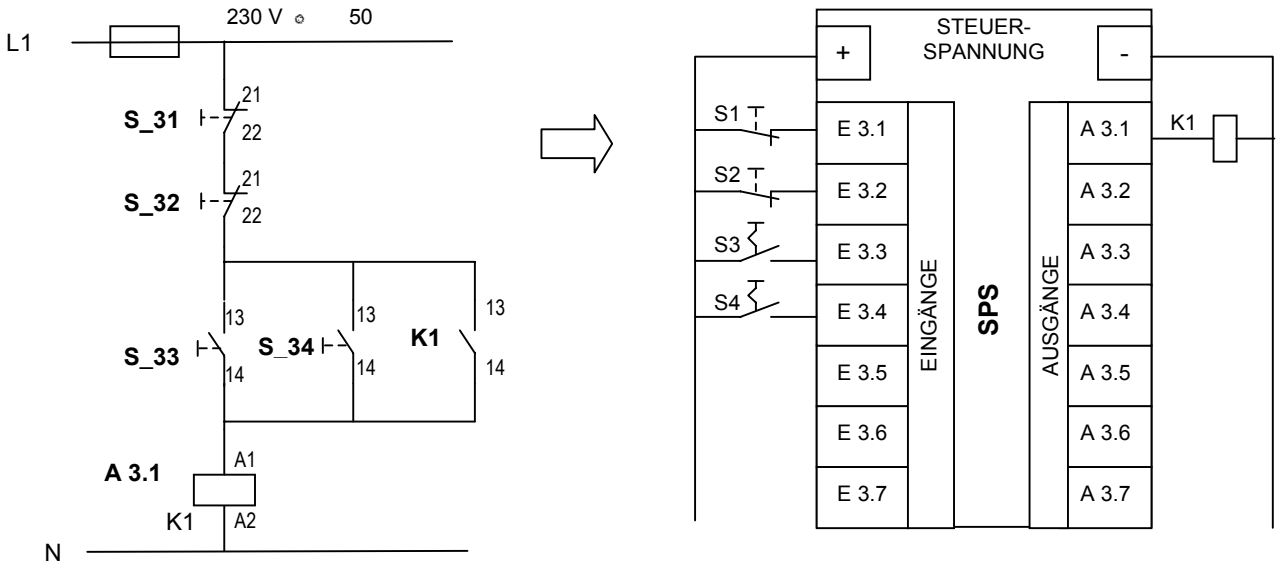



```
U  E  9.0
U  E  9.1
U  E  9.2
O
U  E  9.0
U  E  9.3
=  A  1.2
```

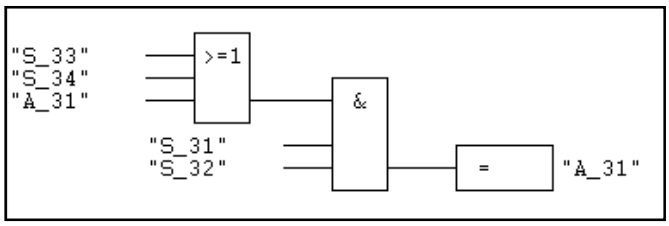
Projekt 3: Übungen Teil 2

Die Vorlage finden Sie unter <Übungen_2_V> im Verzeichnis <Vorlagen>.
Vgl. <Übungen_2> im Verzeichnis <Lösungen>.

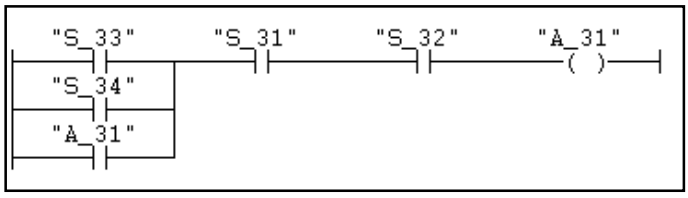
Aufgabe 1)



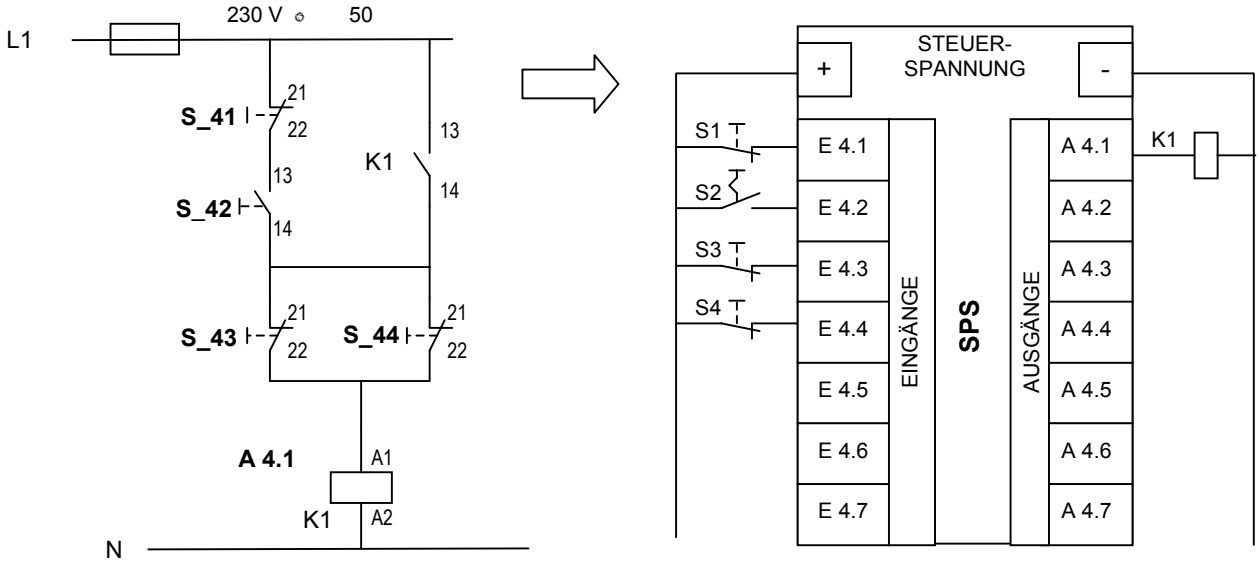
Programm für Aufgabe 1)



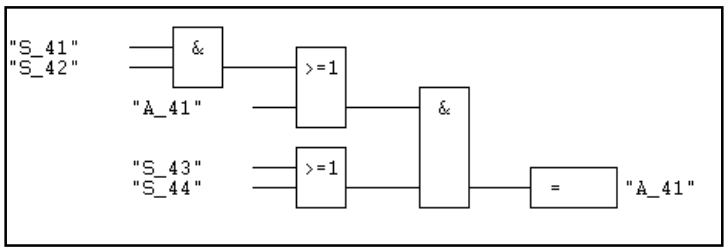

```
U(
O  "S_33"
O  "S_34"
O  "A_31"
)
U  "S_31"
U  "S_32"
=  "A_31"
```



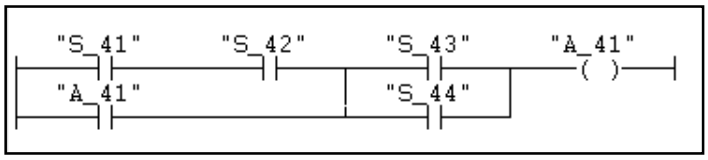
Aufgabe 2)



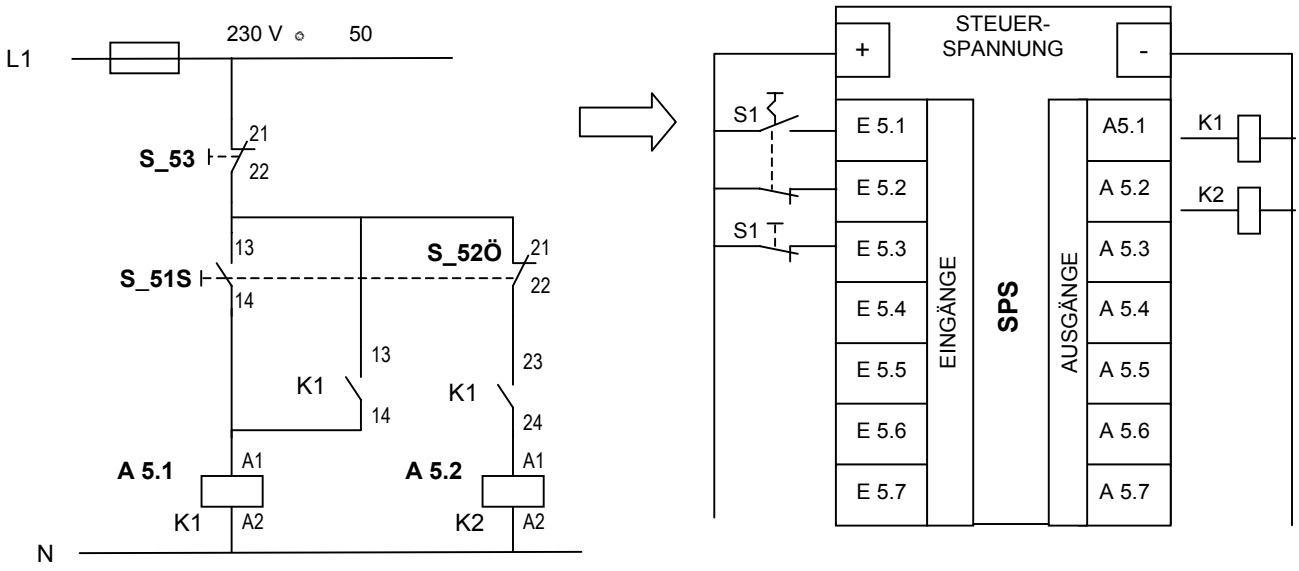
Programm für Aufgabe 2)




```
U(
U   "S_41"
U   "S_42"
O   "A_41"
)
U(
O   "S_43"
O   "S_44"
)
=   "A_41"
```

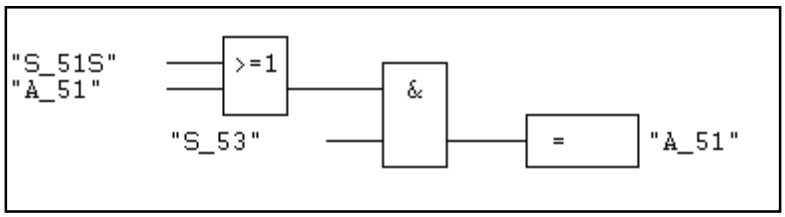


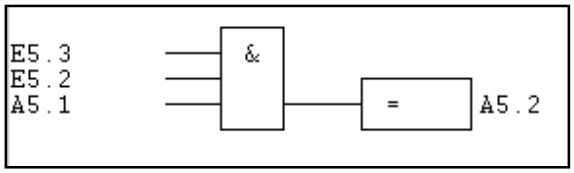
Aufgabe 3)



Programm für Aufgabe 3)

Netzwerk 1 für K1

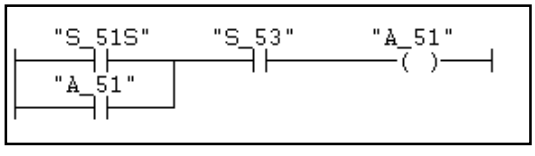


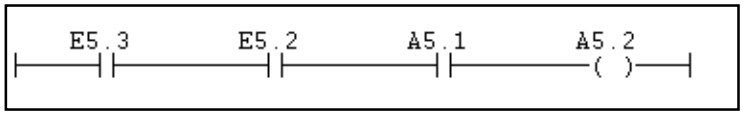


Netzwerk 1

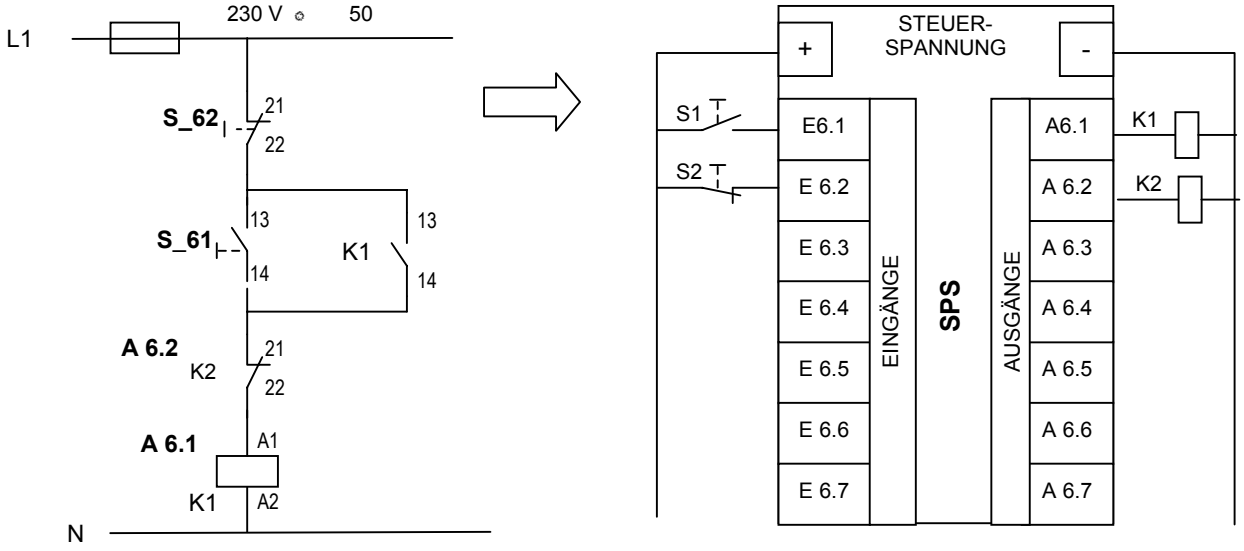
```
      U(
      O  "S_51S"
      O  "A_51"
      )
      U  "S_53"
      =  "A_51"

Netzwerk 2

      U  E   5.3
      U  E   5.2
      U  A   5.1
      =  A   5.2
```

Netzwerk 2 für K2



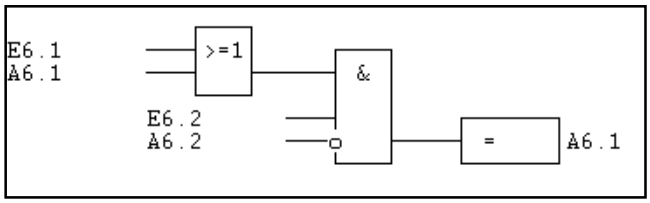


Aufgabe 4) **Eine Alternative folgt auf dem nächsten Blatt**

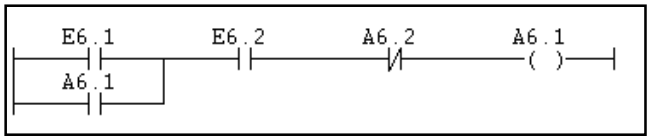


Programm für Aufgabe 4) **<u>K2 als Ausgangsabfrage</u>**

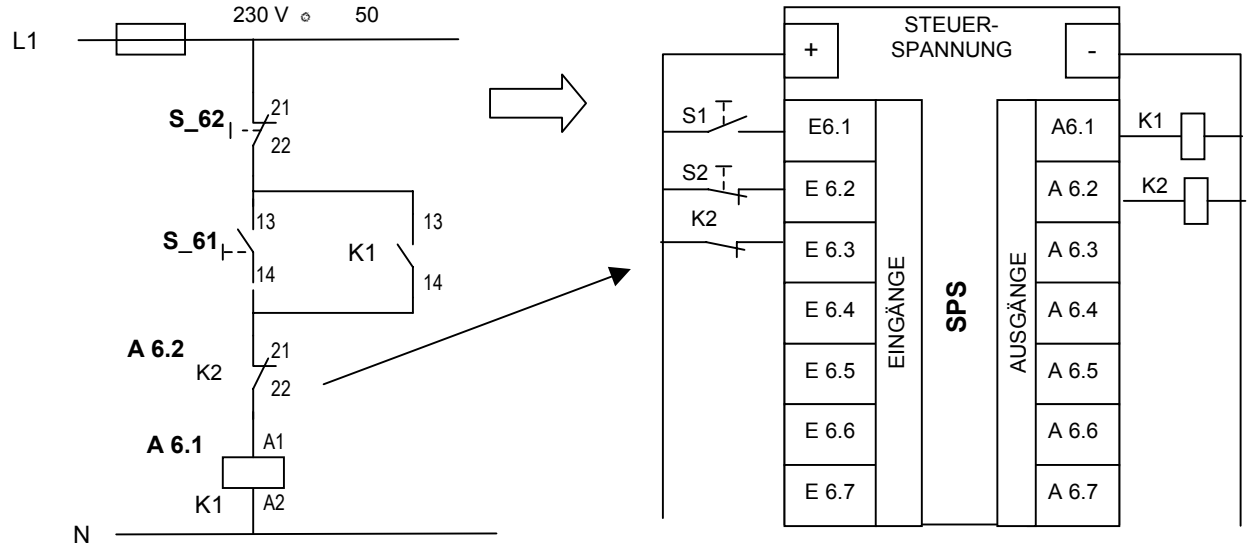
K1 darf anziehen, wenn K2 ausgeschaltet ist, d.h. wenn an A 6.2 keine Spannung liegt.



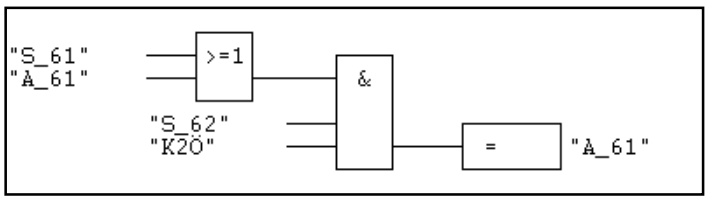

```
U(
O  E   6.1
O  A   6.1
)
U  E   6.2
UN A   6.2
=  A   6.1
```



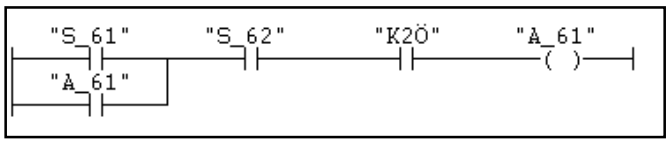
Alternative Lösung für Aufgabe 4) **K2 als Eingangsabfrage**



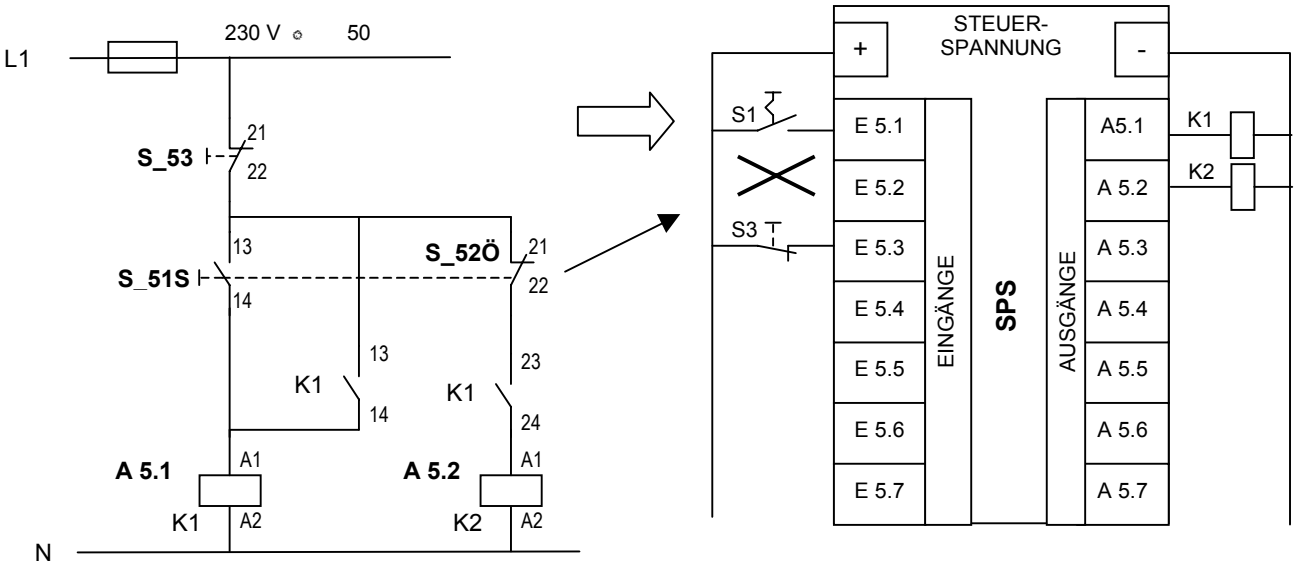
Programm für Aufgabe 4) Alternative: <u>K2 als Eingangsabfrage</u>




```
U(
O  "S_61"
O  "A_61"
)
U  "S_62"
U  "K2Ö"
=  "A_61"
```

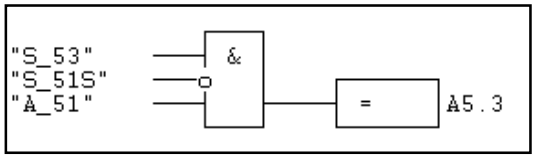


Aufgabe 5) **gleiche Funktion ohne S_52Ö**

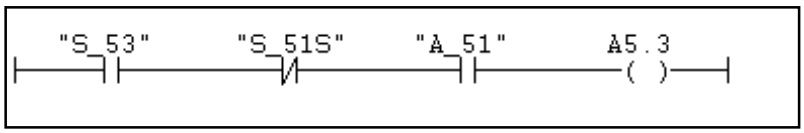


Programm für Aufgabe 5)

Anstelle von S_52Ö an E 5.2 wird S_51S an E 5.1 invertiert abgefragt.



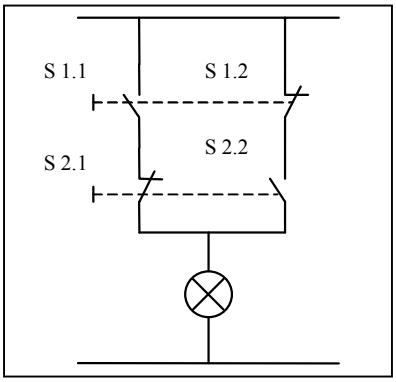

```
U  "S_53"
UN "S_51S"
U  "A_51"
=  A   5.3
```



Projekt 4: Abfrage der Eingänge

Schwerpunkte: Abfrage von Eingangs-Pegeln, Vertiefung im Umgang mit Schließern und Öffnern.

Aufgabe:

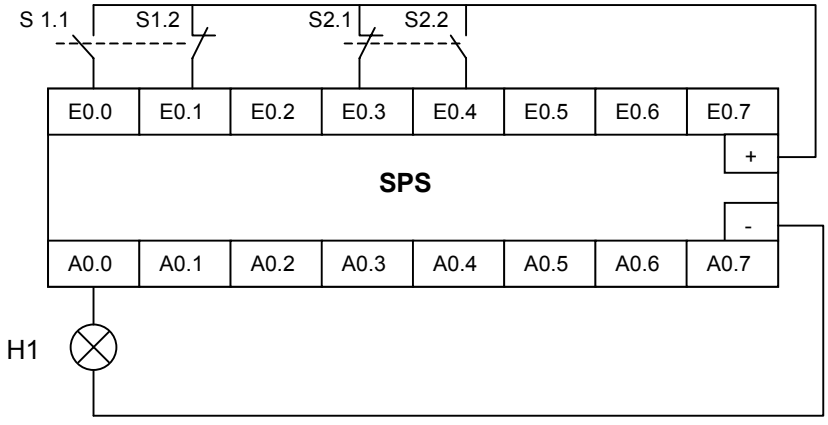


A) Erstellen Sie für die nebenstehende Schaltung den Anschlussplan einer SPS. (Verdrahten Sie **alle** Taster und die Lampe an den Klemmen der SPS).

Ein- und Ausgänge sind frei zu wählen. Tragen Sie die Spannungspegel, die abzufragen sind, an den Klemmen ein.

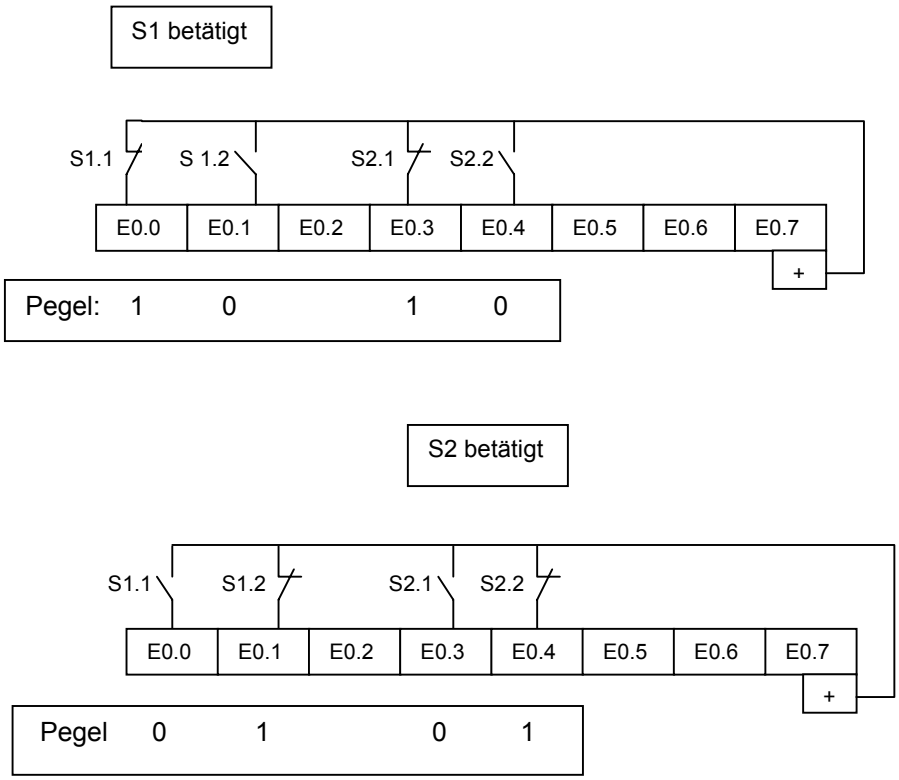
Erstellen Sie das gleichwertige SPS-Programm.
Aus Übungsgründen sollen alle 4 Kontakte abgefragt werden.

Analyse:
Sie erkennen 2 parallel geschaltete Stränge, die jeweils aus einer Reihenschaltung von 2 Tasterkontakten bestehen. Die Lampe leuchtet, wenn entweder der linke oder der rechte Stromzweig geschlossen ist. Beide parallelen Zweige können nicht zur gleichen Zeit geschlossen sein, weil die Kontakte S1.1 und S1.2 bzw. S2.1 und S2.2 mechanisch gekoppelt sind. Wenn ein Kontakt eines Tasterpaares geschlossen ist, ist der andere geöffnet - egal, ob man den Öffner oder den Schließer betrachtet.
Wie müssen die Taster an die SPS angeschlossen werden? Wir haben bereits besprochen, dass jeder Sensor (z.B. Taster) einzeln an einer Klemme angeschlossen wird. Die andere Seite aller Taster wird an die +-Klemme der internen Spannungsquelle gelegt. Die Last (Lampe, Schütz) wird zwischen einer Ausgangsklemme und der Minus-Klemme der Spannungsquelle angeschlossen.
(Die ordnungsgemäße Versorgung der SPS mit der Netzspannung ist hier nicht dargestellt).
Die Wahl der Eingänge ist beliebig gewählt worden.

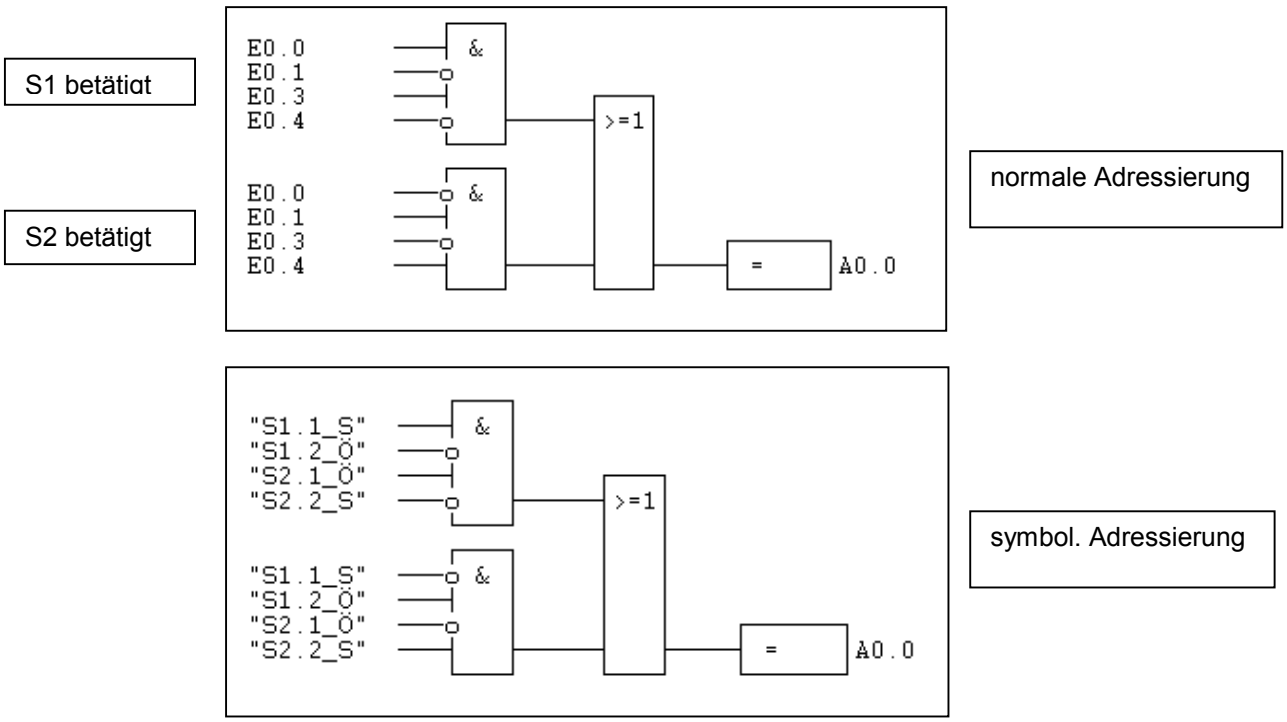


Die Verknüpfungslogik der Verdrahtung wird durch das SPS-Programm ersetzt. Um bei der Verwendung von Öffnern und Schließen etwas "sattelfest" zu werden, soll die Logik '"1 zu 1" nachgebaut werden, d. h., alle Taster sind zu programmieren.

Eingangsabfragen:



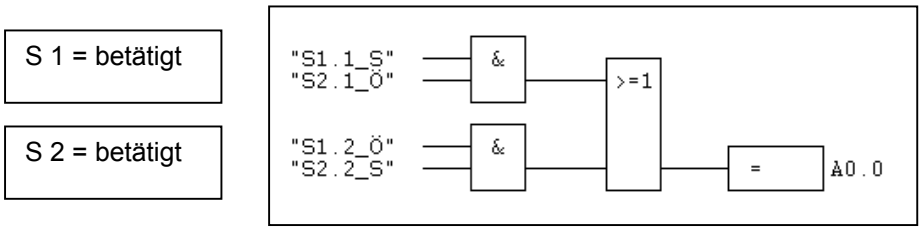
Diese beiden Konstellationen sollen laut alter Verdrahtung die Lampe leuchten lassen.
Es reicht nicht aus, nur **einen** Taster zu berücksichtigen, sondern es muss gleichzeitig abgefragt werden. ob der zweite Taster nicht betätigt wird. Daraus ergibt sich die UND-Verknüpfung aller 4 Eingänge. Da beide Fälle gleichwertig sind, sind sie ODER-verknüpft.



Sie haben gelernt:
Das Betätigen eines Tasters bedeutet nicht automatisch "1"-Pegel. Entscheidend ist, ob der entsprechende Taster geschlossen oder geöffnet ist. Ein geschlossener Kontakt - egal, ob betätigter Schließer oder nicht betätigter Öffner - liefert "1". Hingegen liefert ein offener Kontakt, der abgefragt werden soll, immer einen "0"-Pegel.
Durch die technische Aufgabe ist festgelegt, ob Schließer oder Öffner angeschlossen werden müssen und welche Zustände (gedrückt oder nicht gedrückt) ein Verknüpfungsergebnis liefern sollen. Allein für diese Situation sind die dann vorliegenden Spannungspegel zu berücksichtigen.

Variante / Vertiefung I:

Vielleicht haben Sie sich schon gefragt, warum alle 4 Kontakte UND-verknüpft wurden. Oben erwähnte ich es schon: zur besseren Übung. Derartige Doppeltaster-Abfragen kommen aber auch bei besonders sicherheitsrelevanten Endtastern vor, wenn man sich nicht nur auf einen – mechanisch anfälligen - Kontakt verlassen will. Für die rein logische Lösung der Aufgabe ist das aber eigentlich nicht erforderlich. Es reicht aus, wenn der linke oder der rechte Strompfad geschlossen ist. Dafür ergibt sich folgendes Programm:

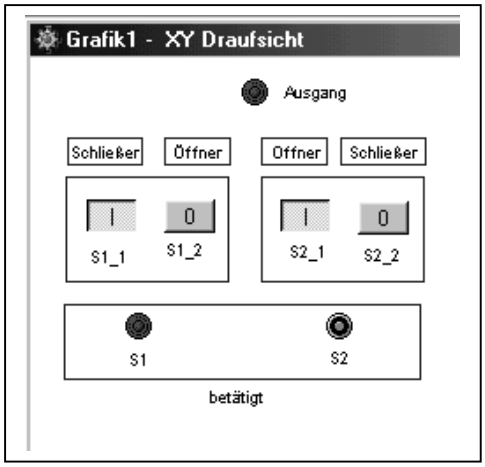


Zur Praxis mit TrySim:

Sie können die Aufgabe prüfen mit der fertigen Schaltung **<XOR>** im Verzeichnis **<Lösungen>.**
Das Programm ist so einfach zu schreiben, dass wir darauf hier nicht weiter eingehen.

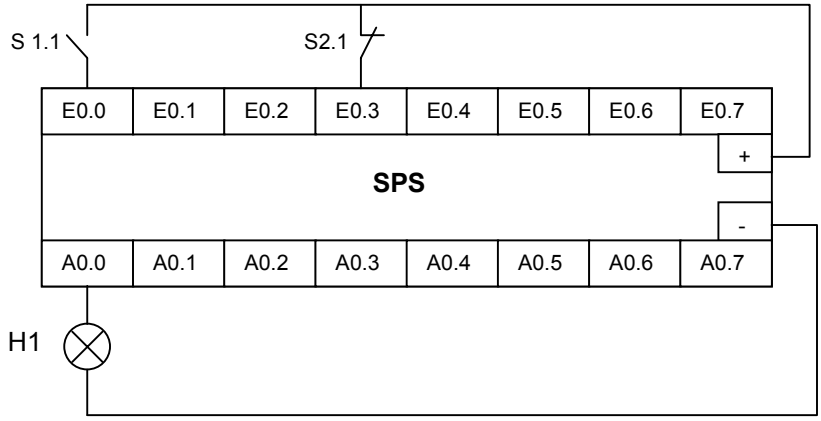
Interessanter ist vielleicht die Nachbildung der Doppeltaster. Sie wurden durch 2 Stufenschalter nachgebildet. Falls Sie das nachbauen wollen: Wählen Sie bitte bei aktivem Grafikfenster ein <Neues Element> und ziehen mit <LM> den Stufenschalter zweimal in das Grafikfenster. Öffnen Sie für einen Schalter mit <rM> das Editierfenster und wählen als <Vater> den anderen Schalter. Lassen Sie in beiden Fällen als <Typ> den "Schließer" stehen. Diese beiden Schalter sind jetzt miteinander gekoppelt, d.h., wenn Sie einen drücken ("1"), spring der zweite automatisch auf "0".
In der derzeitigen TrySim-Version müssen Sie nach dem ersten Start einmal einen Schalter drücken; danach haben Sie immer einen Kontakt geschlossen und den anderen offen.

Für den zweiten Doppeltaster wiederholen Sie bitte die Prozedur.
Weitere Informationen finden Sie unter der TrySim-Hilfe unter <Index>|"Stufenschalter".

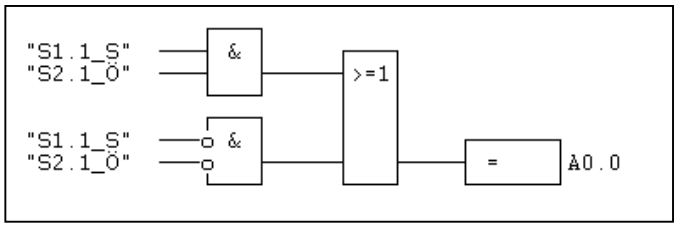


Variante / Vertiefung II:

Ist die Funktion der obigen Schaltung ausschließlich durch die Abfrage der Taster S 1.1 (Schließer) und S 2.1 (Öffner) [linker Strompfad] mittels SPS-Programm nachzubilden?



Es gilt nach wie vor die Aufgabenstellung: Wenn S1 oder S2 betätigt werden, soll die Lampe leuchten - egal, wie die Taster bestückt sind. Analog zu obiger Ausführung ist jeweils 1 Taster zu betätigen, aber beide Pegel sind abzufragen. Wird S1 (Schließer) betätigt, liefern beide Eingänge "1". Wird S2 (Öffner) betätigt, liefern beide Eingänge "0". Auch der letztere Fall kann als SPS-Programm problemlos zu einem Verknüpfungsergebnis (VKE) führen.



Empfehlung:
Es wäre toll, wenn Sie diese Übung als trivial oder überflüssig ansehen könnten. In aller Regel trifft das aber für Anfänger nicht zu und auch vermeintliche Experten machen typische Standardfehler, indem sie über die anscheinend so einfachen Grundüberlegungen hinweggehen (besser wäre: "hinwegsehen", weil auf die oben beschriebene Darstellung im Anschlussplan und den Eintrag der gewünschten Spannungspegel verzichtet wird).
Aus Erfahrung rate ich Ihnen dringend, diese häufigste Fehlerquelle durch eine konsequente Vorüberlegung in Form des Anschlussplanes und der Pegelzuordnung zu vermeiden. Das ist keine Mehrarbeit, denn der Anschlussplan gehört selbstverständlich zu den erforderlichen Unterlagen und sollte zuerst erstellt werden.

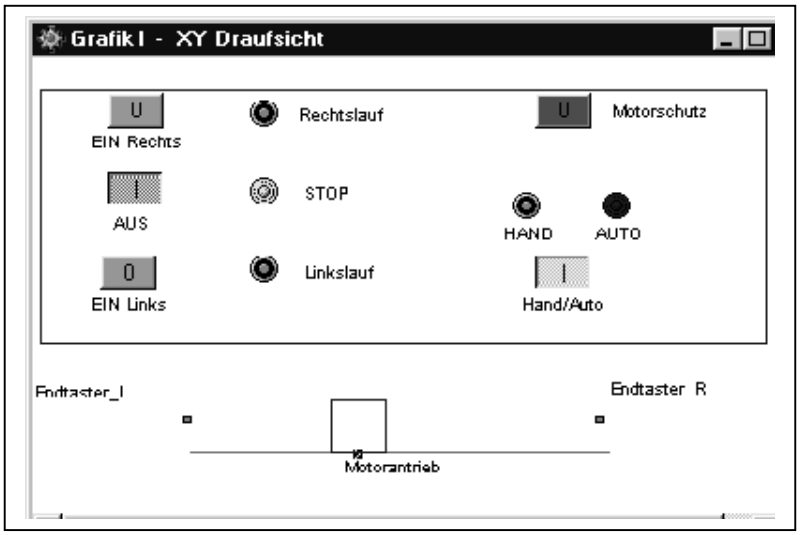
Projekt 5: Wendeschützschaltung

Schwerpunkte: Grundverknüpfungen, Abfrage von Ein- und Ausgängen, SR-FF

Aufgabe: Wendeschützschaltung

Der Tischantrieb einer Schleifmaschine soll in beiden Richtungen betrieben werden und zwar:
1. Handbetrieb: Der Motor kann mit Rechts- oder Linkslauf gestartet werden und wird entweder durch den jeweiligen Endtaster oder durch die <STOP>-Taste angehalten. Handbetrieb darf nur bei nicht betätigtem Wahlschalter (" 0 " - Stellung) möglich sein.
2. Automatikbetrieb: Der Motor kann mit Rechts- oder Linkslauf gestartet werden. Beim Erreichen eines Endtasters kehrt sich die Drehrichtung automatisch um. Abgeschaltet wird über die <STOP>-Taste. Automatikbetrieb ist nur bei betätigtem Wahlschalter (" 1 "-Stellung) möglich.

Das Projekt ist vorbereitet im Ordner <Vorlagen>.



Steuerpult und Anlage

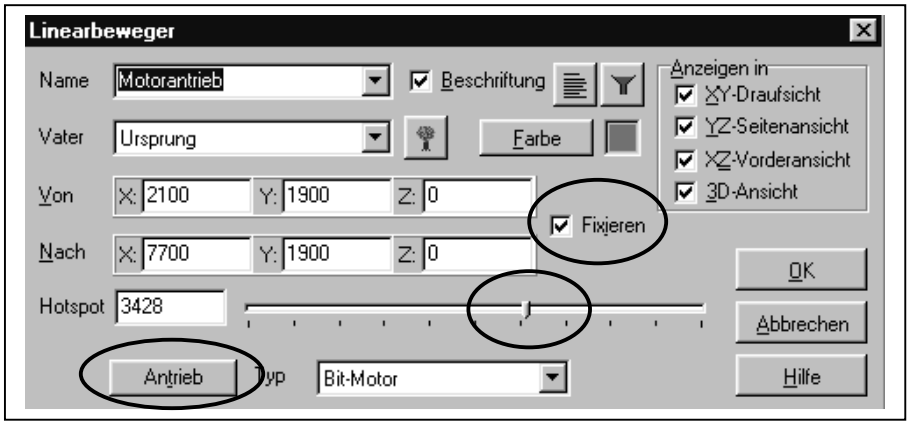
Symbolische Adresse	Operand	Datentyp	Kommentar
EIN_R	E 0.2	BOOL	Taster EIN Rechtslauf (Schließer)
EIN_L	E 0.4	BOOL	Taster EIN Linkslauf (Schließer)
AUS	E 0.3	BOOL	Aus-Taster (Öffner)
MS	E 0.5	BOOL	Motorschutzschalter
WAHL	E 0.6	BOOL	Hand = 0 / Auto = 1
LED_R	A 0.2	BOOL	Anzeige Rechtslauf
LED_L	A 0.4	BOOL	Anzeige Linkslauf
LED_STOP	A 0.3	BOOL	Anzeige STOP
LED_HAND	A 0.5	BOOL	Anzeige Handbetrieb
LED_AUTO	A 0.6	BOOL	Anzeige Automatikbetrieb
MOTOR_R	A 0.0	BOOL	Motorantrieb Rechtslauf
MOTOR_L	A 0.1	BOOL	Motorantrieb Linkslauf
ET_L	E 0.1	BOOL	Endtaster linker Anschlag (Ö)
ET_R	E 0.0	BOOL	Endtaster rechter Anschlag (Ö)
RECHTSFF	M 0.0	BOOL	SR_FF Rechtslauf
LINKSFF	M 0.1	BOOL	SR_FF Linkslauf

Symboltabelle

Zur Praxis mit TrySim:

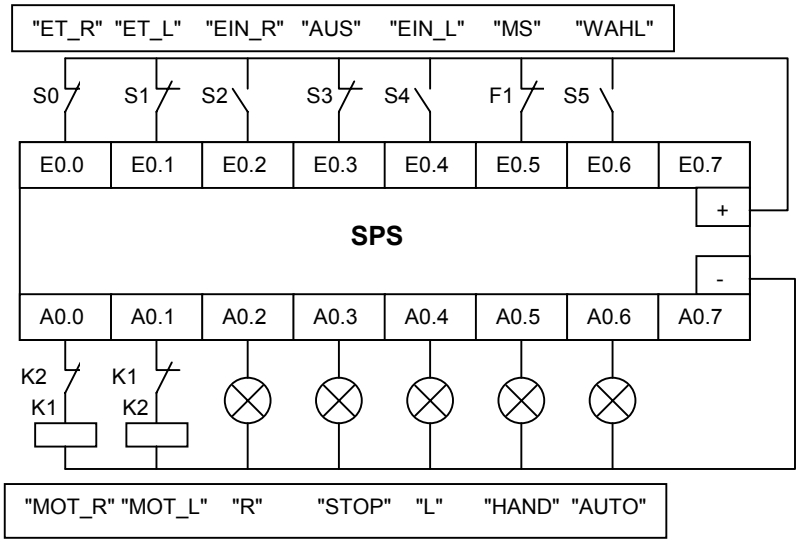
Öffnen Sie im Ordner **<Vorlagen>** bitte **<Wende_V>** und speichern Sie das Projekt in Ihrem Übungsordner unter einem sinnvollen Namen.

Der Motorschutz ist als Schalter vorbereitet. Für den Betrieb muss er von Hand erst einmal geschlossen werden ("1"-Abfrage).
Als Antrieb wurde ein Linearbeweger gewählt. Wenn die "Anlage" ausgeschaltet ist und die Cursorspitze auf den waagerechten Strich des Antriebs zeigt, ändert sich das Cursorsymbol zur Hand. Gleichzeitig werden die zugeordneten Operanden für die Antriebe (V = vorwärts; R = rückwärts) angezeigt.
Wenn Sie in der "Anlage" mit <rM> auf den waagerechten Strich klicken, öffnet sich das zugehörige Editierfenster. Wenn Sie dort das Häkchen im Fenster <Fixieren> wegklicken, können Sie Lage und Länge des Antriebs verändern. Ziehen Sie dazu die Endpunkte des Striches mit <LM>.

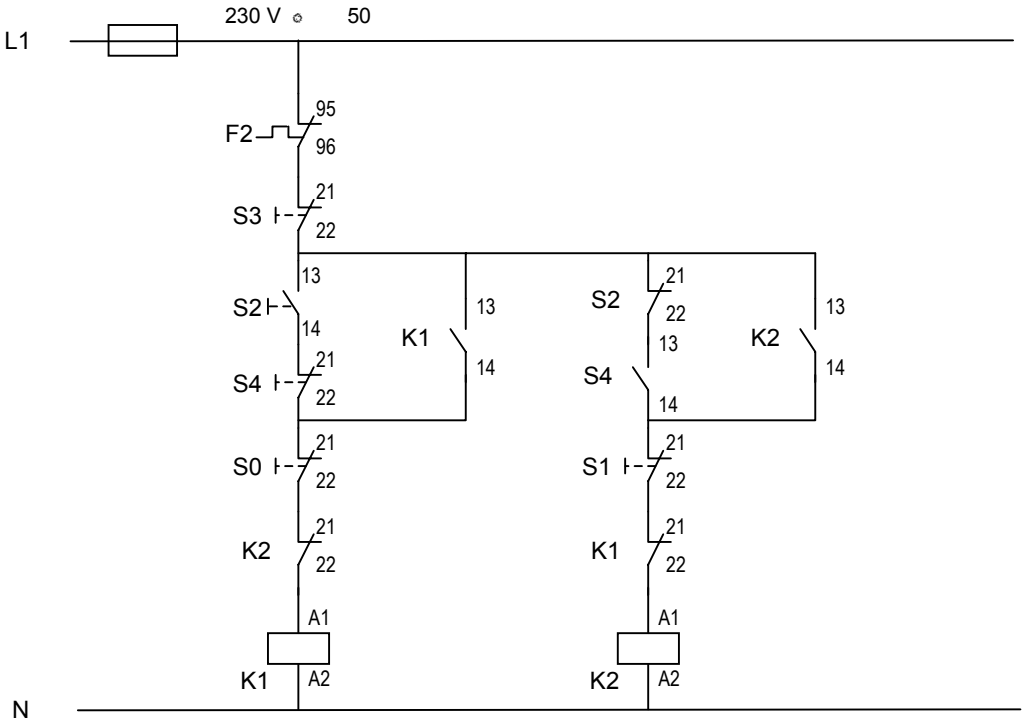


Im Fenster <Hotspot> wird die augenblickliche Position des Punktes angegeben, der längs des Bahnweges durch den Antrieb verstellt wird - bezogen auf den gewählten Startpunkt.
Mit <LM> auf den Schieberegler können Sie die Position des Hortspots verschieben. Die Veränderung wird direkt in die "Anlage" übertragen.
Mit <LM> auf <Antrieb> können Sie weitere Details bestimmen. Sehen Sie dazu auch unter TrySim-Hilfe, <Index> "Linearbeweger".
Am Hotspot wurde als Material ein Kasten angebunden, der nun durch die Operanden A 0.0 (Rechtslauf) und A 0.1 (Linkslauf) hin- und herfahren soll.

Anschlussplan (ohne Netzanschluss)



Verdrahtungsprogrammierte Steuerung:



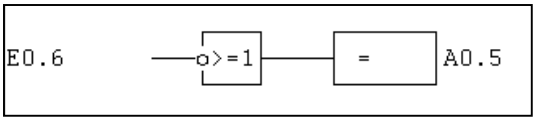
Die Doppeltaster S2 und S4 werden für die SPS durch 2 normale Schließer ersetzt, die beide abgefragt werden. Die Öffner von K1 und K2 sind an der SPS nicht vorhanden. Stellvertretend werden die Pegel der zugehörigen Ausgangsklemmen abgefragt (an denen die entsprechenden Schützspulen angeschlossen sind).

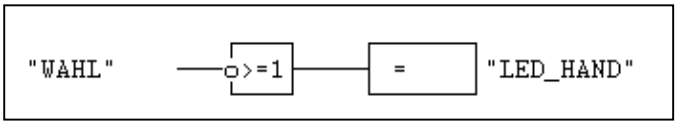
Natürlich wäre es möglich, einen zusätzlichen Kontakt von K1 und K2 an **Eingangsklemmen** zu legen und dort zusätzlich abzufragen, ob die Schütze tatsächlich in der gewünschten Stellung sind. Da in der Simulation aber keine Schütze geschaltet werden, wird diese Art der Lageprüfung hier nicht programmiert.

PROGRAMM:

Netzwerk 1: Auswahl Hand

Für das Programm <Handbetrieb> wird der Wahlschalter nicht gedrückt.
Die LED zeigt den entsprechenden Zustand an.

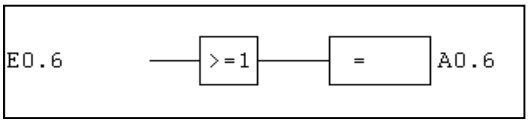


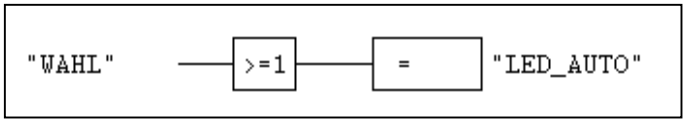


Darstellung: ohne.... ...bzw. mit symbolischer Darstellung der Operanden

Netzwerk 2: Auswahl Automatik

Für das Programm <Automatikbetrieb> wird der Wahlschalter gedrückt.
Die LED zeigt den entsprechenden Zustand an.

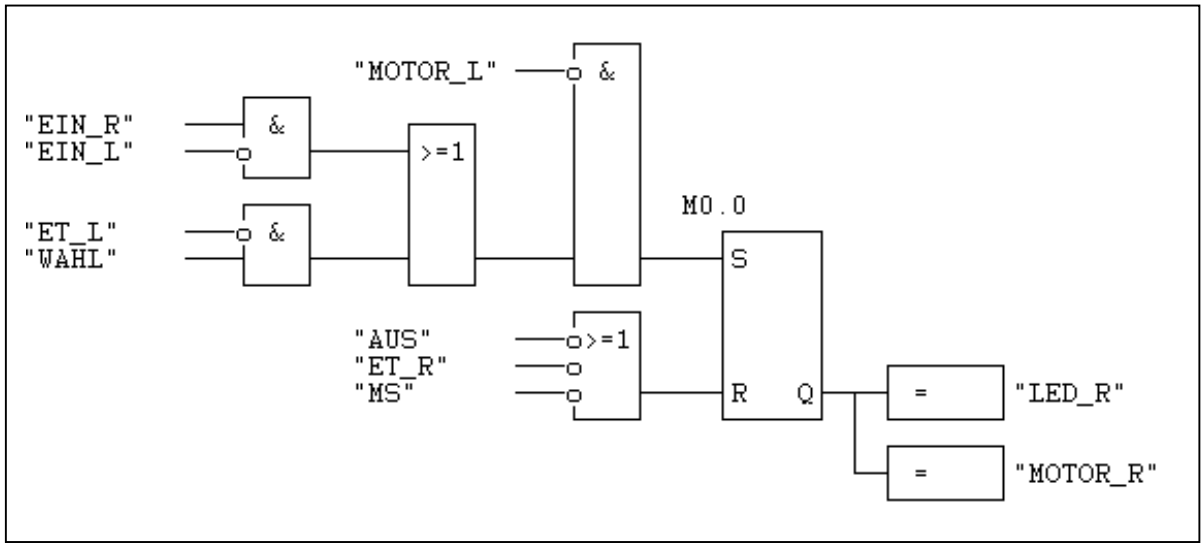




Netzwerk 3: Rechtslauf

Wir wollen uns bei der Aufgabe an den üblichen Verriegelungen der Schützsteuerung orientieren. Zu verhindern ist der gleichzeitige Betrieb von Rechts- und Linkslauf, da sonst im Netz ein Kurzschluss entstehen würde. Es darf zwar jede Drehrichtung gewählt werden, aber gleichzeitig muss überprüft werden, dass der andere Starttaster nicht betätigt wird. In der Schütztechnik wird die Aufgabe mittels Doppeltaster gelöst. Beim SPS-Anschluss ist diese Vielzahl an Tastern nicht erforderlich, weil der Stromkreis durch die Kontakte nicht geschlossen sein muss. Es kann auch der Pegel "0" abgefragt werden.
Da grundsätzlich ein "EIN"-Befehl mit einem betätigten Schließer ("1"-Pegel) gegeben wird und der zweite Schließer unbetätigt sein muss, ist der zweite "Ein"-Taster auf "0"-Pegel abzufragen.

Eine andere Art der Verriegelung ist auch üblich, indem man abfragt, ob das Schütz für die entgegengesetzte Drehrichtung ausgeschaltet ist. In der Schützensteuerung wird das durch eine Reihenschaltung mit einem nicht betätigten Öffner (also in Ruhestellung) des zweiten Schützes erreicht.
Diese Aufgabe ist geeignet, die "0"- / "1"-Problematik bei Öffnern zu durchdenken. Bei einer Eingangsabfrage eines Öffners in Ruhestellung (z.B. bei einem "AUS"-Taster) ist der "1"-Pegel abzufragen, weil die Spannung über den geschlossenen Kontakt an die Eingangsklemme gelegt wird. Bei einem Öffner in Ruhestellung, der einen Schützkontakt darstellt, ist ein "0"-Pegel abzufragen!
Erläuterung: Wenn der Schützkontakt in Ruhestellung ist, bedeutet das, dass die zugehörige Schützspule nicht an Spannung liegt. Somit liegt auch an der entsprechenden SPS-Ausgangsklemme (Operand) auch keine Spannung. Und nur bis zu dieser Ausgangsklemme kann die SPS Werte (Pegel) erfassen. Was außerhalb der SPS angeschlossen ist, kann sie nicht einmal erahnen. Folglich existiert für die SPS auch nicht der Verrriegelungs-Öffner der Schützensteuerung. Die SPS kann also lediglich an der entsprechenden Ausgangsklemme prüfen, ob **keine** Spannung am zweiten Ausgang anliegt. Diese Prüfung ist gleichwertig mit der Verriegelung durch den Öffner des anderen Schützes.

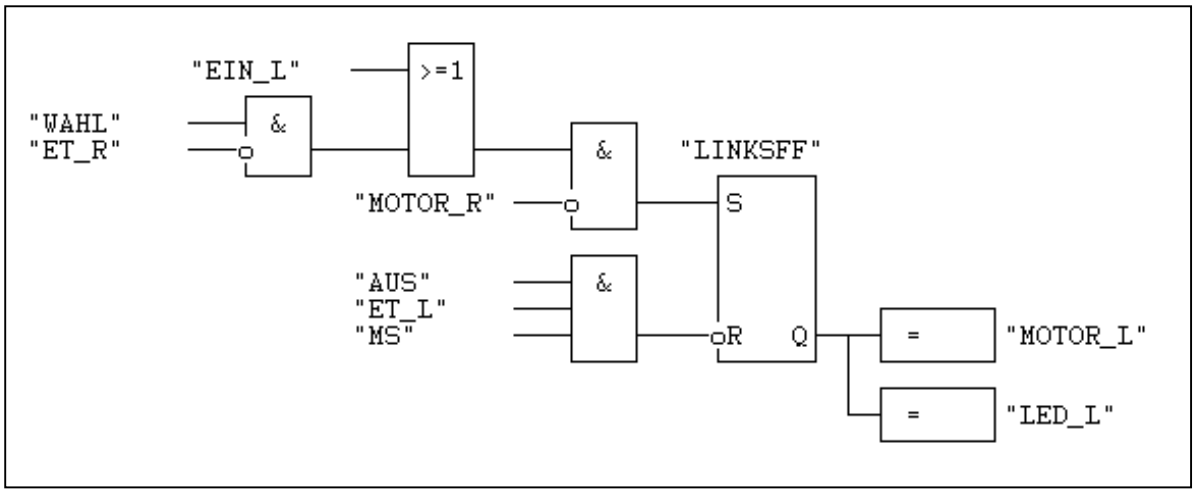


Hinweis:
Diese Art der "Ausgangsverriegelung" im Programm ersetzt nicht die zusätzlich zwingend erforderliche "hardware-Verriegelung" durch die Öffner der Drehrichtungsschütze. Ein Schütz könnte auch im spannungslosen Zustand "kleben", d.h. die Kontakte könnten verschweißt sein o.ä., so dass der Anker nicht abfällt. Nur durch entsprechende Kontaktverriegelungen lässt sich dann ein Kurzschluss vermeiden.
Das schönste SPS-Programm entbindet den Elektriker also nicht von weitergehenden Sicherheitsüberlegungen.

Wenn der Wahlschalter auf "Automatik" steht, also "1"-Pegel führt, und der linke Endtaster angefahren wird (wegen der Drahtbruchsicherheit ein Öffner, der bei Betätigung öffnet), wird auch "Rechtslauf" gestartet.
Alle Rücksetzabfragen ("AUS", "ET_R", "MS") führen im Normalzustand "1"-Pegel. Da das FF mit "1" am <R>-Eingang zurückgesetzt wird, müssen alle Abfragen einzeln invertiert und ODER-verknüpft werden.

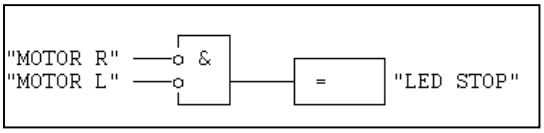
Netzwerk 4: Linkslauf

Ein Beispiel ohne Verriegelung der Eingänge, jedoch mit Abfrage des anderen Ausgangs.



Die Rücksetzbedingungen sind im Prinzip genau so wie im Netzwerk 3, jedoch wurden sie zur Abwechslung gemäß dem vielleicht bekannten de Morgan´schen Gesetz nicht ODER- sondern UND-verküpft (mit geänderten Invertierungen).

Netzwerk 5: STOP-LED



Wenn kein Motorschütz eingeschaltet ist, leuchtet die STOP-LED.

Vgl. TrySim-Projekt <Wende> im Verzeichnis <Lösungen>.

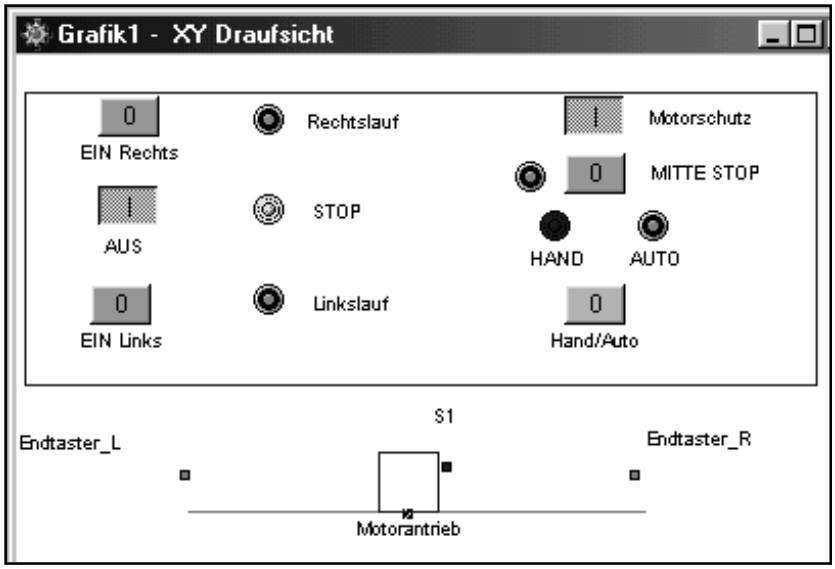
Sie haben gelernt:

- Es gibt kein SPS-Schaltsymbol für einen Öffner, sondern es ist zu entscheiden, ob ein "0"- oder "1"-Pegel abgefragt werden soll.
- Tasterverriegelungen im klassischen Sinne sind bei einem SPS-Programm nicht erforderlich. Ein und derselbe Taster kann in einem Netzwerk auf „1“ und in einem anderen auf „0“ abgefragt werden, so dass durch die UND-verknüpfte Abfrage mehrerer Eingänge auch eine gleichwertige Verriegelung erreicht werden kann.
- Die Softwareverriegelung von Ausgängen (durch die Abfrage der entsprechenden Operanden) ersetzt nicht die externe Kontaktverrieglung (über Schützkontakte).
- Gleichwertige Verknüpfung können nach de Morgan durch unterschiedliche Bausteine erzeugt werden.

Projekt 6: Wendeschützschaltung 2

Aufgabe: Wendeschützschaltung mit Flankenauswertung
Der Tischantrieb einer Schleifmaschine soll in beiden Richtungen betrieben werden und zwar:
1. Handbetrieb: Der Motor kann mit Rechts- oder Linkslauf gestartet werden und wird entweder durch den jeweiligen Endtaster oder durch die <AUS>-Taste angehalten. Handbetrieb darf nur bei nicht betätigtem Wahlschalter (" 0 " - Stellung) möglich sein.
2. Automatikbetrieb: Der Motor kann mit Rechts- oder Linkslauf gestartet werden. Beim Erreichen eines Endtasters kehrt sich die Drehrichtung automatisch um. Abgeschaltet wird über die <AUS>-Taste. Automatikbetrieb darf nur bei betätigtem Wahlschalter (" I "-Stellung) möglich sein.
3. Halt in der Mittelstellung: Wenn bei freiem Lauf die <Mitte>-Taste gedrückt wird, soll der Rohling in der Mitte anhalten. u.z. wenn er von links kommt, vor S1, wenn er von rechts kommt, an S2. Wenn der Rohling gerade den Schalter überfährt, an dem er halten soll, und dann die <Mitte>-Taste gedrückt wird, soll der Rohling weiterfahren und erst auf dem Rückweg in der Mitte halten.
Die Mittelposition soll über eine Flankenauswertung von S1 programmiert werden.

Anlage:

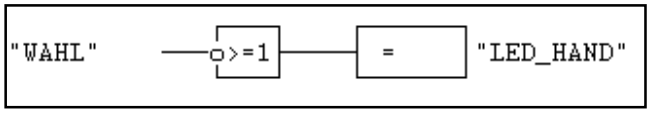


Symbolische Adresse	Operand	Datentyp	Kommentar
MOTOR_R	A 0.0	BOOL	Motorantrieb Rechtslauf
MOTOR_L	A 0.1	BOOL	Motorantrieb Linkslauf
LED_R	A 0.2	BOOL	Anzeige Rechtslauf
LED_STOP	A 0.3	BOOL	Anzeige STOP
LED_L	A 0.4	BOOL	Anzeige Linkslauf
LED_HAND	A 0.5	BOOL	Anzeige Handbetrieb
LED_AUTO	A 0.6	BOOL	Anzeige Automatikbetrieb
LEDMSTOP	A 0.7	BOOL	Anzeige Stop in der Mitte
ET_R	E 0.0	BOOL	Enttaster rechter Anschlag
ET_L	E 0.1	BOOL	Endtaster linker Anschlag
EIN_R	E 0.2	BOOL	Taster EIN Rechtslauf
AUS	E 0.3	BOOL	Aus-Taster (Öffner)
EIN_L	E 0.4	BOOL	Taster EIN Linkslauf
MS	E 0.5	BOOL	Motorschutzschalter (Öffner)
WAHL	E 0.6	BOOL	Hand = 0 / Auto = 1
Mitte_RT	E 0.7	BOOL	Endtaster Mitte rechts S1
Mitte	E 1.1	BOOL	Halt in Mittelstellung (S)
M_STOP	M 3.0	BOOL	Stop in Mitte vorbereiten
P_FLANKE	M 10.1	BOOL	pos. Flanke erforderlich
N_FLANKE	M 10.0	BOOL	negative Flanke erforderlich

Im Verzeichnis <Vorlagen> ist das Projekt unter <Wende_FL_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.
Die vorstehende Symboltabelle und die Anlage sind bereits vorbereitet.

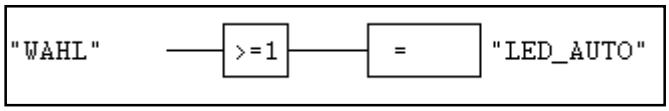
OB 1; Netzwerk 1: Auswahl Hand

Für das Programm <Handbetrieb> wird der Wahlschalter <Hand/Auto> nicht gedrückt.
Die LED zeigt den entsprechenden Zustand an.

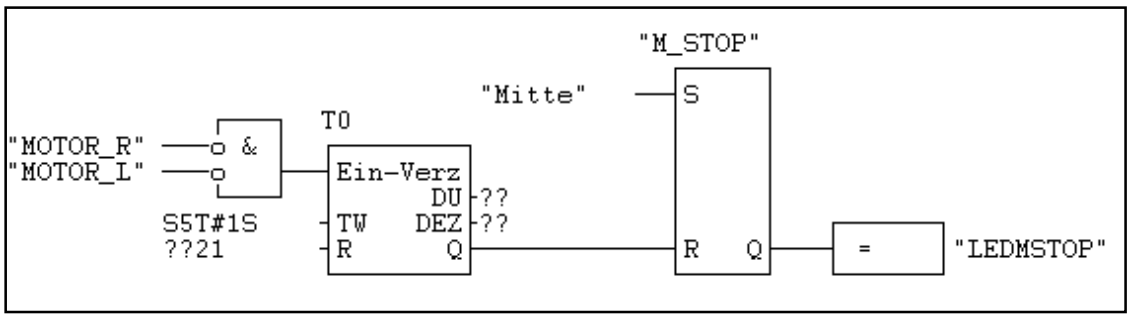


OB 1; Netzwerk 2: Auswahl Automatik

Für das Programm <Automatikbetrieb> wird der Wahlschalter <Hand/Auto> gedrückt.
Die LED zeigt den entsprechenden Zustand an.



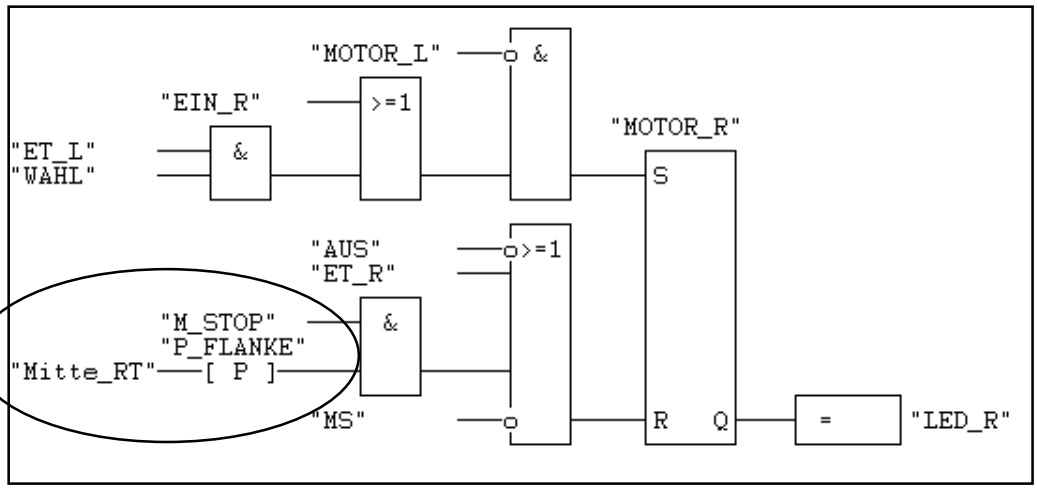
OB 1; Netzwerk 3: Stop in Mitte vorbereiten



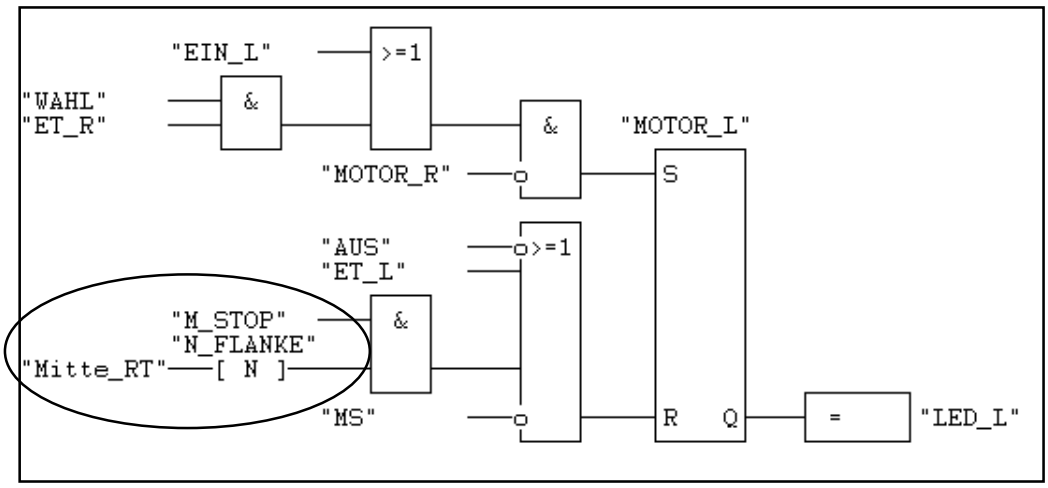
Mit Druck auf Taster <MITTE STOP> speichert „M_STOP“ die Anweisung, den Rohling in der Mittelposition zu stoppen. Diese Anweisung darf nur bis zum Erreichen der Mittelposition gelten, d.h., bei einem Neustart des Motorantriebes müsste <MITTE STOP> erneut betätigt werden. Der Rücksetzbefehl wird vom Stillstand des Motorantriebes abgeleitet. Dabei ergibt sich noch ein kleines Problem: Beim Erreichen eines Endtasters – beim Umschalten der Drehrichtung – sind kurzzeitig auch beide Schütze ausgeschaltet. Das darf aber nicht zum Abschalten von „M_STOP“ führen. Deshalb wird ein Zeitglied dazwischengesetzt, das länger verzögert, als für das reine Umschalten erforderlich ist.

OB 1; Netzwerk 4: Rechtslauf

Im Gegensatz zum vorausgegangenen Projekt <Wende> wird hier nur noch ein Endtaster für die Mittelposition benötigt. Trotzdem soll der Rohling an der gleichen Position anhalten – egal, ob er von links oder von rechts kommt. Der Endtaster <S1>, abgefragt über „Mitte_RT“, ist ein Schließer, d.h., er liefert immer bei Betätigung durch den Rohling eine logische „1“. Das Signal soll aber richtungsabhängig ausgewertet werden. Das wird durch die Auswertung des Signalwechsels (0 → 1 bzw. 1 → 0), also der Flanke, erreicht. Wenn der Antrieb nach recht läuft, soll er stoppen, wenn ein 0 → 1-Übergang („positive Flanke“) vorliegt.
Das Symbol --[P]—wird eingefügt bei aktivem Editierfenster und Wahl des Icons <FUP-Elemente> und dort <Steigende Flanke>. Über dem Symbol sind die „???“ durch einen sonst nirgends benutzten Merkeroperanden zu ersetzten!
Jetzt wirkt das Signal <Mitte_RT> richtungsabhängig.

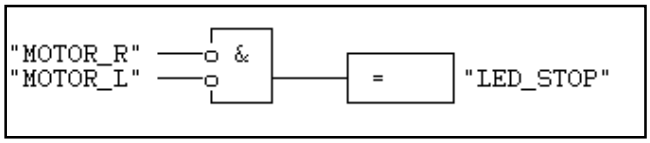


OB 1; Netzwerk 5: Linkslauf



Wenn der Antrieb nach links läuft, soll der Rohling **nach dem Passieren des Endtasters <S1>** stoppen, also wenn ein 1 → 0-Übergang („fallende Flanke“) vorliegt. Deshalb wird hier das entsprechende Symbol eingebaut und einem Merkeroperanden zugewiesen. .

OB 1; Netzwerk 6:



Wenn beide Richtungsschütze ausgeschaltet sind, leuchtet die <STOP>-LED.

---

vgl. TrySim-Projekt <Wende_FL> im Verzeichnis <Lösungen>.

Sie können jetzt:

- Eine Flankenabfrage programmieren.
- Einen Antrieb richtungsabhängig programmieren und lagerichtig stoppen.

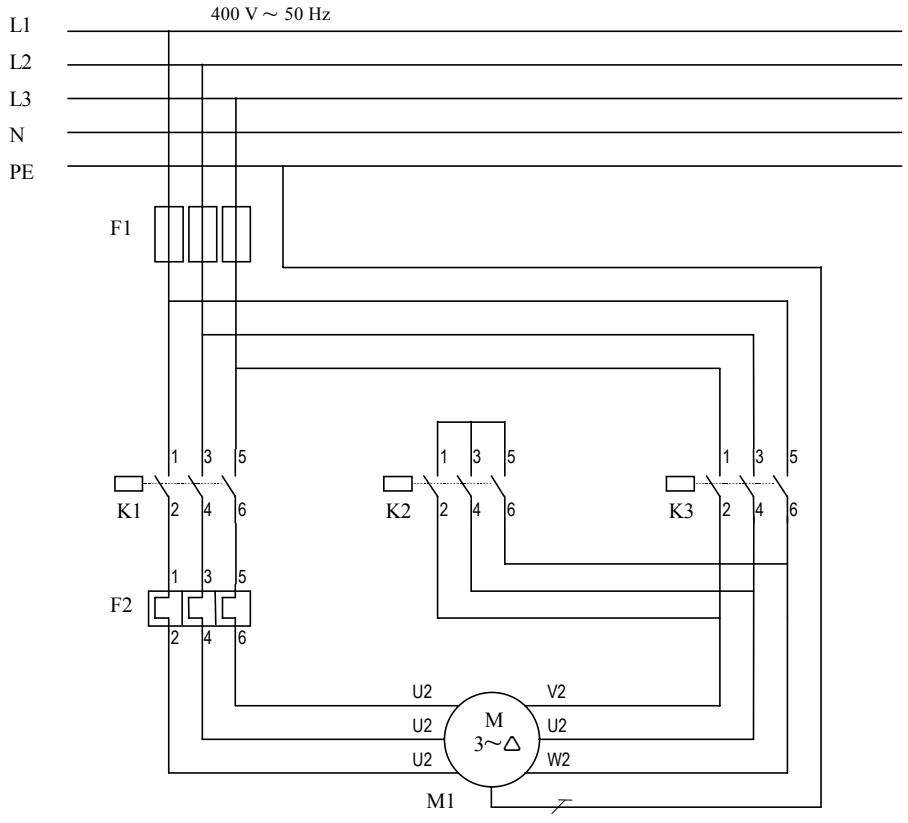
Projekt 7: Stern-Dreieck-Umschaltung von Hand

Schwerpunkte: Grundverknüpfungen, Umwandlung einer Schützensteuerung in ein SPS-Programm.
Richtige Reihenfolge der Verknüpfungen im FUP-Programm.

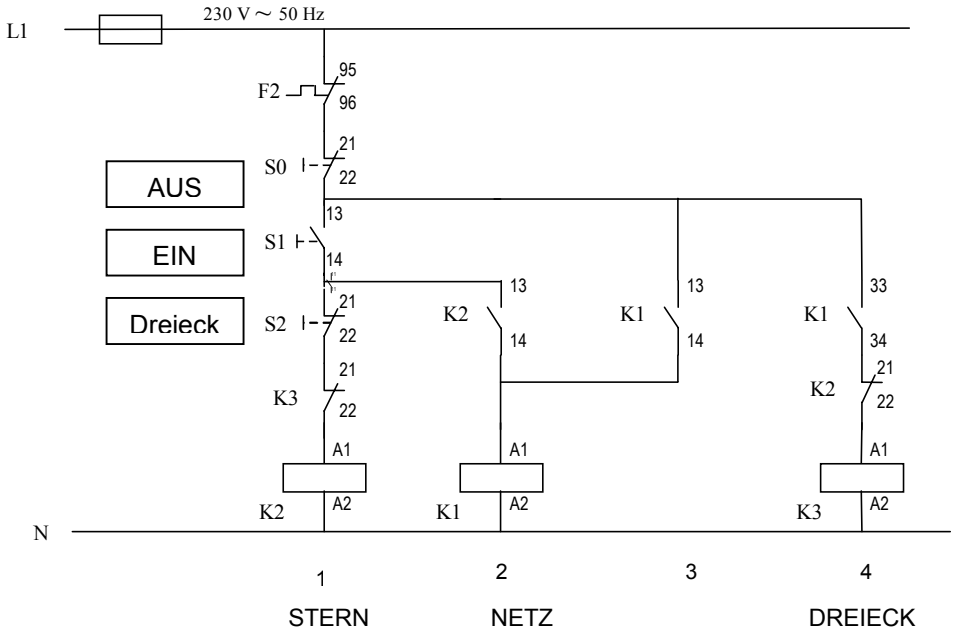
Aufgabe: Stern- / Dreieck-Anlauf

Ein Drehstrommotor soll im Stern anlaufen und von Hand in die Dreieckschaltung umschalten. Natürlich muss der Motor bei Überlast abschalten. Die AUS-Funktion hat Vorrang vor dem Einschaltbefehl.

Laststromkreis

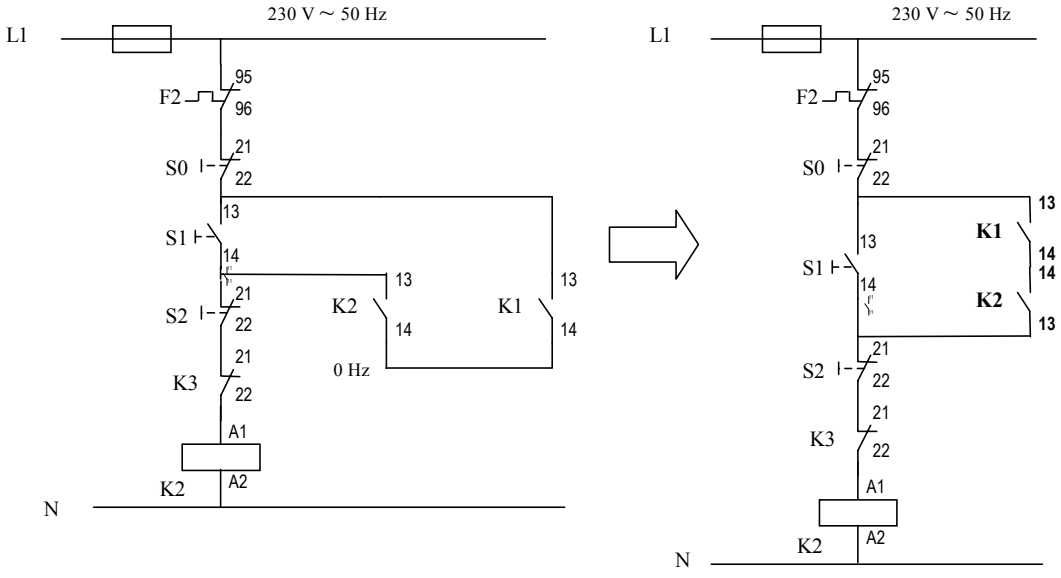


Steuerstromkreis



Zuerst ist es nützlich, die Schaltung zu analysieren. Das Sternschütz dient der Vorbereitung; es wird im spannungslosen Zustand geschlossen. Erst anschließend wird der Motor über das Netzschütz an Spannung gelegt. In der umgekehrten Reihenfolge könnte man zwar auch starten, aber die Fachkraft hat alle Auswirkungen einer Fehlschaltungen zu berücksichtigen: Falls das Sternschütz - aus welchem Grunde auch immer - nicht anziehen würde (der Motor also nicht drehen würde), läge die Wicklung an Spannung. Ein Laie könnte - vorschriftswidrig - die spannungsführenden Klemmen des Motors berühren, weil er glaubt, ein stillstehender Motor wäre spannungslos.
Wenn das Dreieckschütz K3 nicht angezogen hat (K3/21-22 ist in Ruhestellung), kann Sternschütz K2 mit S1 an Spannung gelegt werden. S1 ist ein Taster, K2 muss also in Selbsthaltung gehen können. Hierfür müssen wir die Strompfade 2 und 3 betrachten: Über K2/13-14 wird ebenfalls - als Folge - das Netzschütz K1 eingeschaltet, welches über den Kontakt K1/13-14 in Selbsthaltung geht.
Diese beiden Kontakte K1/13-14 und K2/13-14 liegen parallel zu S1 und ermöglichen gemeinsam die Selbsthaltung für K2. So muss auch die Schaltung für das SPS-Netzwerk (K2) aufgebaut sein. Durch den umgezeichneten Steuerstromkreis wird der Sachverhalt verdeutlicht. Im Netzwerk (K1) werden die selben Kontakte (in anderer logischer Reihenfolge !) ebenfalls benötigt.
Es ist ferner darauf zu achten, dass jedes Netzwerk (für jedes Schütz separat) alle Kontakte des Stromkreises von L1 bis zur Schützspule berücksichtigt, also in diesem Fall immer F2 und S0.
Umgeschaltet in Dreieck wird durch das Öffnen von S2. Nachdem K2 abgefallen ist, wird im Strompfad 4 der Stromkreis für K3 geschlossen.

Vorlage für Netzwerk 1: Sternschaltung, Sternschütz K2

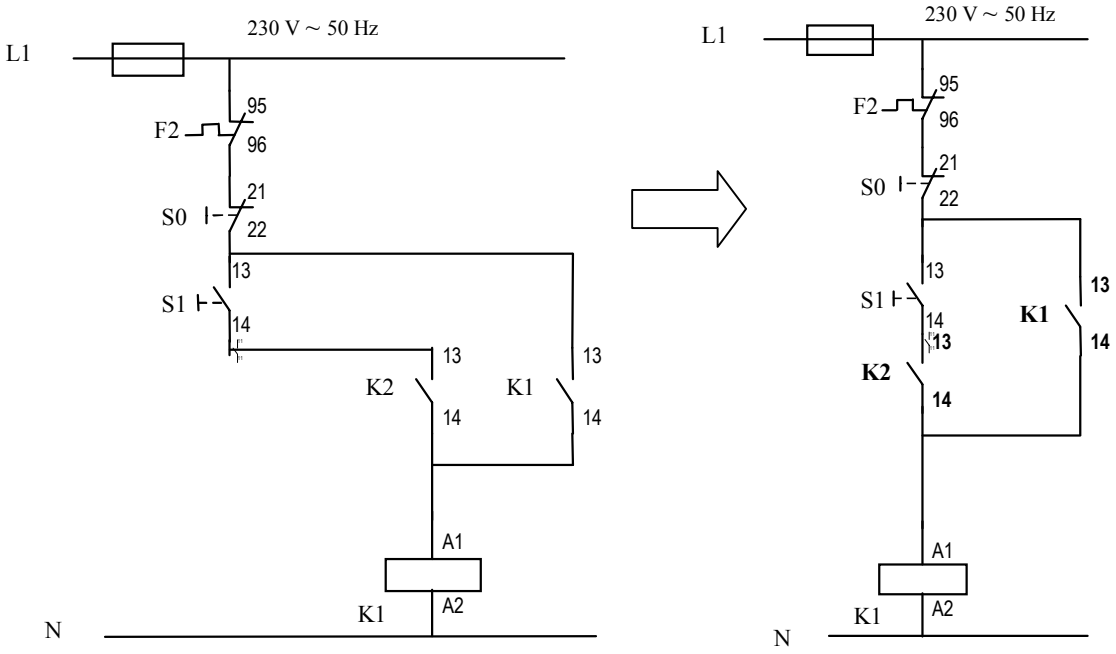


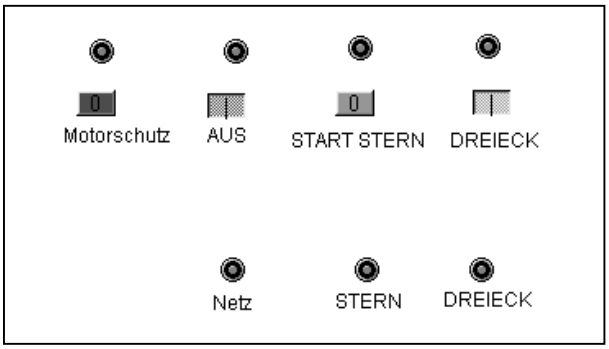
Die Teile der Schaltung, die für das Netzwerk K2 zuständig sind, werden isoliert dargestellt.

Die Schaltung wurde umgezeichnet, um die Parallelschaltung zu verdeutlichen.

Netzwerk 2: Netzschütz

Beide Schütze (K1 und K2) verwenden die selben Kontakte! Diese Kontakte müssen also in beiden Netzwerken programmiert werden.



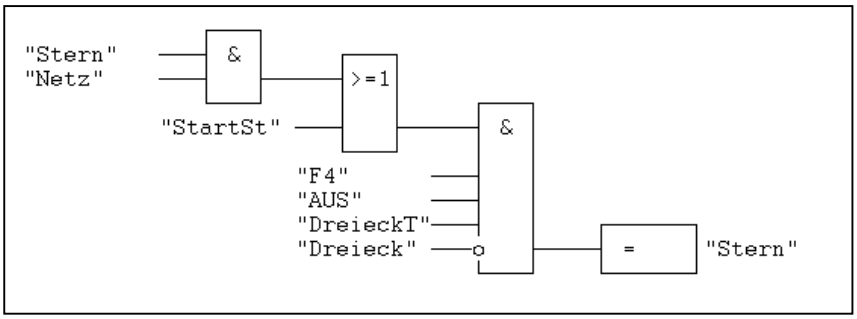


Symboltabelle

Nummer	Symbol	Adresse	Typ	Kommentar
1	F4	E 0.0	BOOL	Motorschutz (Ö)
2	AUS	E 0.1	BOOL	Aus-Taster (Ö)
3	StartSt	E 0.2	BOOL	Starttaster Sternschaltung (S)
4	DreieckT	E 0.3	BOOL	Dreieck-Taster (Ö)
5	Netz	A 0.0	BOOL	Netzschütz
6	Stern	A 0.2	BOOL	Sternschütz
7	Dreieck	A 0.3	BOOL	Dreieckschütz

Programm:

Netzwerk 1: Sternschütz

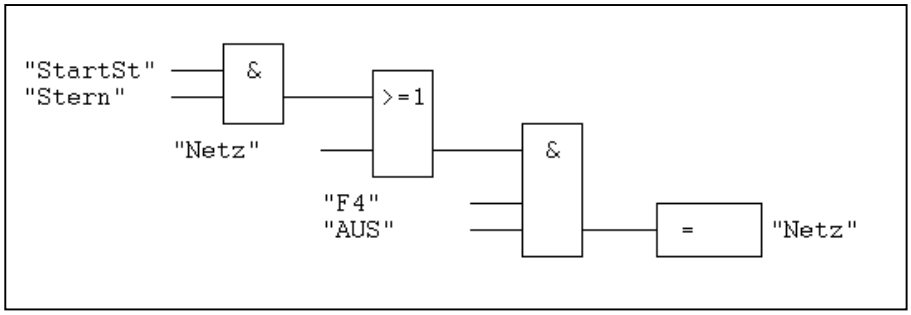


Sternschütz
Die "Netz"-Abfrage entspricht der Verknüpfung der Schützensteuerung.
Im SPS-Programm könnte auf die "Netz"-Abfrage verzichtet werden, weil hier keine verdrahtete Verbindungen zu den anderen Schützen (Netzwerken) bestehen. Jedes Netzwerk ist "autonom" und teilt sich nicht Kontakte wie in dieser Schützensteuerung. (Natürlich könnte man auch die Schützensteuerung durch einen weiteren Schließer von K2 so ändern, dass die Selbsthaltung von K2 und dessen Einschaltkontakt für K1 entkoppelt würden. In der aufgezeigten Standardschaltung wird dieser Kontakt gespart. Sie ist so eine schöne Verständnisübung für die direkte Übersetzung in ein SPS-Programm.

Hier wird nur die **Ausgangs**klemme "Dreieck" abgefragt. Vergleichen Sie die Schützensteuerung: K3/21-22 steht für die SPS-Abfrage nicht zur Verfügung. Stellvertretend wird die SPS-Klemme abgefragt, an der das Schütz angeschlossen ist. Wenn an dieser Klemme keine Spannung liegt, ist auch das Schütz in Ruhestellung - was dem geschlossenen Öffnerkontatkt K3/21-22 entspricht. Wir verlassen uns also darauf, dass das Dreieck-Schütz nicht "klebt".
Besser oder sicherer wäre ein zusätzlicher Quittierkontakt an einer weiteren Eingangsklemme, um einem Kurzschluss vorzubeugen, falls das Dreieckschütz klemmt.

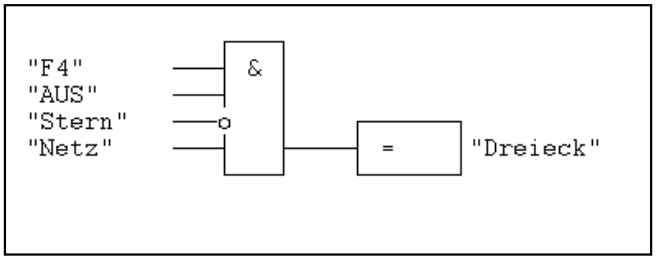
Netzwerk 2: Netzschütz

Die "StartSt"-Abfrage kommt aus der Schützensteuerung und ist nicht unbedingt erforderlich. Das Sternschütz schaltet das Netzschütz ein, das sich dann selbst hält. sofern die Eingänge für den Motorschutzschalter und den Austaster (Öffner) eine logische 1 liefern.



Netzwerk 3: Dreieckschütz

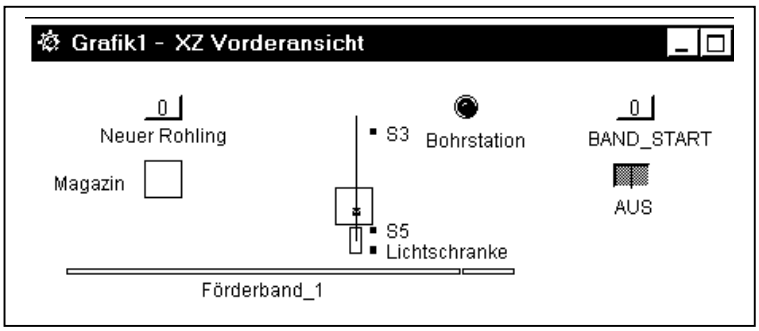
Das Sternschütz fällt im Netzwerk 1 ab, wenn der Taster <DreieckT> geöffnet wird.
Wenn der Ausgang "Stern" 0-Signal führt, kann das Dreieckschütz einschalten.
Es fehlt hier allerdings die Kontrolle, ob das Sternschütz auch tatsächlich abgefallen ist! Besser: eine zusätzliche Quittierabgfrage über einen Eingangskontakt.



Projekt 8: Bohrstation

Schwerpunkte: Anwendung von SR; Merker

Aufgabe: Auf einem Förderband werden Rohlinge zu einer Bohrstation gebracht. Nach dem Einschalten des Bandes kann ein Rohling aus dem Magazin geholt werden. Eine Lichtschranke erfasst die Bohrposition des Rohlings. Das Band stoppt und der Bohrer wird abgesenkt. Nach dem Erreichen der unteren Bohrerposition hebt sich der Bohrer automatisch bis zum oberen Anschlag.
Nun kann der bearbeitete Rohling auf dem Band weiterbefördert werden; das Band läuft selbständig wieder an.
Der Bandvorschub kann über den <AUS>-Taster gestoppt werden.



Die „Anlage“ und die Symboltabelle sind bereits vorbereitet worden.

Symbolische Adresse	Operand	Datentyp	Kommentar
LEDBOHRS	A 0.0	BOOL	Anzeige LED Bohrstation gefüllt
BAND	A 0.1	BOOL	Förderband eingeschaltet
Magazin	A 0.2	BOOL	wirft Rohlinge aus
SENKEN	A 0.3	BOOL	Bohrstation
HEBEN	A 0.4	BOOL	Bohrstation
ROHLING	E 0.0	BOOL	Befehl an Spender; Schließer
BAND_AUS	E 0.1	BOOL	BAND_AUS; Öffner
BANDSTRT	E 0.2	BOOL	BAND-START, Taster; Schließer
BK_oben	E 0.3	BOOL	Bohrkopf Endlage oben; S3; Öffner
BK_unten	E 0.5	BOOL	Bohrkopf Endlage unten; S5; Öffner
Lischr	E 0.6	BOOL	Lichtschranke hat "1", wenn Bohrstation erreicht
BOHRMERK	M 1.2	BOOL	merkt sich den Bohrvorgang

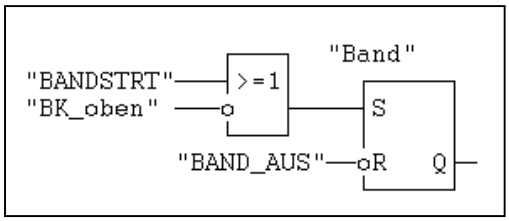
Theorie:
Dieses Beispiel gibt Einblick in die Verriegelungs- und Verknüpfungsproblematik.
Was heißt das? Nun, eine relativ einfache Abfolge von kleinen Schritten führt leicht zu Problemen, wenn man z. B. - wie hier - einen Vorgang (Transport eines Gutes) unterbrechen muss und diesen Vorgang dann wieder weiter führen soll. (Sie werden sehen, was damit gemeint ist).

Zur Praxis mit TrySim:

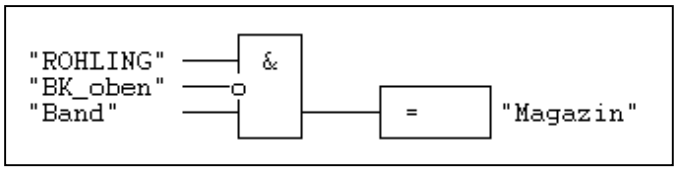
Sie bekommen von Projekt zu Projekt weniger Hilfe bei (hoffentlich!) bekannten Details. Falls Sie doch noch mehr Anleitung benötigen, blättern Sie bitte zu den vorherigen Projekten zurück.

- Starten Sie bitte das Programm TrySim.
- Öffnen Sie für die Projektierung im Ordner **<Vorlagen>** das Projekt **<Bohr_V>**, legen sich eine Kopie in Ihrem Ordner an und
- klicken Sie das **<FUP>**-Icon an, um in FUP zu programmieren. Klicken Sie mit <LM> in das freie Editierfenster.
- Klicken Sie auf das Icon Nr. 17 **(SR)**. (Bitte aufpassen: Nicht mit RS verwechseln)

Beginnen Sie die Programmierung in ganz kleinen Schritten:
Was soll zuerst passieren? Das Förderband ist über einen Taster (Schließer <BAND_START> = symbolisch Eingang „BANDSTRT“) einzuschalten. Solange der Taster <AUS> (Öffner, symbolischer Eingang „BAND_AUS“) nicht gedrückt wird, soll der Bandmotor (SPS-Ausgang "BAND" = A 0.1) erst einmal laufen.
Betrachten oder bauen wir dazu das **Netzwerk 1**: Der Ausgang "BAND" soll über ein SR-Flip-Flop angesteuert werden. Zum Setzen reicht erst einmal die "1"-Abfrage vom Eingang "Band_Start". Rückgesetzt wird der Antrieb, wenn Taster <AUS> gedrückt wird, d.h., dann liegt eine logische "0" am Eingang "R" des Flip-Flops an. Umgekehrt heißt das, wenn dieser Taster nicht gedrückt wird, liegt immer eine logische 1 an. Da aber ein SR-FF mit einer „1“ zurückgesetzt wird, muss die <AUS>-Taster-Abfrage negiert werden.
Der vorstehende Schritt kann schon einmal allein getestet werden.



Netzwerk 1

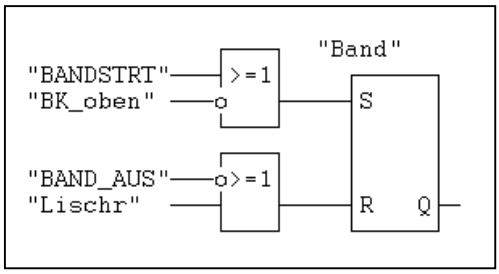


Netzwerk 2

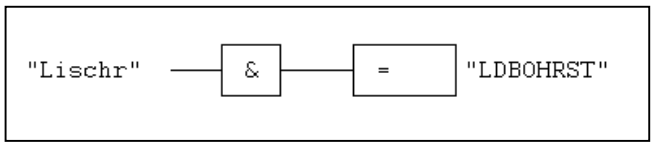
Nach dem ersten Erfolg soll jetzt (im **Netzwerk 2**) ein Rohling auf das Band gelegt und transportiert werden. (Dieser Vorgang gilt natürlich nur für die Simulation; in der Praxis könnte man sich eine Ventilsteuerung für ein Magazin vorstellen).

Unter der Bedingung, dass das Band eingeschaltet ist, kann ein neuer Rohling (oder auch mehrere ?) dem Magazin entnommen werden. Dabei muss der Bohrkopf oben sein, damit der Rohling nicht seitlich dagegen fährt und den Bohrer abbricht. Achten Sie bitte darauf: "BK_oben" (S3) ist ein Öffner!

Beim Test fährt dieser Rohling nun das ganze Band entlang und verschwindet rechts im Vernichter. Da das so nicht geplant ist, muss das Band (Netzwerk 1) also nicht nur vom "AUS"-Taster, sondern auch von der Lichtschranke an der Bohrstation abgeschaltet werden. Die Lichtschranke hat einen Schließerkontakt, d.h., wenn der Rohling dort ankommt, schließt der Kontakt. Diese logische "1" setzt den Antrieb zurück. Also ist diese Abfrage mit der "AUS"-Abfrage ODER-verknüpft.



Ergänzung im Netzwerk 1

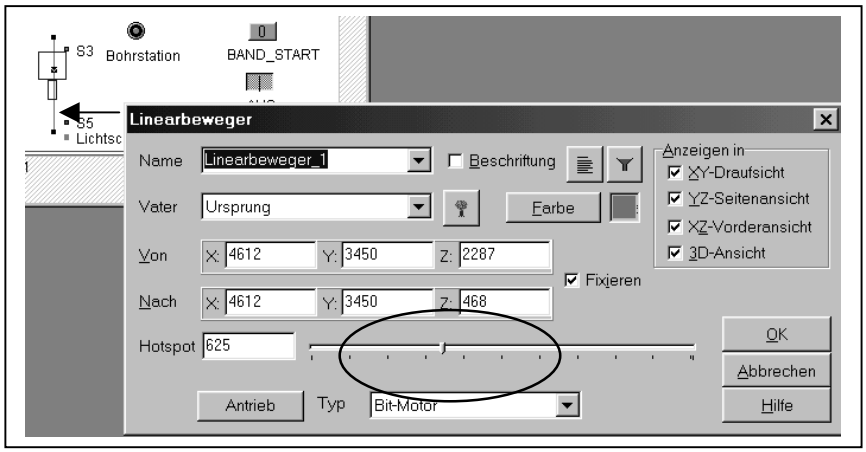


Netzwerk 3

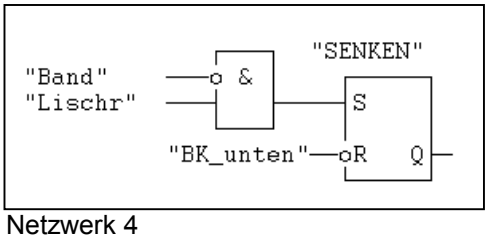
Testen Sie auch diesen Schritt.

Solange der Rohling in der Bohrstation liegt, wird im **Netzwerk 3** von der Lichtschranke („LiSchr“ = E 0.6) die LED "LDBOHRST" eingeschaltet, die als TrySim-Element "Blinker" ausgewählt wurde.
Gleichzeitig wird im **Netzwerk 4** durch die Lichtschranke der Bohrer gesenkt. Dazu muss das Band allerdings abgeschaltet sein. Dieses Senken wird wieder durch ein SR-FF gesteuert. Der Senkvorgang wird beendet, wenn der untere Endtaster S5 erreicht ist. Aus Sicherheitsgründen sind diese Endtaster Öffner, d. h., in der unteren Bohrer-Position liegt eine "0" an.
Zum Abschalten wird aber wieder eine "1" benötigt. Deshalb muss dieser Eingang negiert werden.

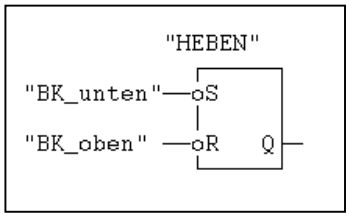
Tipp zum TrySim-Linearbeweger, der als Antrieb für die Bohrspindel dient:
Der "Bohrkopf" ist an einem Linearbeweger fixiert. Die Operanden für "Senken" und "Heben" sind - wie bei den anderen Elementen auch - im ausgeschalteten Zustand mit der Maus zu erfragen, wenn der Cursor auf den schmalen grauen Strich (am Bohrkopf, aber außerhalb der Kästen) gesetzt wird. Falls Sie den Vorgang "Senken" oder "Heben" wegen eines Irrtums beim Programmieren nicht wunschgemäß durch die Endtaster stoppen können, fährt der Bohrkopf sehr realistisch bis an das mechanische Ende (und frisst sich dort fest). Sie müssten eigentlich ein Notprogramm schreiben, um den Bohrkopf in den Arbeitsbereich zurück zu bekommen. In der Simulation ist der Schaden erfreulicherweise geringer - Sie können den "Schlitten" "mechanisch" verschieben: Klicken Sie bitte mit <rM> auf den Strich. Es öffnet sich das Editierfenster des Linearbewegers. Dort können Sie mit dem Schieber die Position des Bohrkopfes verschieben.



Der Endtaster S5 gibt im **Netzwerk 5** gleichzeitig den Setz-Befehl, den Bohrkopf zu heben.

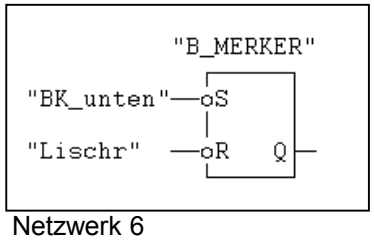


Netzwerk 4

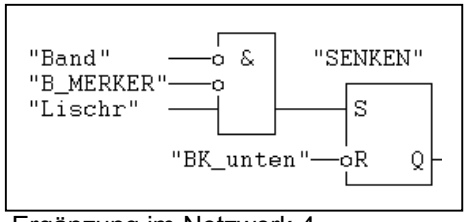


Netzwerk 5

Nun ergibt sich ein Problem, auf das wir eingangs schon hindeuteten: Sowie der untere Endtaster wieder frei ist, ist die Setzbedingung für "SENKEN" wieder gegeben (Bandantrieb steht, Lichtschranke liefert "1"). Daraus ergibt sich in der Simulation eine lustige Zappelei, weil der Bohrkopf nicht weiß, ob er nun rauf- oder runterfahren soll. Eine Möglichkeit dieses Chaos zu beenden, besteht in einer zusätzlichen Verriegelung: Wenn der Bohrer unten ankommt, wird im **Netzwerk 6** ein Merker gesetzt.



Netzwerk 6



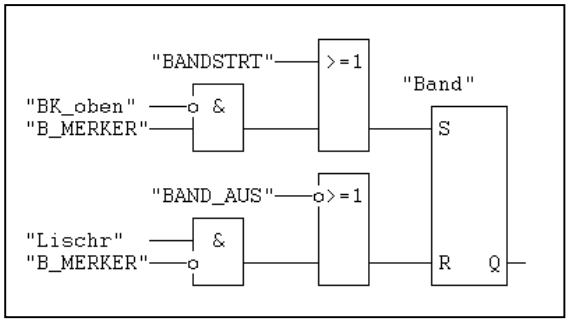
Ergänzung im Netzwerk 4

Dieser Merker hat ausschließlich die Aufgabe, sich dauerhaft zu merken, dass der Bohrer schon einmal unten war.

Der Merker wird im Netzwerk 4 zusätzlich abgefragt und verriegelt den "S"-Befehl. Nur wenn er nicht gesetzt ist, kann dort der Setzbefehl für "SENKEN" wirksam werden. Wenn also "Heben" eingeleitet

wird, hat "BAND" Pegel „0“ und "Lichtschranke" „1“, aber "B_Merker" führt „1“. Der Bohrkopf fährt ungestört hoch bis zum Endtaster S3.

Das "Heben" wird im Netzwerk 5 mit einer „0“ am Endtaster S3 beendet, d.h., für den Rücksetzbefehl muss das Signal negiert werden.
Erst wenn der Bohrkopf oben angekommen ist, darf das Band weiterlaufen. Dafür darf aber kein neues Netzwerk programmiert werden! Warum denn nicht?
Denken Sie bitte an die beschriebene Problematik bei der Verwechslung von SR und RS.
Dasselbe gilt auch für ganze Netzwerke. Würde in einem Netzwerk für einen Ausgang das Verknüpfungsergebnis (VKE) "1" sein, in einem folgenden Netzwerk aber andere Bedingungen für den selben Ausgang zu "0" führen, würde dieser Ausgang niemals auf „1“ gesetzt werden, es sei denn, das folgende Netzwerk hätte auch (oder nur) ein VKE = „1“.
Also: der Bandmotor muss im alten Netzwerk 1 wieder gestartet werden.
Auch hier muss das Programm erkennen, dass bereits gebohrt wurde und der Bohrer wieder oben angekommen ist. Deshalb ist als vollständige Lösung am S-Eingang eine ODER-Verzweigung nötig: Gestartet wird entweder über den Start-Taster **oder** über die Weiterschaltbedingung (der Bohrmerker führt „1“-Signal **und** der Bohrkopf ist oben, d.h., S3 führt „0“-Signal).



vollständiges Netzwerk 1

Bisher hatte die Lichtschranke mit 1-Signal das Band abgeschaltet. Nach dem Bohren liegt das Werkstück aber immer noch am alten Platz, so dass die „1“ nach wie vor ein Einschalten verhindern würde. Also koppeln wir den Rücksetzbefehl der Lichtschranke mit dem Bohrmerker: Nur wenn der Merker eine „0“ führt (nur bei der Bandzufuhr vor dem Bohren) stoppt die Lichtschranke das Band. Nach Aufheben dieser Verriegelung kann das Band wieder starten und das Werkstück weiter transportieren.

Nun muss die Anlage wieder bereit sein, die Prozedur zu wiederholen. Der Bohrmerker im Netzwerk 5 muss noch zurückgesetzt werden. Dazu verwenden wir einfach die Lichtschranke, die bei weiterfahrendem Werkstück „0“-Pegel führt.

Vgl. TrySim-Projekt <Bohr> im Verzeichnis <Lösungen>.

Kritische Würdigung:
Diese "Anlage" ist bei weitem nicht praxisgerecht. Es gibt z. B. kein Not-Aus, nach dem Ausschalten ist u.U. der Bohrkopf nicht in der Ausgangsposition und muss "von Hand" (im Editierfenster) wieder dorthin geschoben werden, usw. Hier sollte nur die Notwendigkeit von Merkerfunktionen aufgezeigt werden. Bessere Lösungen folgen in den nächsten Projekten.

Sie haben gelernt:

Alle Ein- und Ausschaltverknüpfungen für einen Ausgangsoperanden - und seinen sie noch so zeitversetzt- müssen in nur **einem** Netzwerk diesem Operanden zugewiesen werden.

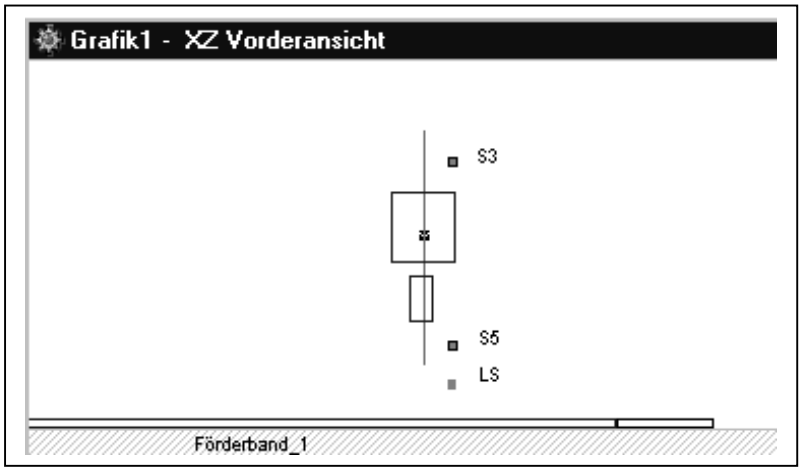
Für einige Verknüpfungen werden "Gedächtnisfunktionen" benötigt. Diese erreicht man mit einem SR-gespeicherten Merkeroperanden.

Projekt 9: Erläuterungen zur Verwendung von Öffnern

Schwerpunkte: Anwendung von SR; Merker; vgl. Trysim-Projekt <LinBew> im Verzeichnis <Lösungen>.

Aufgabe: Hier geht es nicht um ein neues Projekt, sondern um einige Details bei der Verwendung des TrySim-Elements <Öffner>.
Vielleicht sind Sie bei dem letzten Projekt auf unerklärliche Schaltzustände eines SR-FlipFlops gestoßen (falls Sie sich nicht an die Anleitung gehalten haben). Es geht jetzt darum, den Zustand eines einzigen Netzwerkes – eines SR-FFs – zu analysieren.

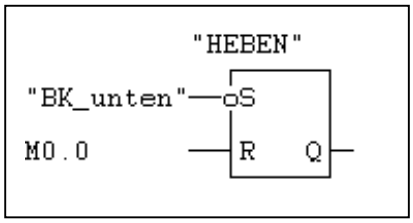
Aus dem letzten Projekt wurde nur der Linearberweger des Bohrkopfes übernommen.



Der Bohrkopf wurde bewusst so platziert, dass er zwischen den Endtastern „startet".

Außerdem wurden alle Netzwerke gelöscht, bis auf den Antrieb für den Hebevorgang.

Netzwerk (alt) 5: HEBEN
Der untere Endschalter <S5> gibt den Befehl, den Bohrkopf zu heben, bis er oben angekommen ist.



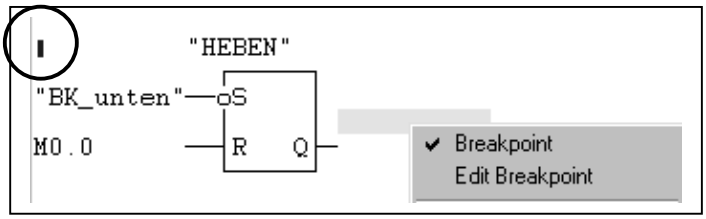
Um das Problem zu verdeutlichen, welches entstehen kann, wurde der Rücksetzeingang ganz beziehungslos dem Merkeroperanden M 0.0 zugeordnet. Dieser Merker führt immer „0". d. h. er wird weder durch die Anlage noch durch das Programm auf „1" gesetzt. Damit kann <R> nie wirksam werden.
Wenn Sie dieses Programm geladen haben und sich im ausgeschalteten Zustand die Spannungspegel der Endtaster ansehen (evtl. mit dem Lupe-Icon vergrößern), erkennen Sie an Hand der grauen Färbung die „0"-Pegel. Wenn Sie nun mit <rM> die Editierfenster der Endtaster öffnen, erkennen Sie, dass <Öffner> als Typ ausgewählt wurden.
Sie würden doch bestimmt rot gefärbte Tasterquadrate erwarten („1"-Pegel), weil unbetätigte Öffner – wie vorher so eindringlich gelernt – eine „1" liefern. Keine Angst, diese Logik wird jetzt nicht wieder auf den Kopf gestellt. Die Ursache liegt woanders:
Erst beim Programmstart ordnet TrySim die gewünschten Pegel zu, d.h., erst nach dem Programmstart wird der „Kontakt" zum Öffner. In der Ausgangsphase liefern alle Operanden „0".

Zum weiteren Verständnis für einige Grenzsituationen müssen Sie wissen, in welcher Reihenfolge TrySim die Anlage mit dem Programm verknüpft. In der jetzigen Version ruft TrySim zuerst das Programm auf und dann (im allerersten Programmzyklus) werden die Öffner-Pegel in der Anlage gesetzt.

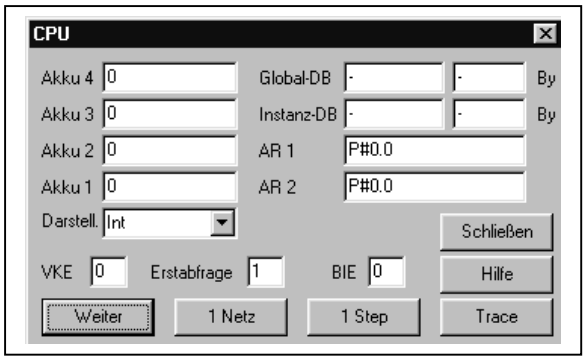
Wofür ist das Wissen wichtig? Es soll ein Fehler analysiert werden.
Betrachten Sie bitte – immer noch im ausgeschalteten Zustand (!) – das Netzwerk.
Der Bohrkopf soll gehoben werden, wenn er unten am Endtaster <S5> („BK_unten“) angekommen ist. Dieser Öffner wird dann betätigt und liefert dann „0“. Diese „0“ setzt also das SR-FlipFlop <HEBEN>. In der dargestellten Startposition ist zu erwarten, dass beide Endtaster eine „1“ liefern. Damit darf <HEBEN> nicht einschalten. Wenn Sie nun aber die Anlage und das Netzwerk mit dem Auge-Icon aktivieren, läuft der Bohrkopf dennoch nach oben und das SR-FF ist (rot) gesetzt.

Die Erklärung ist gemäß vorstehend beschriebener Zusammenhänge eigentlich ganz einfach: Da TrySim mit der Programmabfrage beginnt, liegt zuerst noch eine „0“ an <S5> an. Dieser extrem kurze Zustand zu Beginn der ersten Zyklusabfrage reicht aus, um den programmgerechten Einschaltpegel zu liefern. Unmittelbar danach werden die Öffner „präpariert“, d.h., sie liefern die „1“. Dann ist es aber schon zu spät.
Sichtbar ist bei der Analyse nur die wunschgemäße Situation, dass an <S5> (der gar nicht betätigt wurde) eine logische „1“ anliegt und somit <HEBEN> eigentlich gar nicht gesetzt werden konnte.

Natürlich müssen Sie mir diesen „Spuk“ nicht glauben, sondern Sie haben trotz der kurzen Zykluszeit die Möglichkeit, diese These in aller Ruhe zu kontrollieren. Dazu können Sie mit <rM> im Editierfenster des Netzwerkes <Brakepoint> anklicken. Links oben zeigt sich dann ein kleiner blauer Strich und vor <Brakepoint> wird ein Häkchen gesetzt.



Mit diesem Befehl wird das Programm nur bis zu diesem Netzwerk abgearbeitet und endet hier. Wenn Sie jetzt mit dem Auge-Icon starten, wird das SR-FF <HEBEN> aktiv. In der Anlage sehen Sie deutlich, dass die Endtaster noch auf „0“ stehen. Wenn Sie dann in dem neu eingeblendeten CPU-Fenster auf <1 Step> drücken, sehen Sie, wie dann – anschließend – auch die Endtaster rot werden, die Öffner also richtig arbeiten.



Erst diese Situation wird im normalen Programm angezeigt. Kein Wunder, wenn Sie staunen, dass wie von Geisterhand der Bohrkopf nach oben fährt, obwohl sichtbar die „1“ an <S5> anliegt.

Schließen Sie bitte mit <Schließen> das vorstehende Fenster und entfernen das Häkchen vor <Brakepoint>.
Sie haben vielleicht schon beobachtet, dass nach obigem Testlauf die Icons <Anlage starten> und <Anlage stoppen> aktiv sind. Sie können / müssen nun ein Icon betätigen, um wieder zur normalen Programmbearbeitung zurück zu kommen.

Schlussfolgerungen:

Die hier künstlich geschaffene Situation ist zu vermeiden, indem definierte Rücksetzbedingungen programmiert werden. In <BOHR> ist der <R>-Operand ebenfalls ein Ö ffner, der im ersten Zyklus (noch) eine „0“ liefert, so dass über den invertierten Eingang das SR-FF zwangsweise zurückgesetzt wird. Deshalb tritt bei der richtig angeschlossenen Anlage dieses Phänomen gar nicht auf.

Allerdings sind in der Praxis Situationen denkbar, die der hier beschriebenen leicht ähneln können. Nicht durch Simulation, sondern durch verzögerte Signalbildung oder –weitergabe können sich zeitkritische Vorgänge ungünstig überlagern, so dass stabile Rücksetzbedingungen wichtig sind.

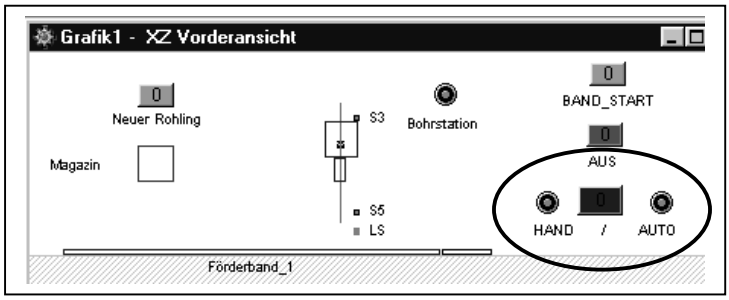
vgl. Trysim-Projekt <LinBew> im Verzeichnis <Lösungen>.

Projekt 10: <Bohrstation, automatischer Betrieb>

Schwerpunkte: Anwendung von SR, Merker, Verriegelungen

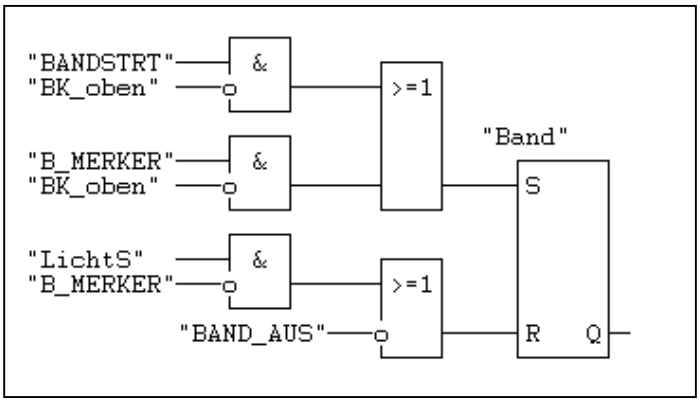
Aufgabe: Das von Ihnen erstellte Projekt 8, die Bohrstation, soll für einen Automatikbetrieb erweitert werden. Dafür muss in das Pult ein Schalter eingebaut werden. Bei nicht betätigtem Schalter fährt die Anlage im bisherigen Handbetrieb. Bei betätigtem Schalter soll - nachdem das Band eingeschaltet wurde – je Bohrzyklus ein Rohling bearbeitet werden. Wenn der Bohrkopf oben steht und die Lichtschranke einen freien Platz meldet, wird automatisch dem Magazin ein neuer Rohling entnommen. Zusätzlich sind noch einige Feinheiten zu ergänzen, die die Anlage praxisgerechter machen. Die NOT-AUS-Funktion ist erforderlich.

Zur Lösung mit TrySim:
Zuerst wird das Pult erweitert. Dazu öffnen Sie bitte bei aktivem Grafikfenster mit <LM> das Icon <Neues Element>. Wählen Sie dort unter <Bedienung> den Schalter aus, den Sie mit gedrückter <LM> in die Anlage ziehen. 2 LEDs für die Anzeige von Hand- bzw. Automatikbetrieb können Sie ebenso „einbauen“.



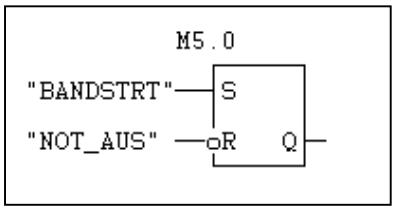
Zu bedenken ist, dass der Bohrkopf zu jeder Zeit abzuschalten sein muss. Deshalb soll der bisherige <AUS>-Taster jetzt die NOT-AUS-Funktion übernehmen. Dazu wird die symbolische Adresse in der Symboltabelle auch geändert. Vorher sollten Sie aber das SPS-Editierfenster aktiv klicken, um anschließend unter <Ansicht> das Häkchen vor <Symbolische Darstellung> zu entfernen. Wenn Sie sonst die symbolische Adresse ändern würden, könnte das Programm die alte symbolische Adresse nicht mehr zuordnen und Sie bekämen einen entsprechenden Verweis.
Öffnen Sie bitte mit dem Icon die Symboltabelle und ändern in Zeile 7 den Text in der Spalte <Symbol>. Anschließend müssen Sie mit <LM> in eine andere Zeile klicken, um die Änderung wirksam werden zu lassen. Sie können dann die Symboltabelle wieder schließen und zur symbolischen Darstellung zurück kehren.

Da der Bohrkopf jetzt bei NOT-AUS in jeder Position stehen bleiben kann, reicht es auch nicht aus, allein mit <BAND_START> das Förderband zu starten. Deshalb wird dieser Operand mit <S3> (= BK_oben) verknüpft.



Nach <NOT-AUS> und dessen Freigabe (!) darf die Anlage nicht wieder selbstständig anlaufen. Deshalb wird jetzt ein SR-FlipFlop benötigt, mit dem eine drahtbruchsichere Verriegelung ermöglicht wird.
Wählen Sie das Netzwerk 2 im Editor aus und dann über <Bearbeiten>|<Neues Netzwerk> ein leeres Netzwerk. Die Netzwerknummern passen sich automatisch an.

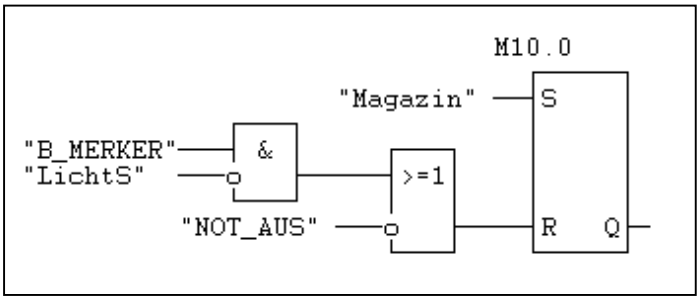
Neues Netzwerk 2: Not-Aus-Verriegelung



Mit dem <BAND_START> wird dieses Verriegelungs-FF gesetzt. Der geöffnete <NOT-AUS>-Taster setzt das FF zurück. Bei Drahtbruch liegt ebenfalls eine „0“ am <R>-Eingang an.

Netzwerk 4 (!): Nur 1 Rohling

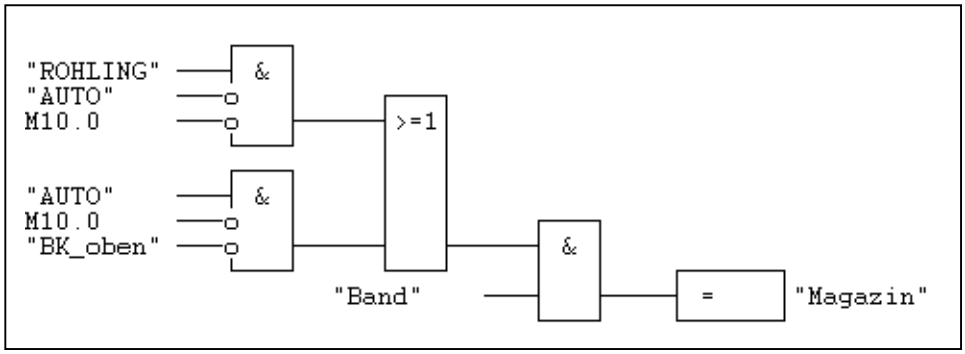
Vielleicht haben Sie beim Anlagentest den Taster <Neuer Rohling> schon einmal längere Zeit gedrückt gehalten und haben festgestellt, dass der neue Rohling – wie geplant - erst dann dem Magazin entnommen wurde, wenn der Bohrkopf oben angekommen war. Warum wird dann aber nur ein Rohling freigegeben? Dieser Befehl müsste theoretisch in jedem Zyklus erneut aktiv werden. Diese rasche Wiederholung wird jedoch von TrySim „verhindert“. Drücken Sie bitte zum Verständnis bei ausgeschalteter Anlage mit <rM> die Umrandung des Magazins. Es öffnet sich das zugehörige Editierfenster. Dort erkennen Sie, dass die Taktrate, d.h. die Verzögerung zwischen 2 Dynamiks, auf 10 s voreingestellt ist. Erst diese interne Verzögerung verhindert, dass mehrere Rohlinge in rascher Folge erzeugt werden.
Ändern Sie bitte diesen Wert auf 1s und starten Sie die Anlage. Wenn Sie jetzt den Taster <neuer Rohling> gedrückt halten, sehen Sie den Unterschied. Es soll jetzt Ihre Aufgabe sein, je Bohrtakt dem Magazin wirklich nur einen Rohling zu entnehmen – egal, wie lange Sie drücken.
Fügen Sie bitte bei aktivem Netzwerk 4 wieder mit <Bearbeiten>|<Neues Netzwerk> ein leeres Netzwerk ein und programmieren Sie ein „Gedächtnis“ für den allerersten Zyklus, in dem ein Rohling angefordert wird. Dieser Merker soll verhindern, dass mehrere Rohlinge während eines Bohrzyklusses erzeugt werden.



Beim ersten „1“-Signal vom <Magazin> wird der Merker gesetzt. Nachdem der Rohling die Bohrstation verlassen hat und die Lichtschranke wieder „0“ führt, wird diese Merkerinformation nicht mehr benötigt und kann gelöscht werden. Allerdings führt <LS> auch „0“, wenn noch gar kein Rohling in der Bohrstation liegt. Deshalb muss „LichtS“ mit „B_MERKER“ verriegelt werden. (B_MERKER“ wird erst aktiviert, wenn ein Rohling gebohrt wird). Zusätzlich wird M10.0 mit <NOT-AUS> zurückgesetzt.

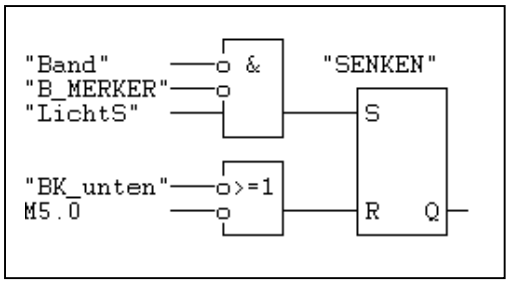
Netzwerk 3 (!): Magazinfreigabe / Rohling

Zum Verständnis wurde Netzwerk 4 zuerst erläutert. Logisch gehört es aber hinter Netzwerk 3.
Das alte <Magazin>-Netzwerk wird jetzt angepasst.



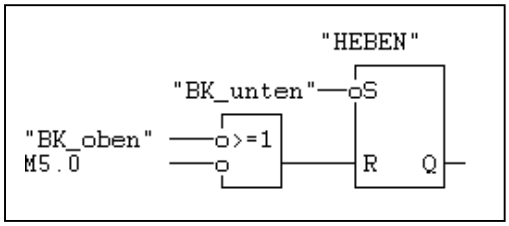
Der Taster <Neuer Rohling> soll nur im <HAND>-Betrieb wirksam sein, d.h. wenn „AUTO“ eine „0“ führt. Außerdem darf der Rohling-Merker noch nicht gesetzt worden sein.
Im „AUTO“-Betrieb kann auch nur einmal ein Rohling aktiviert werden (wenn der Rohling-Merker noch nicht gesetzt wurde). Gestartet wird der Vorgang, wenn der Bohrkopf oben angekommen ist.

Neues Netzwerk 6: BOHRER senken



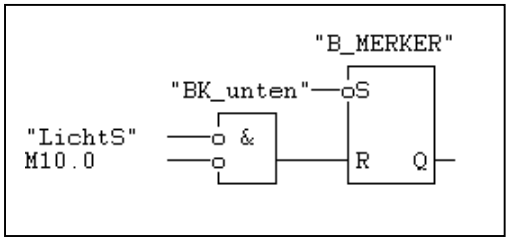
Die zusätzliche Abfrage von „M5.0“ ermöglicht das dauerhafte Ausschalten von „SENKEN“.
Würde man über <NOT-AUS> abschalten, könnte nach dem Entriegeln „SENKEN“ wieder von allein starten, da die Setzbedingungen noch erhalten geblieben sind. Mit <BAND_START> könnte „M5.0“ wieder entriegelt werden und „SENKEN“ fortgesetzt werden, ohne dass das Band anläuft (weil das von „B_MERKER“ verhindert wird).

Netzwerk 7: BOHRER heben



In dieser Version wird auch das „HEBEN“ dauerhaft unterbunden. Allerdings kann ohne eine zusätzliche Handsteuerung „Heben“ nicht fortgesetzt werden. Benutzen Sie dafür die Editiermaske des Linearantriebs.

Netzwerk 8: Bohrmerker

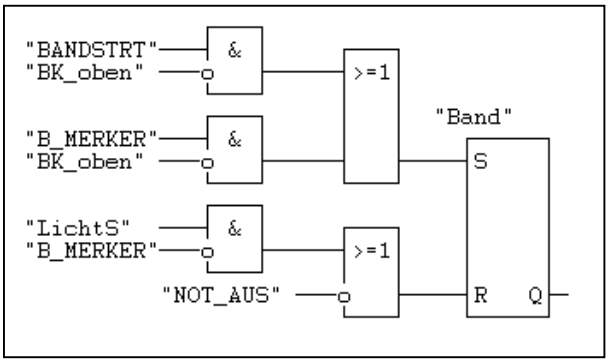


Da im Netzwerk 6 der „B_MERKER“ und die „LichtS“ zum Rücksetzen benötigt wurden, wird sicherheitshalber der „B_MERKER“ erst dann ausgeschaltet, wenn M10.0 schon „0“ führt.
Sonst könnte es passieren, dass „LichtS“ allein den „B_MERKER“ auf „0“ setzt und dann im nächsten Zyklus M10.0 nicht rückzusetzen ist.

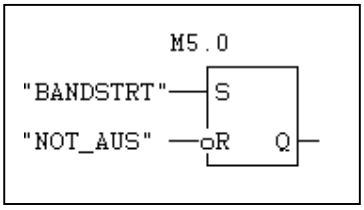
Weitere Änderungen sind nicht geplant.. Vgl. TrySim <Lösungen>|<Bohr_AUTO>.

Es folgt noch einmal die Zusammenstellung aller Netzwerke.

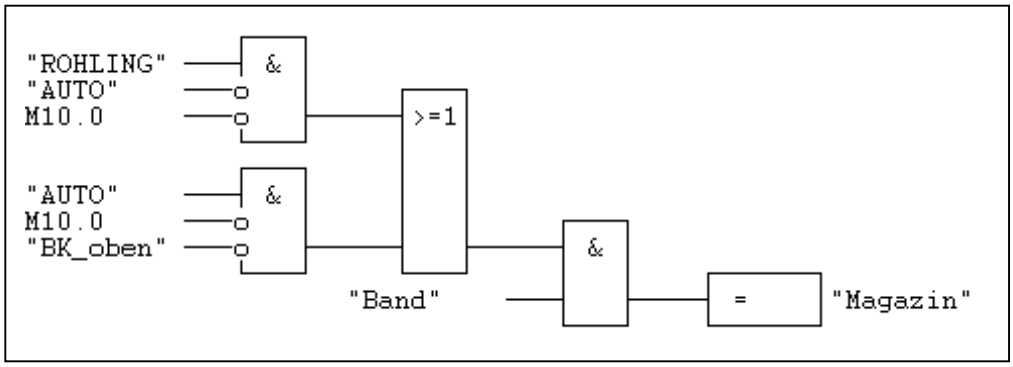
Netzwerk 1: Förderband



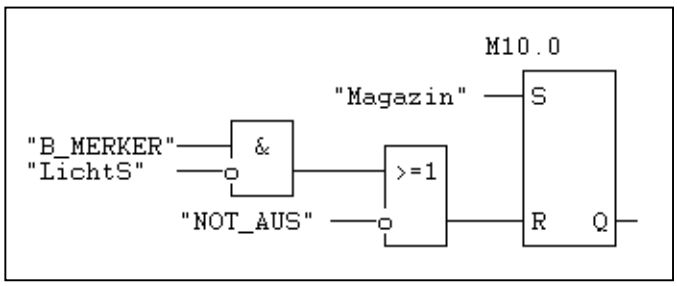
Netzwerk 2: Not-Aus-Verriegelung



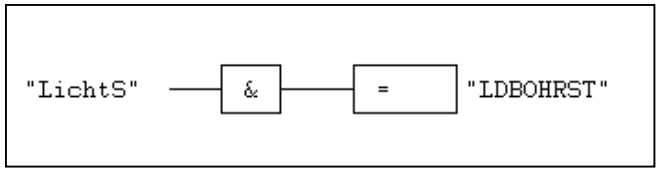
Netzwerk 3: Neuer Rohling



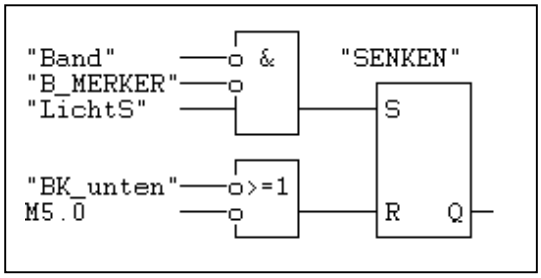
Netzwerk 4: Nur ein Rohling



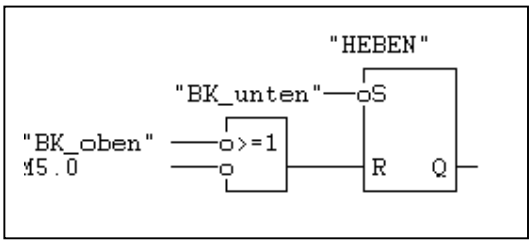
Netzwerk 5: Bohrposition
Die LED zeigt an, wenn der Rohling die Bohrposition erreicht hat.



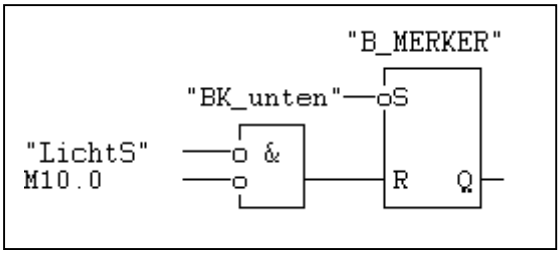
Netzwerk 6: Bohrstation erreicht, SENKEN
Nachdem das Band steht, wird durch die Lichtschranke die Bohrstation gesenkt, bis der Bohrer unten angekommen ist.



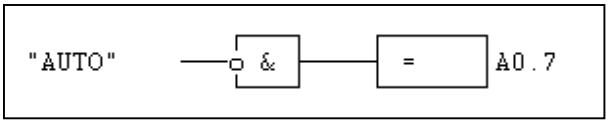
Netzwerk 7: Bohren beendet, HEBEN
Der untere Endschalter gibt den Befehl, den Bohrkopf zu heben, bis er oben angekommen ist.



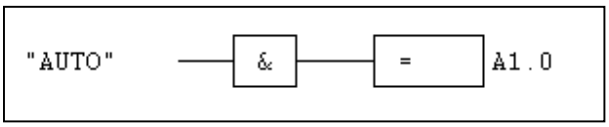
Netzwerk 8: Bohrmerker



Netzwerk 9: LED Handbetrieb



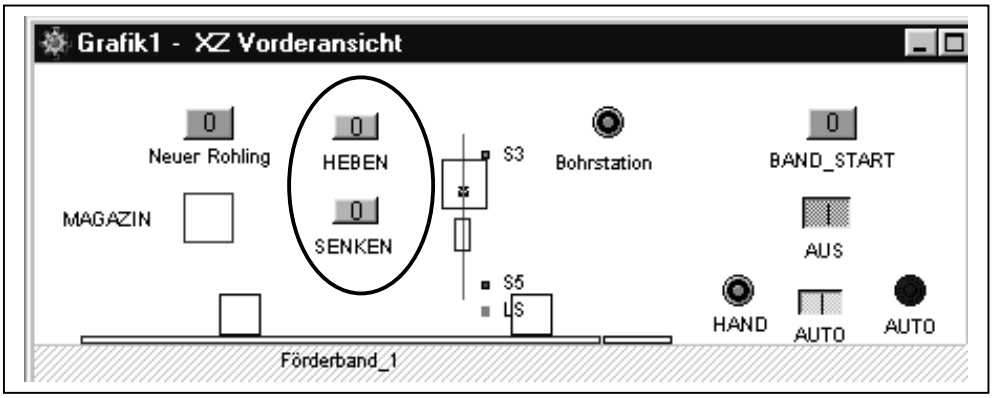
Netzwerk 10: LED Automatikbetrieb



Projekt 11: Bohrstation 2

Schwerpunkte: Flankenauswertung,
Der Aufbau und die Aufgabe entspricht dem Projekt 10, bzw. <Bohr_AUTO>. Details: siehe dort

Sie werden nach <NOT-AUS> feststellen, dass der Bohrkopf nicht mehr in seine Ausgangslage fährt und somit die Anlage nicht mehr steuerbar ist. Das ist natürlich nicht praxisgerecht. Deshalb ist eine Möglichkeit zu schaffen, den Bohrkopf nach einem NOT-AUS-Befehl (und nur dann) von Hand entweder in die obere Endlage zu bringen oder den Bohrvorgang fortzusetzen. Dazu werden zusätzlich 2 Taster in das Pult „eingebaut“. Es empfiehlt sich, zuerst eine Kopie von Ihrem vorherigen Projekt zu erstellen und mit dieser umbenannten Kopie weiter zu arbeiten.



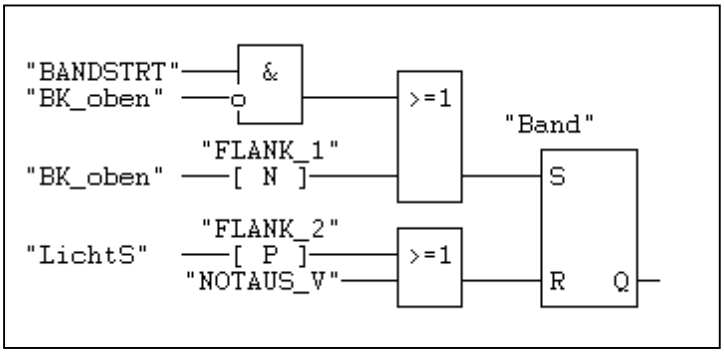
Zusätzlich soll eine Lösungsvariante der bisherigen Aufgabenstellung erarbeitet werden. Durch die schon bekannte Flankenauswertung kann das Programm z.T. vereinfacht werden.
Zur Erinnerung:
Durch die Zwischenschaltung der Flankenauswertung wird erreicht, dass ein Eingangssignal nur einmal - nämlich beim Übergang von "0" auf "1" (--[P]--) bzw. von "1" auf "0" (--[N]--) - wirksam wird. Der zugehörige Merkeroperand kann beliebig gewählt werden, darf jedoch nirgends sonst in der Steuerung benutzt werden.

Symbolische Adresse	Operand	Datentyp	Kommentar
LDBOHRST	A 0.0	BOOL	Anzeige Bohrstation
Band	A 0.1	BOOL	Förderband eingeschaltet
Magazin	A 0.2	BOOL	wirft Rohlinge aus
SENKEN	A 0.3	BOOL	Bohrstation
HEBEN	A 0.4	BOOL	Bohrstation
LED_HAND	A 0.7	BOOL	LED Handsteuerung
LED_AUTO	A 1.0	BOOL	LED AutomatikSteuerung
ROHLING	E 0.0	BOOL	Befehl an Spender; Schließer
NOT_AUS	E 0.1	BOOL	ALLES AUS; Öffner
BANDSTRT	E 0.2	BOOL	Taster; Schließer
BK_oben	E 0.3	BOOL	Bohrkopf Endlage oben; S3; Öffner
AUTO	E 0.4	BOOL	Umschalter Hand/Auto = 1
BK_unten	E 0.5	BOOL	Endtaster Bohrkopf unten; S5; Öffner
LichtS	E 0.6	BOOL	LS hat 1, wenn Bohrstation erreicht
T_HEBEN	E 0.7	BOOL	Taster Bohrkopf heben von Hand
T_SENK	E 1.0	BOOL	Taster Bohrer senken von Hand
MAG_VERR	M 4.0	BOOL	verhindert 2. Rohling
FLANK_1	M 20.0	BOOL	Flankenauswertung
FLANK_2	M 20.1	BOOL	Flankenauswertung
FLANK_3	M 20.2	BOOL	Flankenauswertung
FLANK_4	M 20.3	BOOL	Flankenauswertung
NOTAUS_V	M 50.0	BOOL	Not-Aus-Verriegelung

Symboltabelle

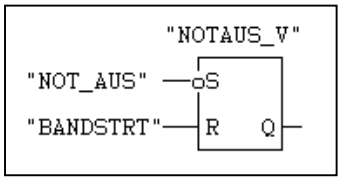
Die neuen Taster und einige Merker sind in die Symboltabelle aufzunehmen, damit konsequent mit den symbolischen Adressen gearbeitet werden kann.

Netzwerk 1: Förderband (Einführung der Flankensteuerung)



Das Band wird durch den Taster <BAND_START> gesetzt, wenn der Bohrkopf oben steht. („BK_oben“ liefert als betätigter Öffner dann eine „0“). Es wird ausgeschaltet, wenn der Rohling die Lichtschranke erreicht.
Wenn der Bohrkopf nach dem Bohren wieder oben angekommen ist, wechselt „BK_oben“ von „1“ auf „0“. Diese negative Flanke wird ausgewertet und das Band fährt wieder an. Der Merker, der festhält, ob der Bohrvorgang schon beendet wurde, ist hier nicht mehr erforderlich.
Beim zweiten Bandstart (nach dem Bohrvorgang) hat die Lichtschranke keine Ausschaltfunktion, weil sie dauerhaft „1“ führt und somit kein erneuter „0“/“1“-Übergang entsteht.
Das Band bleibt so lange eingeschaltet, bis es mit dem <AUS>-Taster ausgeschaltet wird. Beim Steuern des Bohrkopfes von Hand – nach NOT-AUS – darf das Band nicht selbständig anlaufen, auch wenn der obere Endtaster S3 („BK_oben“) eine negative Flanke erzeugt. Deshalb wird – im nächsten Netzwerk – der NOT-AUS-Befehl gespeichert. Dieser verhindert oben, dass das Band anläuft, solange dieser NOT-AUS-Merker nicht zurückgesetzt wurde.

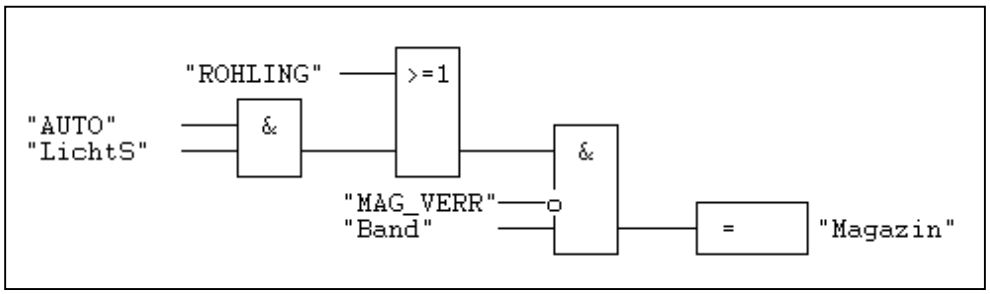
Netzwerk 2: NOT-AUS-Verriegelung



Wenn <AUS> gedrückt wird, soll alles stehen bleiben. Der Taster hat praktisch NOT-AUS-Funktion.
Erst wenn der <AUS>-Taster bzw. in der Praxis der NOT-AUS-Taster entriegelt wäre, könnte mit <BAND_START> diese Merkerverriegelung aufgehoben werden.

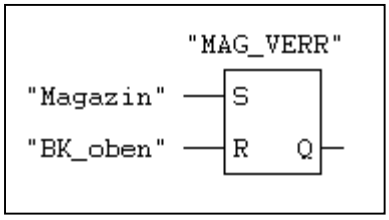
Im Vergleich mit der vorherigen Bohranlage wurde hier die Verriegelungslogik umgekehrt. Dieser Merker wird gesetzt, wenn <AUS> betätigt wird. Es soll nur gezeigt werden, dass es auch „anders herum“ geht. Wichtig ist nur, dass bei Drahtbruch oder beim AUS-Befehl eine „0“ ausgewertet wird.

Netzwerk 3 Neuer Rohling



Wenn das Band läuft, kann von Hand ein neuer Rohling über das Magazin auf das Band gelegt werden. Ein weiterer Rohling wird durch das folgende Netzwerk 4 verhindert, da dann „MAG_VERR“ eine „1“ liefert.
Im Automatikbetrieb wird bei leerem Band – also zu Beginn – durch Druck auf <Neuer Rohling> wie beim Einzelbetrieb der erste Rohling ausgeworfen. Danach wird von der Lichtschranke ein neuer Rohling angefordert. Eine erneute Betätigung von <Neuer Rohling> bleibt wirkungslos.

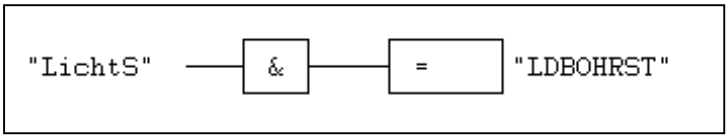
Netzwerk 4: Nur 1 Rohling bei Einzelbetrieb



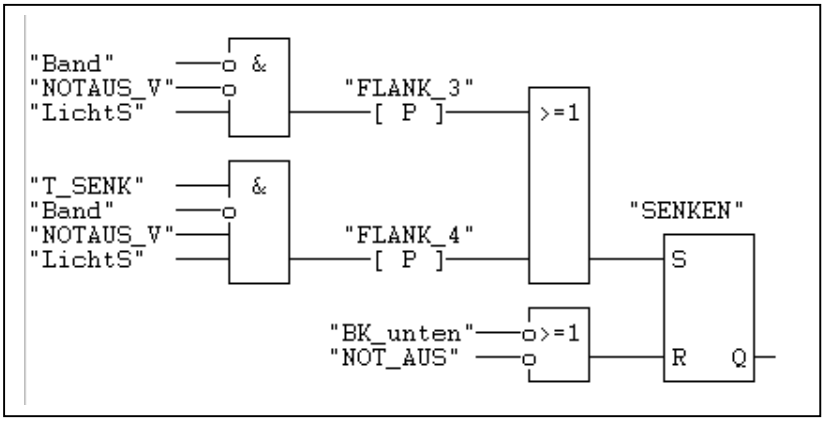
Durch die erste Rohling-Anforderung erhält das <MAGAZIN> eine „1“. Damit wird auch „MAG_VERR“ dauerhaft auf „1“ gesetzt. Dadurch wird in vorstehendem Netzwerk 3 verhindert, dass „Magazin“ noch einmal aktiviert werden kann.

Wenn der Bohrkopf runter fährt, wird diese Information nicht mehr benötigt, weil das ausgeschaltete Band ebenfalls eine „1“ am „Magazin“ verhindert.

Netzwerk 5: Bohrposition (unverändertes Netzwerk, aber neue Netzwerknummer)
Die LED zeigt an, wenn der Rohling die Bohrposition erreicht hat.



Netzwerk 6: Bohrer senken (Einführung der Flankensteuerung)

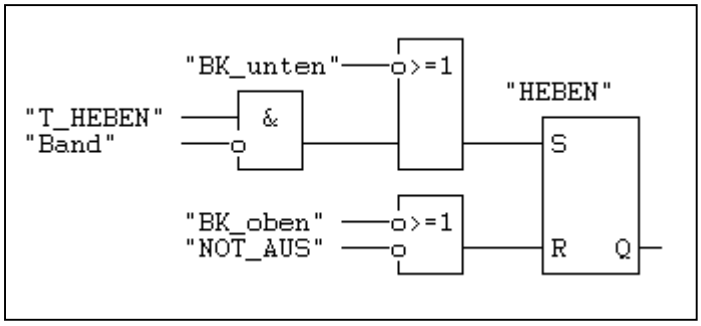


Der Bohrkopf kann gesenkt werden, wenn

1) im Normalbetrieb („NOTAUS_V“ ist nicht gesetzt) das Band steht und die Lichtschranke durch den Rohling auf „1“ gewechselt hat, oder wenn
2) nach NOT-AUS („NOT-AUS_V“ = „1“) das Band ebenfalls steht und die Lichtschranke durch den Rohling auf „1“ gewechselt hat und zusätzlich der Taster <SENKEN> (=“T_SENK“) gedrückt wird.

Die Flankenauswertungen verhindern ein mehrmaliges Senken. Auch hier wird kein Verriegelungsmerker benötigt. Allerdings kann nach NOT-AUS der Bohrkopf mehrmals gesenkt werden, was aber in der Praxis unschädlich sein dürfte.
Wenn der Bohrkopf unten angekommen ist, oder bei betätigtem <AUS>-Taster, wird „SENKEN“ zurückgesetzt.

Netzwerk 7: Bohrer heben

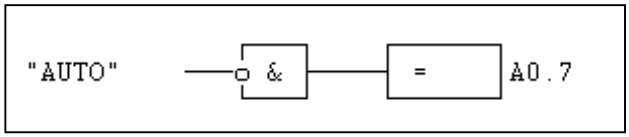


Zum Heben des Bohrkopfes nach einem NOT-AUS von Hand wird ein weiterer Taster (S) in das Pult gebaut. Wenn die Anlage (das Band) steht, kann der Bohrkopf hochfahren.

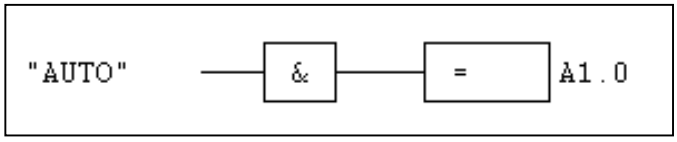
Ein kurzer „0“-Impuls von <NOT-AUS> ist zum um Ausschalten ausreichend.

Der bisherige Bohr-Merker wird nicht mehr benötigt. Das Netzwerk kann gelöscht werden.

Netzwerk 8: LED Steuerung von Hand / LED. (unverändertes Netzwerk)



Netzwerk 9: Automatikbetrieb / LED. (unverändertes Netzwerk)



Vermutlich können Sie jetzt Ihre Anlage ziemlich praxisgerecht bedienen. Sie können z.B. nach der Unterbrechung des Bohrvorganges mit <AUS> entscheiden, ob Sie den Bohrkopf heben oder senken wollen. <SENKEN> würde allerdings bedeuten, dass gebohrt wird. Anschließend würde der Bohrkopf nach oben fahren. Jetzt würde man erwarten, dass mit <BAND_START> das Teil abtransportiert würde. Im Netzwerk 1 verhindert nur noch „NOTAUS_V“ (Netzwerk 2) den Start. Mit <BAND_START> wird dieses jedoch entriegelt, so dass im nächsten Zyklus das Band anlaufen könnte.

Was passiert aber tatsächlich? Sie werden feststellen, dass statt dessen der Bohrkopf noch einmal gesenkt wird. Wie ist das möglich? Dieser Fall ist ein Beispiel dafür, dass nicht nur das einzelne Netzwerk logisch richtig programmiert sein muss, sondern dass die (richtige) Anordnung der Netzwerke ganz wichtig ist.
Im Netzwerk 1 ist die Starbedingung zwar gegeben, durch die „1“ an <R> wird der Start aber verhindert – das Band läuft (noch) nicht. Direkt danach (Netzwerk 2) ändert sich diese „Blockade“ zwar, hat aber für diesen Zyklus keine Auswirkung mehr auf das Band. Aber im Netzwerk 6 sind dadurch alle Bedingungen für „SENKEN“ gegeben. Folglich wird diese Operation zuerst ausgeführt. Wenn der Bohrkopf dann wieder oben angekommen ist, kann auch das Band den Rohling abtransportieren.
Wie wird das Problem gelöst? Ganz einfach: Setzen Sie bitte das Netzwerk 2 an die erste Stelle. Dazu aktivieren Sie im Editor das Netzwerk 2 und wählen <Bearbeiten>|<Netzwerk ausschneiden>. “Löschen“ Sie es ruhig mit <OK>, es geht Ihnen nicht verloren. Wählen Sie jetzt Netzwerk 1 aus und wählen <Bearbeiten>|<Netzwerk einfügen>. Sie werden sehen, dass sich jetzt die Reihenfolge der Netzwerke geändert hat.
Jetzt stimmt die Reihenfolge: Zuerst wird der Merker „NOTAUS_V“ entriegelt, so dass im neuen Netzwerk 2 das Band direkt aktiviert werden kann. Somit ist die Startbedingung für „SENKEN“ nicht mehr gegeben und das Programm arbeitet wunschgemäß.

Vgl. TrySim <Lösungen>|<BOHR_FL_AUTO>.

Projekt 12: Ablaufsteuerung

Schwerpunkte: Nachbildung des Prinzips der Ablaufsteuerung, SR-FF

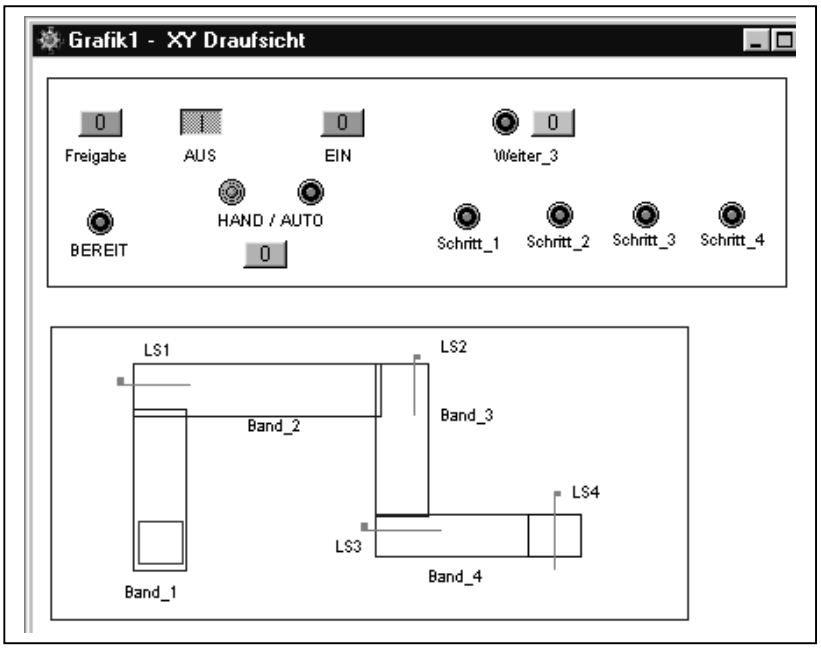
Aufgabe: Eine kleine Fertigungsstraße besteht aus 4 Transportbändern, über die in einem Zyklus immer nur 1 Paket befördert werden soll.

Nach der Freigabe mit einem Schlüsselschalter kann die Anlage mit dem Taster <EIN> gestartet werden. Es läuft nur Band 1. Wenn das Paket bei der Lichtschranke LS1 angekommen ist, wird das zweite Band ein- und das erste ausgeschaltet, usw. Nach dem 2. Band ist eine zusätzliche Weiterschaltbedingung durch den Taster <Weiter_3> zu erfüllen.

Mit dem Wahlschalter HAND / AUTO kann entschieden werden, ob sich der Zyklus automatisch wiederholen soll, oder nach dem letzten Takt beendet werden soll.

Bei <Aus> muss die Anlage sofort stehen. Wird zwischendurch die Freigabe beendet, wird der Zyklus zu Ende gefahren. Bei laufendem Programm darf nicht neu zu starten sein.
Nach <Aus> bei laufender Anlage sind noch vorhandene Pakete mit der TrySim-Löschfunktion (<Anlage>|<Dynamiks>|<Normale löschen> zu "entsorgen".
Die Aufgabe ist mit Schrittmerkern zu programmieren.

Theorie:
Dieses Projekt ist eine Anwendung des Prinzips der Schrittketten- oder auch Ablaufsteuerung, wie sie in den Sprachen AWL, FUP und KOP einfach zu erstellen ist. Die jetzige Aufgabe ist eine reine Folgeschaltung, in der ohne weitere Randbedingungen das reine Prinzip der Ablaufsteuerung überschaubar dargestellt wird.



Der Unterschied zum Projekt <F-Band> besteht darin, dass jedes Transportband nur so lange eingeschaltet wird, wie es zum Transport benötigt wird. Dadurch wird der Charakter der Ablaufsteuerung über Schrittmerker deutlich. Die Schritt-LEDs erlauben eine gute und einfache Kontrolle des Ablaufs und der raschen Fehlerdiagnose.

Symbol. Adresse	Operand	Datentyp	Kommentar
INIT	A 0.0	BOOL	LED Freigabe-Anzeige
LAusg_1	A 0.1	BOOL	LED Ausgang 1
LAusg_2	A 0.2	BOOL	LED Ausgang 2
LAusg_3	A 0.3	BOOL	LED Ausgang 3
LAusg_4	A 0.4	BOOL	LED Ausgang 4
LED_Hand	A 0.5	BOOL	LED Hand / 1 Durchgang
LED_Auto	A 0.6	BOOL	LED Automatik / Dauerbetrieb
Band_1	A 0.7	BOOL	Bandantrieb 1
Band_2	A 1.1	BOOL	Bandantrieb 2
Band_3	A 1.3	BOOL	Bandantrieb 3
Band_4	A 1.5	BOOL	Bandantrieb 4
Blink_W3	A 1.0	BOOL	Blink-LED Weiter_3
Paket	A 1.7	BOOL	Generator / Paketspender
Freigabe	E 0.0	BOOL	Freigabeschalter
AUS	E 0.1	BOOL	AUS-Taster (Ö)
Weiter_3	E 0.3	BOOL	zus. Weiterschaltbed. f. Band 3 (S)
EIN	E 0.6	BOOL	EIN-Taster (S)
Auto	E 0.7	BOOL	Wahlschalter HAND / AUTOMATIK
LS1	E 1.0	BOOL	Lichtschranke Ende Band 1
LS2	E 1.1	BOOL	Lichtschranke Ende Band 2
LS3	E 1.2	BOOL	Lichtschranke Ende Band 3
LS4	E 1.3	BOOL	Lichtschranke Ende Band 4
SRM1	M 0.1	BOOL	Schrittmerker 1
SRM2	M 0.2	BOOL	Schrittmerker 2
SRM3	M 0.3	BOOL	Schrittmerker 3
SRM4	M 0.4	BOOL	Schrittmerker 4
Startmrk	M 0.5	BOOL	Startmerker
Leeren	E 0.2	BOOL	Bänder abräumen

Symboltabelle

Es soll jetzt eine Struktur geschaffen werden, die einen übersichtlichen Ablauf ermöglicht.

Zur Methode der Schrittketten-Programmierung:

Die Aufgabe wird gedanklich in kleine, aufeinander folgende Phasen (Schritte) aufgeteilt.
In diesem Fall also:

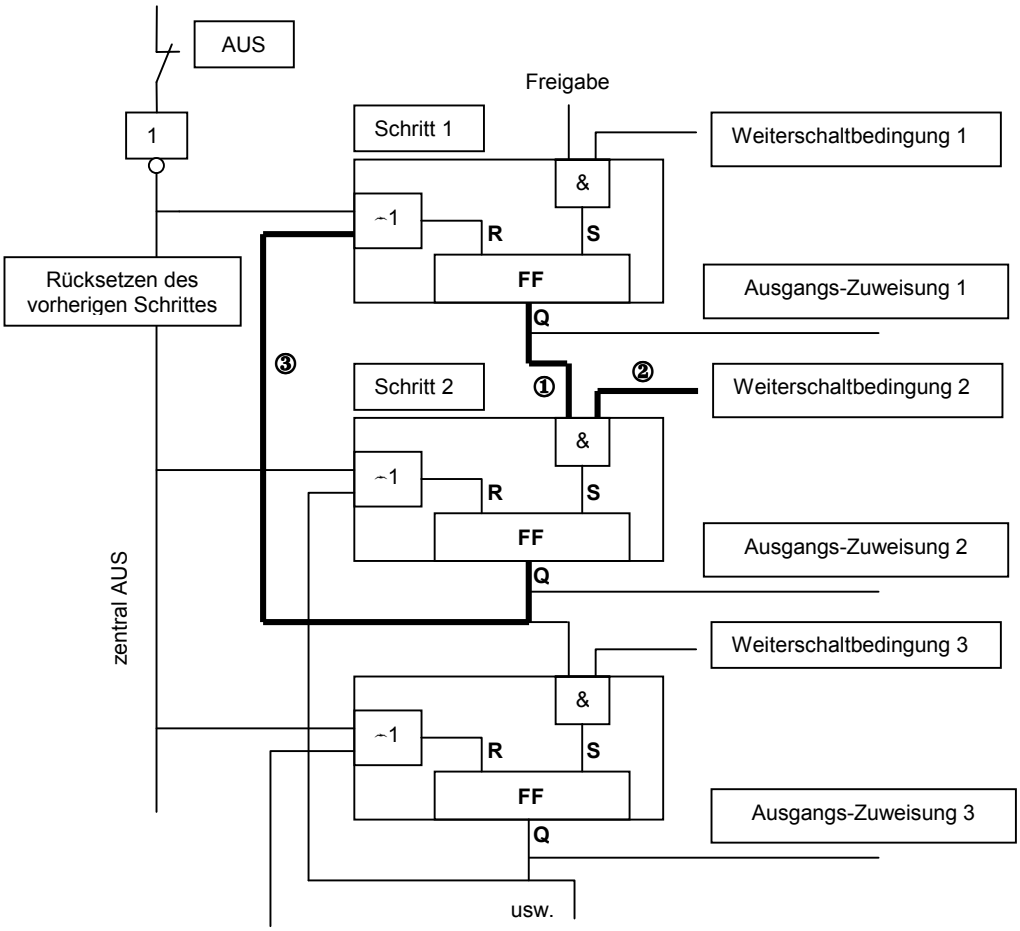
- Nur Band 1 ein
- Band 2 ein / Band 1 aus
- Band 3 ein / Band 2 aus
- Band 4 ein / Band 3 aus
- Band 4 aus.

Diese Schritte beschreiben, was wir wollen. Fast jeder Laie könnte diese Schrittliste erstellen. Und viel mehr benötigt die Fachkraft auch nicht.
Dabei ist lediglich zu berücksichtigen, dass der Folgeschritt immer durch den vorherigen "freigeschaltet" wird. Nach dieser Freigabe ist dann nur noch eine - meistens von außen kommende - Weiterschaltbedingung abzufragen (UND-verknüpft) und der nächste Schritt wird gesetzt. Die Schritte werden alle durch SR-Flip-Flops erzeugt. Dabei machen wir uns noch keine Gedanken, wodurch z.B. das Band eingeschaltet wird, sondern weisen den Schritten nur Merkeroperanden zu.

Jetzt kommt das wichtige Prinzip:

- Zuerst wird der nachfolgende Schritt aktiviert. **① + ②**
- Dieser neu eingeschalte SR-Merker schaltet den vorstehenden Schritt aus **③**

Prinzip der Schrittkette



Dieses Prinzip wird jetzt in TrySim-Netzwerke umgesetzt. Jedes Netzwerk besteht aus einem SR-FF. Vor dem S-Eingang wird eine UND-Verknüpfung gesetzt, deren Eingänge mindestens aus dem Ausgang des vorherigen Schrittmerkers und der Weiterschaltbedingung bestehen. (Nur die Ansteuerung des ersten Schrittmerkers sieht etwas anders aus).

Nachdem alle Schrittmerker in simpler und immer gleicher Weise mit Operanden und Kommentaren versehen wurden, sind separat (!) die Verknüpfungen zwischen den Merkern und den Ausgangsoperanden zu programmieren.

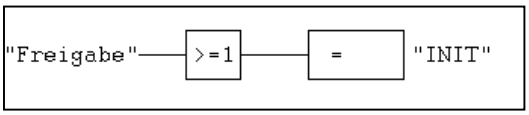
Zur Praxis mit TrySim:

Sie bekommen von Projekt zu Projekt weniger Hilfe bei (hoffentlich !) bekannten Details. Falls doch mehr Anleitung benötigen, blättern Sie bitte zu den vorherigen Projekten zurück.

- Starten Sie bitte das Programm TrySim.
- Öffnen Sie für die Projektierung im Ordner **<Vorlagen>** das Projekt **<AS1_SR_V>**, legen sich eine Kopie in Ihrem Ordner an und
- klicken Sie in das Editierfenster.

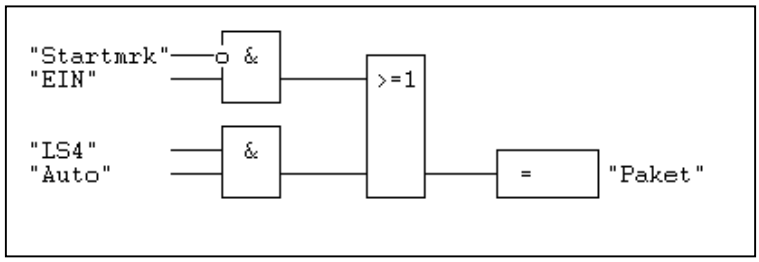
Programmerstellung

Netzwerk 1: Initialisierung (Hauptschalter)



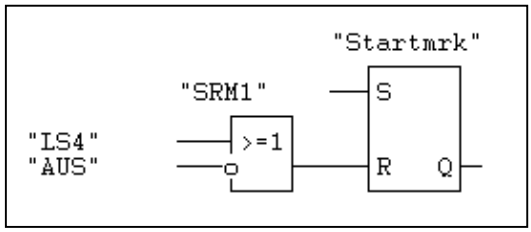
Wenn der <Freigabe>-Schalter betätigt / geschlossen ist, leuchtet die LED.

Netzwerk 2: Paket aus Magazin holen



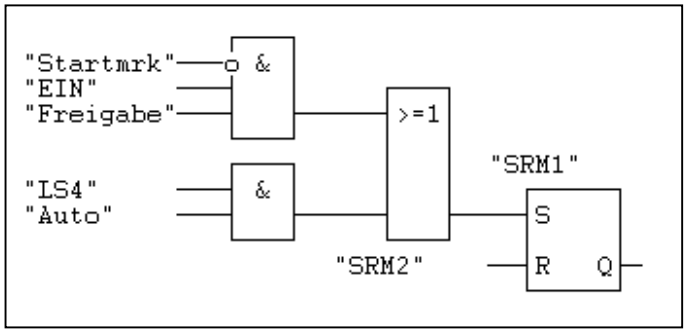
Pro Zyklus kann nur ein Paket geholt werden, weil anschließend "Startmrk" auf "1" gesetzt wird. Bei Automatikbetrieb wird ein Impuls von "LS4" für ein neues Paket benutzt.

Netzwerk 3: Startmerker



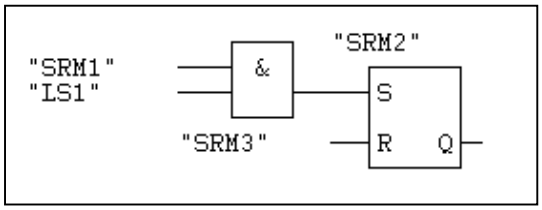
Wenn das erstemal "SRM1" gesetzt wird (im folgenden Netzwerk), wird "Sartmrk" dauerhaft auf "1" gesetzt. Erst beim Ende des gesamten Ablaufes wirdderBaustein zurückgesetzt - oder durch Druck auf <AUS> (betätigter Öffner = "0"; zurückgesetzt wird mit "1" an R).

Netzwerk 4: Schrittmerker 1



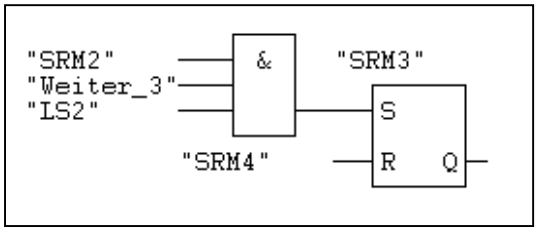
Nur beim ersten Takt, wenn der Startmerker nicht gesetzt ist, kann über <EIN> gestartet werden, sofern die Anlage freigegeben wurde.
Im Netzwerk 3 wird nach dem letzten Takt "Startmrk" zurückgesetzt, so dass über <EIN> neu gestartet werden kann. Bei Automatikbetrieb beginnt der Zyklus von allein neu, wenn "LS4" an Ende der Anlage "1" liefert.

Netzwerk 5: Schrittmerker 2



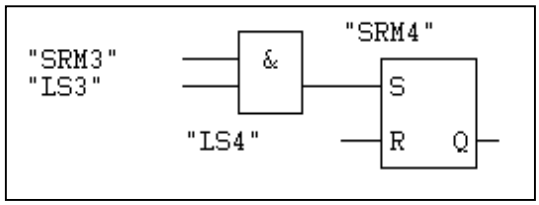
Durch die Vorbereitung vom "SRM1" und die Weiterschaltbedingung "LS1" wird dieser Schritt gesetzt. Dadurch wird der vorherige Schritt zurückgesetzt.
(Siehe dort).

Netzwerk 6: Schrittmerker 3



Durch die Vorbereitung vom "SRM2" und die Weiterschaltbedingung "LS2" wird dieser Schritt gesetzt, wenn zusätzlich <Weiter_3> betätigt wird. Der vorherige Schritt wird zurückgesetzt. (Siehe dort).
Anm.: <Weiter_3> wurde als Beispiel für zusätzliche Weiterschaltverknüpfungen gewählt.

Netzwerk 7: Schrittmerker 4



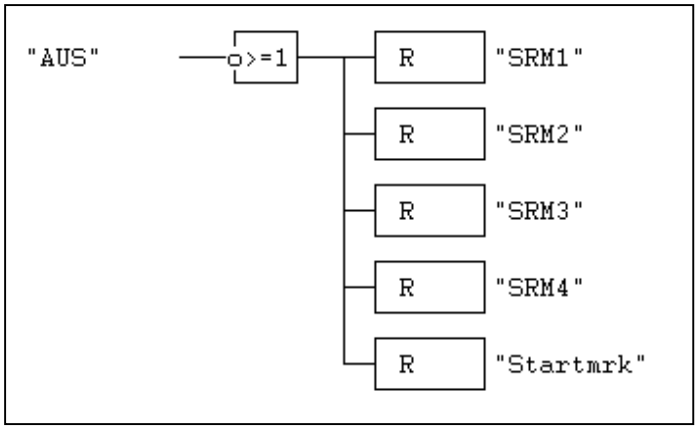
Der Setzvorgang wiederholt sich in immer gleicher Weise. Zurückgesetzt wird dieses letzte Netzwerk ausnahmsweise direkt von "LS4". Sonst schaltet dieses Netzwerk bei Einzelbetrieb nicht ab. (Es müsste ja sonst vom folgenden (ersten!) Netzwerk ausgeschaltet werden).

Netzwerk 8: Zentrales Rücksetzen bei AUS

```
ON "AUS"
R   "SRM1"
R   "SRM2"
R   "SRM3"
R   "SRM4"
R   "Startmrk"
```

Um die Verknüpfungslogik (Rücksetzen durch das folgende Netzwerk) deutlich herauszustellen, wurde das Rücksetzen bei <AUS> nicht in vorstehende Netzwerke aufgenommen. Wie Sie sehen, geht das auch zentral in einem separaten Netzwerk, das den Schrittmerkern folgen muss (damit die Anweisung durch die Reihenfolge dominiert).

Mehrere Zuordnungen können einfach in AWL geschrieben werden. Schalten Sie dafür das AWL-Icon an. Durch anschließendes Umschalten zu FUP erhalten Sie folgende Darstellung:



Netzwerk 9: Anzeige Hand / Automatikbetrieb

```
U  "Auto"
=  "LED_Auto"
UN "Auto"
=  "LED_Hand"
```

Netzwerk 10: Zentrale Zuordnung der Schrittmerker zu den Ausgängen

```
U  "SRM1"
=  "LAusg_1"
=  "Band_1"
U  "SRM2"
=  "LAusg_2"
=  "Band_2"
U  "SRM3"
=  "LAusg_3"
=  "Band_3"
U  "SRM4"
=  "LAusg_4"
=  "Band_4"
```

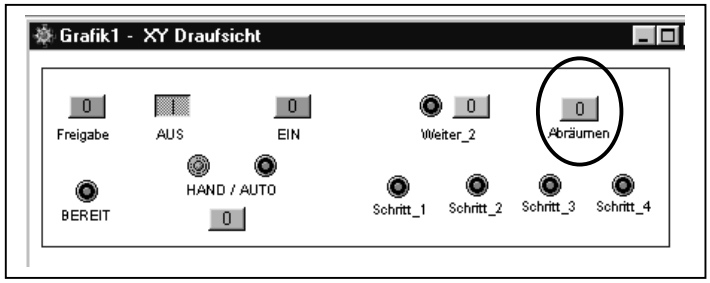
In diesem Netzwerk werden zentral alle Schrittmerker abgefragt und die Ausgänge zugewiesen.
Diese gemeinsame Zuweisung ist nur in AWL möglich. In FUP oder KOP müssten Sie sonst für jeden Ausgang ein separates Netzwerk wählen.

Netzwerk 11: Weiter_3 Hinweisleuchte

```
U  "LS2"      // Wenn die Lichtschranke 2 erreicht ist ...
=  A  1.0     // ...wird die Blink-LED eingeschaltet, um auf die Weiterschaltbedingung von Hand
              //hinzuweisen.
```

Damit ist die Aufgabe gelöst. Sie können Ihre Lösung auch mit TrySim <AS1_SR> im Verzeichnis <Lösungen> vergleichen

Es bleibt das Problem, dass nach dem Ausschalten bei laufendem Betrieb ein Paket irgendwo rumliegt und nicht entfernt werden kann, ohne den Zyklus durcheinander zu bringen. Denn zu starten ist die Anlage nur mit Band 1 und weitergeschaltet wird nur nach Betätigung der zugehörigen Lichtschranken.
Da hilft nichts: Wir müssen noch etwas umbauen.

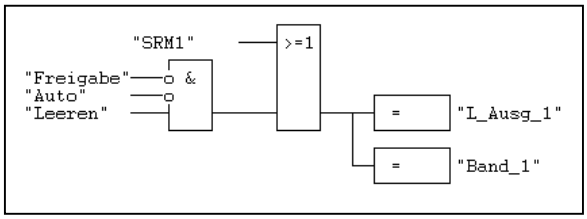


Es wird ein separater Schaltbefehl zum "Abräumen" benötigt. Dazu wird bei ausgeschalteter Anlage und aktivem Grafikfenster ein Schalter in das Pult "gebaut", d.h., Sie ziehen mit <LM> von <Anlage | Neues Element> den <Schalter> in das Pult. Mit kurzem (!) <rM>-Klick auf den neuen Schalter öffnet sich das Editierfenster, indem Sie z. B. den vorgeschlagenen Operanden, oder den Namen, oder die Schalterfarbe ... ändern können. Es ist auch möglich, den Schalter gegen "Verrutschen" mit <Fixieren> zu sichern.

Die Bänder sollen jetzt nicht nur über die Schrittmerker, sondern auch separat anzusteuern sein. Natürlich nicht während des Betriebes. Dazu sind ODER-Verknüpfungen erforderlich. Denken Sie bitte daran: Ein Ausgang darf nur in einem einzigen Netzwerk programmiert werden, damit alle Verknüpfungen wunschgemäß funktionieren. Würden Sie also in einem Netzwerk einen Operanden so programmiert haben, dass das VKE "1" wäre und in einem anderen Netzwerk das VKE desselben Operanden "0", so wird nur das VKE des zuletzt programmierten Netzwerkes ausgeführt. Die ODER-Verknüpfung löst das Problem. Die einzige Ausnahme der wiederholten Programmierung eines Ausganges in verschiedenen Netzwerken ist der Setz- (S) bzw. Rücksetzbefehl (R), wie noch gezeigt wird.

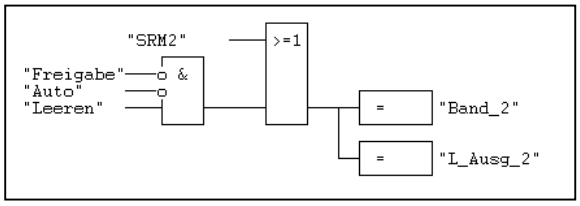
Wegen der Übersichtlichkeit wird das bisherige Netzwerk 10 (alle Ausgänge) gelöscht und jeder Ausgangs-Operand wird in FUP in einem eigenen Netzwerk programmiert.

neu: Netzwerk 10: Zuordnung des Schrittmerkers 1 zum Ausgang Band 1



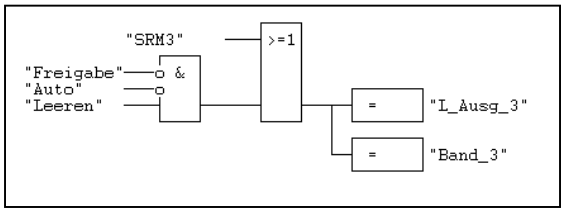
Das Band 1 wird entweder vom Schrittmerker 1 oder - wenn <Freigabe> und <Auto> ausgeschaltet sind - durch den Schalter <Abräumen>.
Diesem Schalter wurde in der Symboltabelle bereits die symbolische Adresse "Leeren" zugeordnet.

Netzwerk 11: Zuordnung des Schrittmerkers 2 zum Ausgang Band 2

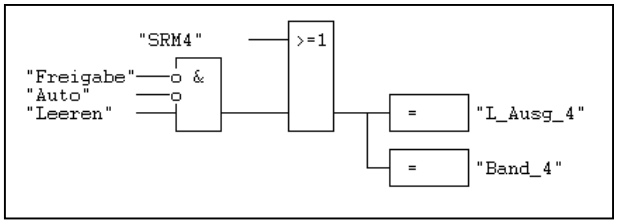


Das Band 2 wird entweder vom Schrittmerker 2 oder - wenn <Freigabe> und <Auto> ausgeschaltet sind - durch den Schalter <Abräumen>.

Netzwerk 12: Zuordnung des Schrittmerkers 3 zum Ausgang Band 3



Netzwerk 13: Zuordnung des Schrittmerkers 4 zum Ausgang Band 4



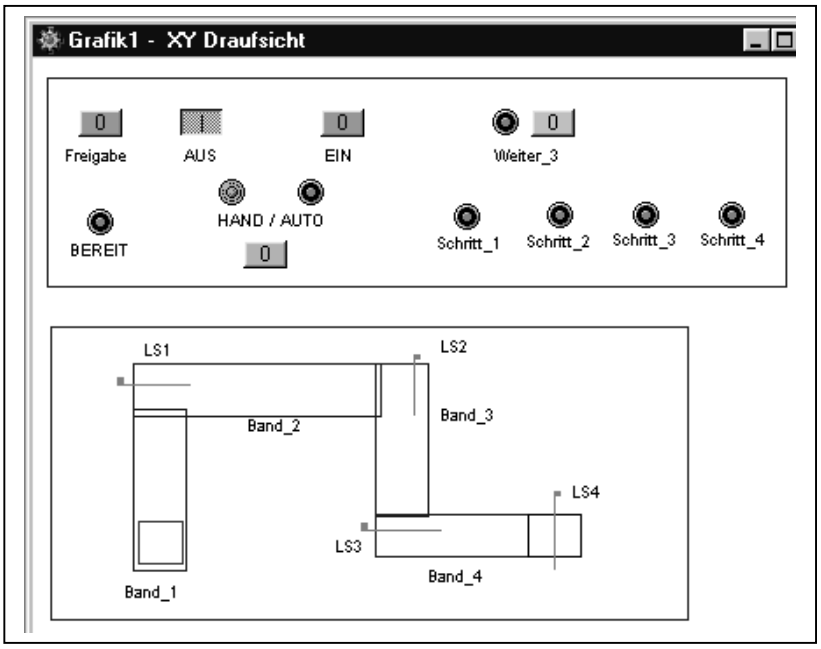
Sie haben gelernt:

Anlagen, die eine fest vorgegebene Reihenfolge von Abläufen aufweisen, lassen sich durch eine Reihe von Schritten programmieren.
Deshalb spricht man bei dieser Methode auch von der **Schrittkette** oder der **Ablaufsteuerung.**
Jeder Schritt wird durch ein SR-FlipFlop erzeugt und dient als Merker für den augenblicklich aktiven Schritt **(Schrittmerker).**
In einer linearen Schrittkette kann immer nur jeweils1 Schritt aktiv sein.
Der jeweils aktive Schritt liefert eine "1", die mit einer Weiterschaltbedingung UND-verknüpft wird.
Der neu aktivierte Schritt setzt den vorherigen Schritt zurück.
Die eigentlichen Ausgangsoperanden werden nicht in der Schrittkette, sondern danach (!) durch Abfrage der Schrittmerker programmiert.
Not-Aus wirkt auf jeden Schrittmerker.
Wenn Sie die einzelnen Schritte auf LEDs führen, kann der Wartungsdienst sehr schnell den Fehler einkreisen. Es ist nur die jeweilige Weiterschaltbedingung zu prüfen.
Jeder Ausgang wird nur einmal (in einem Netzwerk) zugewiesen.
Die Schrittkette darf erst dann erneut gestartet werden, wenn es keine kritischen Überschneidungen mit Befehlen aus anderen Netzwerken gibt. In der Regel gibt erst der letzte Schrittmerker den ersten wieder frei.

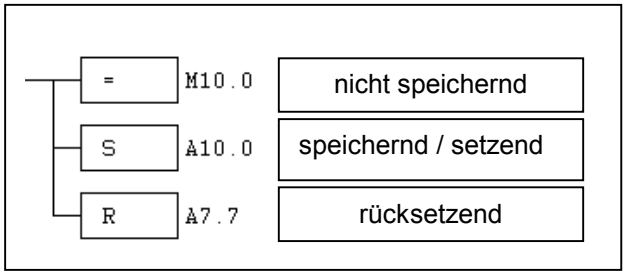
Projekt 13: Ablaufsteuerung 2

Schwerpunkte: Nachbildung des Prinzips der Ablaufsteuerung,
getrennte speicherbare Ausgänge S und R

Aufgabe: Eine kleine Fertigungsstraße besteht aus 4 Transportbändern, über die in einem Zyklus immer nur 1 Paket befördert werden soll.
Nach der Freigabe mit einem Schlüsselschalter kann die Anlage mit dem Taster "EIN" gestartet werden. Es läuft nur Band 1. Wenn das Paket bei der Lichtschranke LS1 angekommen ist, wird das zweite Band ein- und das erste ausgeschaltet, usw. Nach dem 2. Band ist eine zusätzliche Weiterschaltbedingung durch den Taster <Weiter_3> zu erfüllen.
Mit dem Wahlschalter <HAND / AUTO> kann entschieden werden, ob sich der Zyklus automatisch wiederholen soll, oder nach dem letzten Takt beendet werden soll.
Bei Aus muss die Anlage sofort stehen. Wird zwischendurch die Freigabe beendet, wird der Zyklus zu Ende gefahren. Bei laufendem Programm darf nicht neu zu starten sein.
Nach Auch bei laufender Anlage sind noch vorhandene Pakete mit der TrySim-Löschfunktion <Anlage | Dynamiks | Normale löschen> zu "entsorgen".
Die Aufgabe ist ohne SR-Bausteine, aber mit speichernden Zuweisungsblöcken zu programmieren.



Theorie:
Dieses Projekt ist eine Variante der Ablaufsteuerung <AS1_SR>. Die Verknüpfungsaufgabe ist gleichgeblieben, jedoch wird sie ohne SR-Bausteine realisiert. Schon im Netzwerk 8 der Projektes <AS1_SR> haben wir beim Übersetzen von AWL in FUP eine neue Art von Elementen kennengelernt, aber noch nicht sonderlich beachtet. Ein Ausgangsblock kann unterschiedliches Speicherverhalten haben.



Die Eigenschaft des Ausgangsblocks wird geändert, indem Sie den gewünschten Block markieren und die Leertaste drücken.

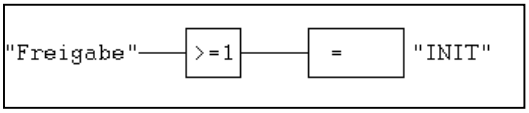
Anstelle der SR-Schrittmerker sollen jetzt die Schrittmerker als Ausgangsblöcke programmiert werden. Durch diese Methode wird der vorherige Schrittmerker unmittelbar, d.h. im selben Bearbeitungszyklus zurückgesetzt. Bei einem SR-Schrittmerker erfolgt das Rücksetzen erst im Folgezyklus. Das mag in den meisten Fällen unkritisch sein. Hier soll auch nur die Methode und der Unterschied aufgezeigt werden.

Alle Operanden bleiben gleich. Benutzen Sie auch für dieses Projekt die Vorlage <AS1_V>. Vergleichen Sie mit dem TrySim-Projekt <AS1_VAR1> im Verzeichnis <Lösungen>.

Symbol. Adresse	Operand	Datentyp	Kommentar
INIT	A 0.0	BOOL	Freigabe-Anzeige
LAusg_1	A 0.1	BOOL	LED Ausgang 1
LAusg_2	A 0.2	BOOL	LED Ausgang 2
LAusg_3	A 0.3	BOOL	LED Ausgang 3
LAusg_4	A 0.4	BOOL	LED Ausgang 4
LED_Hand	A 0.5	BOOL	LED Hand / 1 Durchgang
LED_Auto	A 0.6	BOOL	LED Automatik / Dauerbetrieb
Band_1	A 0.7	BOOL	Bandantrieb 1
Band_2	A 1.1	BOOL	Bandantrieb 2
Band_3	A 1.3	BOOL	Bandantrieb 3
Band_4	A 1.5	BOOL	Bandantrieb 4
Blink_W3	A 1.0	BOOL	Blink-LED Weiter_3
Paket	A 1.7	BOOL	Generator / Paketspender
Freigabe	E 0.0	BOOL	Freigabeschalter
AUS	E 0.1	BOOL	AUS-Taster (Ö)
Weiter_3	E 0.3	BOOL	zus. Weiterschaltbed. f. Band 3 (S)
EIN	E 0.6	BOOL	EIN-Taster (S)
Auto	E 0.7	BOOL	Wahlschalter HAND / AUTOMATIK
LS1	E 1.0	BOOL	Lichtschranke Ende Band 1
LS2	E 1.1	BOOL	Lichtschranke Ende Band 2
LS3	E 1.2	BOOL	Lichtschranke Ende Band 3
LS4	E 1.3	BOOL	Lichtschranke Ende Band 4
zuletzt	M 4.0	BOOL.....	Ende-Merker
SRM1	M 4.1	BOOL	Schrittmerker 1
SRM2	M 4.2	BOOL	Schrittmerker 2
SRM3	M 4.3	BOOL	Schrittmerker 3
SRM4	M 4.4	BOOL	Schrittmerker 4
Startmrk	M 4.5	BOOL	Startmerker

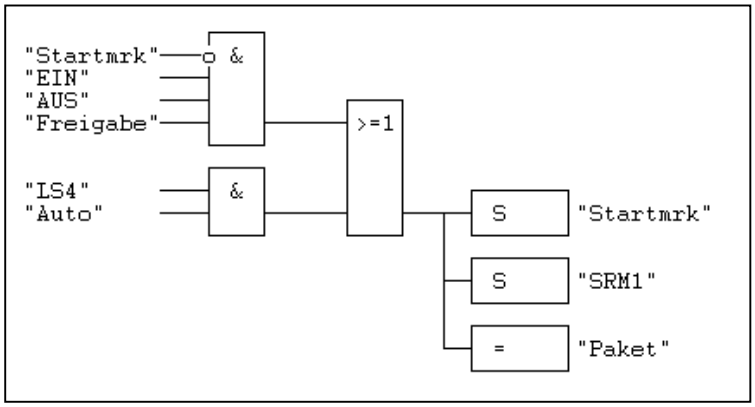
Symboltabelle

Netzwerk 1: Initialisierung (Hauptschalter)



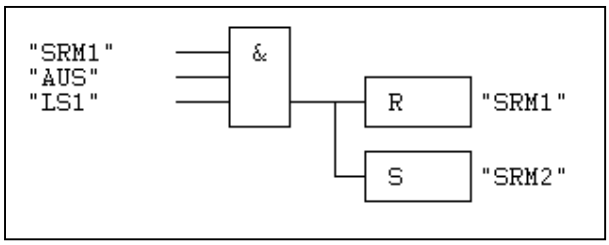
Hier hat sich nichts geändert.

Netzwerk 2: Schrittmerker 1 - Band 1
Nur beim ersten Takt, wenn der Startmerker nicht gesetzt ist, kann über "EIN" gestartet werden.



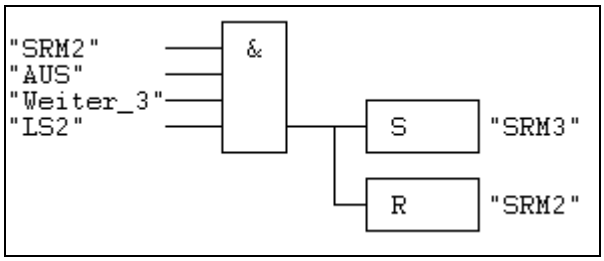
Der Startmerker verhindert, dass ein weiteres Paket angefordert werden kann, solange noch eines unterwegs ist.
"Startmrk" bleibt gesetzt, ebenfalls "SRM1".

Netzwerk 3: Schrittmerker 2 - Band 2
Durch die Vorbereitung vom Schrittmerker 1 und die Weiterschaltbedingung wird der nächste Schritt gesetzt und gleichzeitig der vorherige Schritt zurückgesetzt.

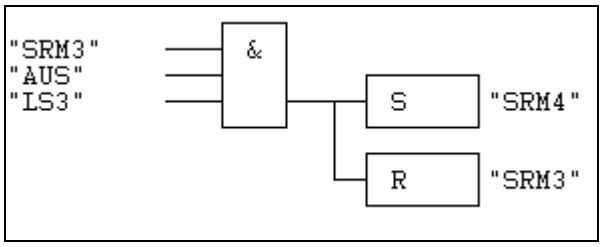


Der "SRM1" setzt zusammen mit den Weiterschaltbedingungen „SRM2“. Es sieht zwar so aus, als würde sich vorher „SRM1“ selbst zurücksetzen. Beide Befehle werden in diesem Netzwerk „in Auftrag gegeben“, aber erst am Zyklusende unabhängig voneinander ausgeführt.

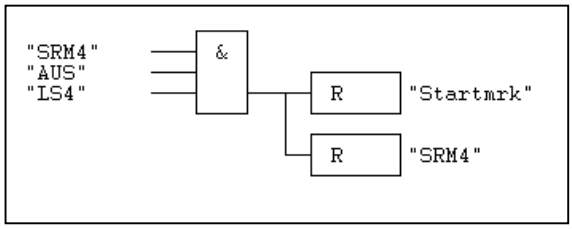
Netzwerk 4: Schrittmerker 3 - Band 3



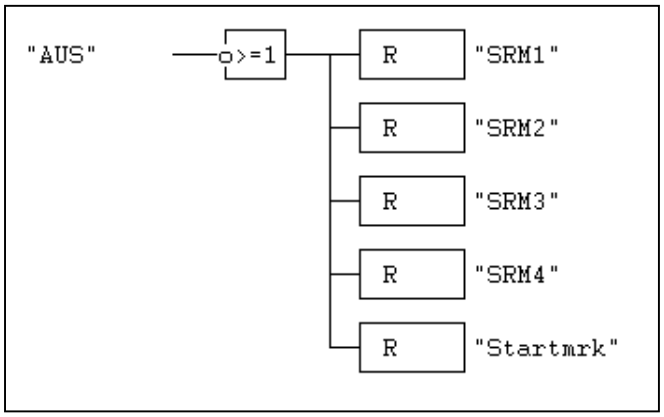
Netzwerk 5: Schrittmerker 4 - Band 4



Netzwerk 6: Schritt 5 Weiterschalten zum START



Netzwerk 7: Zentrales Rücksetzen bei NOT-AUS




```
ON "NOT-AUS"
R  "SRM1"
R  "SRM2"
R  "SRM3"
R  "SRM4"
R  "Startmrk"
```

AWL

FUP (wie gehabt)

Bei den restlichen Netzwerken treten keine Änderungen auf.

Netzwerk 8: Anzeige Hand / Automatikbetrieb

```
U    "Auto"
=    "LED_Auto"
UN   "Auto"
=    "LED_Hand"
```

Netzwerk 9: Zuweisung der (Motor-) Ausgängen und LEDs entsprechend der Schrittmerker

```
U    "SRM1"
=    "Ausg_1"
=    "Band_1"
U    "SRM2"
=    "Ausg_2"
=    "Band_2"
U    "SRM3"
=    "Ausg_3"
=    "Band_3"
U    "SRM4"
=    "Ausg_4"
=    "Band_4"
```

Netzwerk 10: Weiter_3 Hinweisleuchte

```
U    "LS2"          // Wenn die Lichtschranke 2 erreicht ist ...
=    „Blink_W3“     // wird die Blink-LED eingeschaltet, um auf die Weiterschaltbedingung von
                    //Hand hinzuweisen.
```

Was kann eine SPS? B) Zeitsteuerungen

Projekt 14: 3 Förderbänder (Automatisches Einschalten)

Schwerpunkte: Zeitglieder, Änderungen und Ergänzungen im Projekt.

Aufgabe: Es ist eine Schaltung für das **Zusammenspiel von 3 Förderbändern** zu entwerfen. Dabei ist auf die logisch richtige Reihenfolge beim Ein- und Ausschalten zu achten, damit kein Stau entstehen kann, bzw. die Bänder zum nächsten Einschalten entlastet sind. Die Problematik des Zusammenspiels wurde bereits bei vorherigen Projekten erörtert. Jetzt soll die Anlage mit nur einem Taster gestartet werden. Die anderen Bänder schalten sich in der richtigen Reihenfolge automatisch zu. Bei "NOT-AUS" stehen die Bänder sofort, bei "AUS" fahren die Bänder leer und schalten nacheinander ab.
Sie können an Ihr Projekt anknüpfen, das Sie aus der Vorlage <F_Band_V> entwickelt haben. Wir empfehlen, von der **<SR>-Variante** auszugehen. Erstellen Sie bitte von diesem lauffähigen Projekt eine Kopie und nennen diese z. B. <F_Band_AUTO>.

Theoretische Erörterung der Aufgabe:

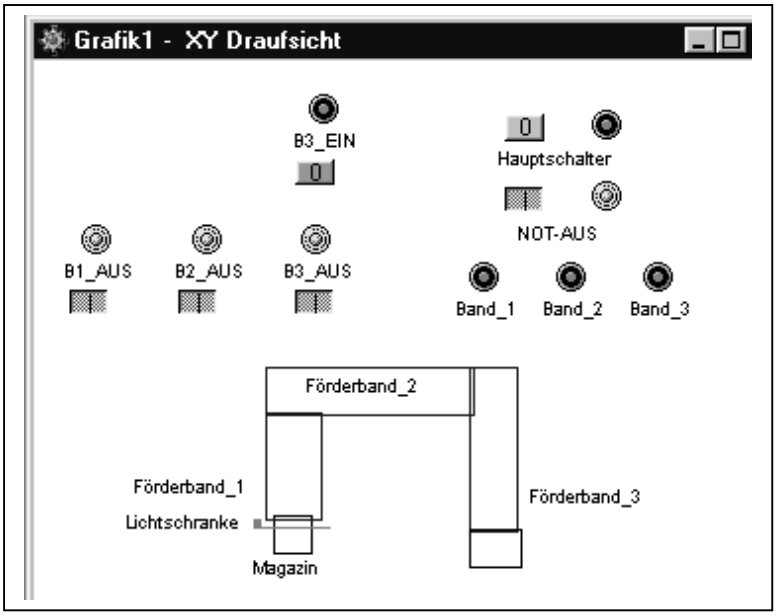
In dieser Kopie soll nicht alles neu programmiert werden, sondern es werden nur die erforderlichen Anpassungen und Änderungen vorgenommen - ein klassischer Anwendungsfall in der Praxis.
Die "EIN"-Taster für das zweite und erste Band sind funktionslos, bzw. sie sind "auszubauen" (Wie, folgt unten). Die Anlage soll sich aber weiterhin so verhalten, als würde sie von diesen Schaltkontakten gesteuert. Da liegt es nahe, Zeitrelais einzubauen, und deren **Ausgänge (!) an die Stelle der "EIN"-Taster** einzubauen. Es sind also nur die Operanden auszutauschen. Es empfiehlt sich, weitere Änderungen wegen der Übersichtlichkeit und der Vertrautheit mit der bisherigen Anlage in neuen Netzwerken vorzunehmen (obwohl es natürlich auch funktionieren würde, wenn alle Änderungen in die alten Netzwerke "hineingewürgt" würden). Es ist eine Geschmacks- oder Erfahrungssache, ob man lieber in einem Netzwerk den Gesamtzusammenhang verfolgen möchte, oder lieber Teilaufgaben separat aufbaut.
Das dritte Band, das zuerst starten soll, darf natürlich nicht zeitverzögert anlaufen. Stellen Sie sich bitte vor, der Bediener drückt <EIN> und nichts passiert. Vielleicht greift er gar in die Maschine, weil er glaubt, sie sei nicht startbereit!
Nach Möglichkeit sollte ein Programm immer ereignisabhängig verriegelt sein und nicht nur zeitabhängig. Was heißt das? Würde mit dem "EIN"-Befehl das 3. Band anlaufen und mit demselben Taster die Zeitrelais aktiviert werden, könnte es sein, dass Band 3 nicht anläuft (z.B. Lastssicherungen defekt, Schütz klemmt, Spule hat Unterbrechung) und nach dem Zeitablauf startet Band 2. Das darf nicht sein (Gefahr: siehe oben). Ereignisabhängig wäre z.B. die Quittierung der Schützstellung durch einen zusätzlich abgefragten Schützkontakt an der Eingangsklemmleiste oder zumindest die Verriegelung über die Anschlussklemme der Schützspule.

Zur Praxis mit TrySim:

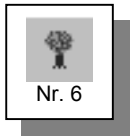
Hinweis: Falls sich beim Test zu viele Pakete unkontrolliert stauen, können diese gelöscht werden mit:<Anlage>|<Dynamiks>|<Normale löschen>.

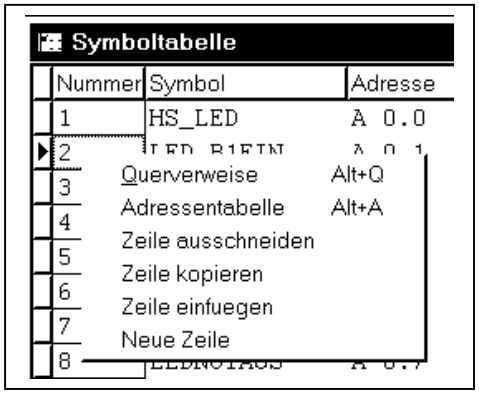
Das Pult der alten Anlage wird "umgebaut" für die neue Aufgabe. Klicken Sie bitte bei ausgeschalteter Anlage auf den Taster <B1_EIN> und halten Sie <rM> gedrückt: Es öffnet sich ein Kontextfenster, in dem Sie <löschen> drücken können. Schon haben Sie den Taster "spurenfrei" aus Ihrem Pult entfernt. Verfahren Sie bitte mit <B2_EIN> und der LED ebenso. Der Text zwischen LED und Taster stammte von der LED. Er wird gleichzeitig mit der LED gelöscht. Mit **kurzem** Klick <rM> können Sie im Editierfenster den Text ändern. Wenn Sie das Häkchen im Fenster <Fixieren> entfernen, können Sie das Element ganz oder den Text allein verschieben.

Das „umgebaute“ Pult:

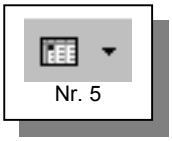


Nicht nur im Pult, sondern auch im Elementbaum (Icon Nr. 6) sind die Elemente gelöscht. Allerdings noch nicht in der Symboltabelle.





Wählen Sie Icon Nr. 5 und markieren unter <Nummer> die Zeile, die Sie löschen wollen. Mit <rM> öffnet sich obiges Fenster und Sie können <Zeile ausschneiden> wählen. Verfahren Sie so mit allen Zeilen, die nicht mehr benötigt werden und passen Sie den Text der symbolischen Adressen und Kommentare an.

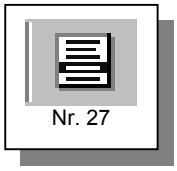


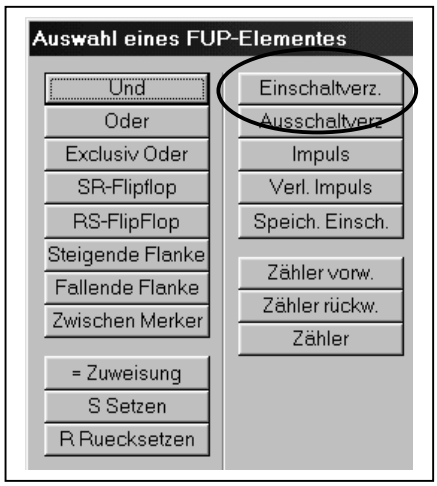
Dann könnte Ihre Symboltabelle so aussehen:

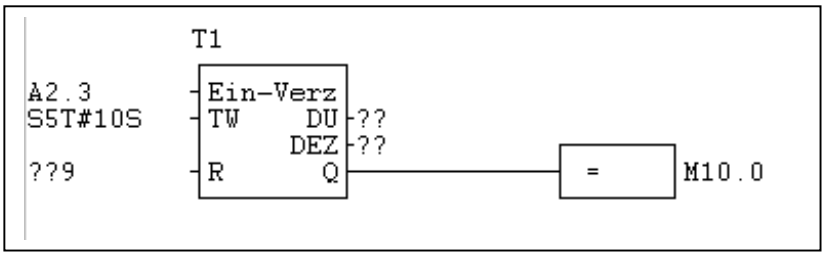
Symbol	Operand	Datentyp	Kommentar
LED_HS	A 0.0	BOOL	Hauptschalter-Pegel
LED_B1AUS	A 0.2	BOOL	Pegel Taster B1_AUS
LED_B2AUS	A 0.4	BOOL	Pegel Taster B1_AUS
LED_B3EIN	A 0.5	BOOL	Pegel Taster B1_EIN
LED_B3AUS	A 0.6	BOOL	Pegel Taster B1_AUS
LEDNTAUS	A 0.7	BOOL	Pegel NOT-AUS-Taster
LED_B1	A 1.1	BOOL	LED BAND 1
LED_B2	A 1.2	BOOL	LED BAND 2
LED_B3	A 1.3	BOOL	LED BAND 3
Band_1	A 2.1	BOOL	Motor Band 1
Band_2	A 2.2	BOOL	Motor Band 2
Band_3	A 2.3	BOOL	Motor Band 3
MAGAZIN	A 2.5	BOOL	Paketgenerator
Hauptsch	E 0.0	BOOL	Hauptschalter
B1_AUS	E 0.2	BOOL	Band1 AUS (Ö)
B2_AUS	E 0.4	BOOL	Band2 AUS (Ö)
B3_EIN	E 0.5	BOOL	Band 3 Ein (S)
B3_AUS	E 0.6	BOOL	Band3 AUS (Ö)
NOT-AUS	E 0.7	BOOL	Taster NOT-AUS (Ö)

Jetzt werden die Netzwerke angepasst: Da die LEDs für Band 1 und Band 2 nicht mehr vorhanden sind, sollten auch die zugehörigen Netzwerke gelöscht werden. In der Vorlage lagen sie in den Netzwerken 1 bis 4. Öffnen Sie bitte Netzwerk 1 und wählen <Bearbeiten> und <Netzwerk ausschneiden>. Das nachfolgende Netzwerk wird jetzt Nr. 1. Achten Sie also auf den Inhalt der Netzwerke beim Löschen und nicht auf die Netzwerknummer.

Die anderen Netzwerke bleiben erhalten, werden aber noch angepasst. Vorher beschäftigen wir uns jedoch mit den Zeitgliedern. Weiter oben wurde bereits vorgeschlagen, neue Funktionen in neuen Netzwerken zu programmieren. Aktivieren Sie das letzte (leere) Netzwerk und klicken dann auf Icon Nr. 27. Es öffnet sich ein Auswahlfenster mit FUP-Elementen.



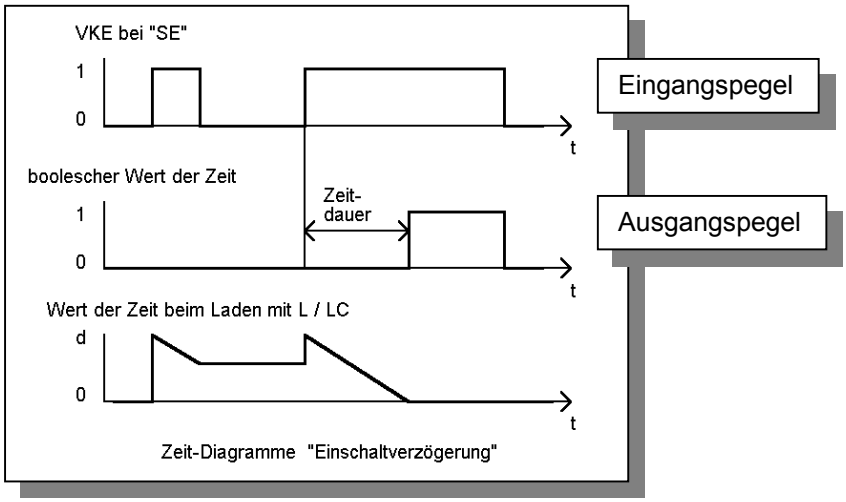




Einschaltverzögerung Band 2
Minimalkonfiguration
T1: Bezeichnung des Timers
A 2.3: wählbarer Operand (für die Eingangsabfrage)
S5T#: muss geschrieben werden,
gefolgt von der Verzögerungszeit; hier: **10s**

Auswahl, Teilansicht
Dort wählen Sie bitte die <Einschaltverz.> und tragen die Werte nach vorstehender Liste ein.

Vergleichen Sie bitte in der TrySim-Hilfe unter <Index> Eingabe: "SE" , <Anzeige>, dann <Operationen mit Zeiten>. Wenn Sie dort auf "Einschaltverzögerung" klicken, kommen Sie zu folgendem Zeitdiagramm:



Sie erkennen, dass der Ausgang auf "1" gesetzt wird, nachdem eine "1" am Eingang angelegt wurde und <Zeitdauer> bei "1"-Pegel verstrichen ist. Beachten Sie bitte: Der Ausgang fällt sofort auf "0" zurück, sobald der Eingang "0"-Pegel führt. Ein Rücksetzen ist nicht erforderlich.
Der Ausgangsoperand kann als "T1" aufgerufen werden (also nicht: "T1.Q"). Für eine sichere Abfrage sollte jedoch immer ein Zuweisungsblock dahinter stehen (oben: M10.0).
Die Eingabe der Zeit ist im Hilfe-Index unter "S5TIME" erläutert:

Eine typische S5TIME-Konstante sieht so aus:

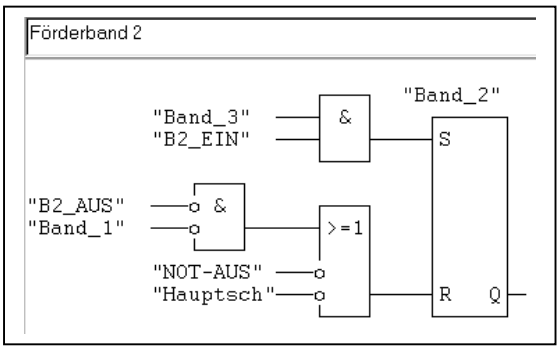
S5T#2H	2 Stunden
S5T#20MS	20 Millisekunden
S5T#1M5S	1 Minute und 5 Sekunden
S5T#1H200S	1 Stunde und 200 Sekunden, wird automatisch umgewandelt nach:
S5T#1H3M20S	1 Stunde und 3 Minuten und 20 Sekunden

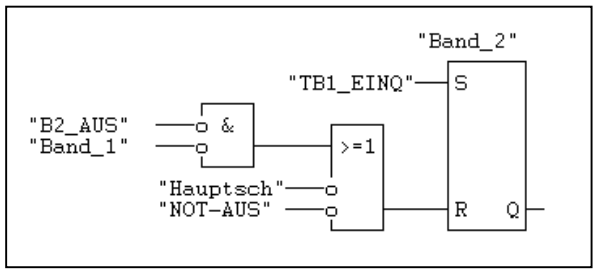
Sobald das Band 3 läuft, wird der Timer T1 für die erste Zeitverzögerung gestartet. Dieser Timer startet nach Zeitablauf das 2. Band. In dem vorhandenen Netzwerk für Band 2 werden die Änderungen vorgenommen: Die symbolische Adresse für T1 kann als "TB2_EIN" in die Symboltabelle eingetragen werden, der am Ausgang liegende Merker M 10.0 als "TB2_EINQ".
Klicken Sie mit <LM> in die letzte Zeile hinter den Kommentar und drücken die <Enter>-Taste. Eine neue Zeile öffnet sich.

Ergänzung der Symboltabelle:

TB2_EIN	T 1	TIMER	Einschaltverzögerung Band 2
TB2_EINQ	M 10.0	BOOL	Ausgang von TB2_EIN (Anschluss Q)
TB1_EIN	T 2	TIMER	Einschaltverzögerung Band 1
TB1_EINQ	M 10.1	BOOL	Ausgang von TB1_EIN (Anschluss Q)

Hinweis: Vergessen Sie bitte nicht, nach einer Änderung im Editierfenster das Übertragungs-Icon zu betätigen, damit die Änderung auch in die Anlage übertragen wird.





Das alte Netzwerk wird .....>>>..... umgebaut.

Wenn Sie nur den Operanden ändern wollen, markieren Sie die Stelle und schreiben einfach drauf los. Schließen Sie mit <Enter> ab. Wenn Sie einen Eingang oder einen Baustein löschen wollen, markieren Sie diesen und drücken die <Entf>-Taste.
Sicherer ist es, <symbolische Darstellung> auszuschalten und die leichter zu schreibenden direkten Operanden einzutragen.

In der symbolischen Darstellung muss der Ausdruck exakt der Schreibweise in der Symboltabelle entsprechen, sonst erkennt das Programm die Variable nicht.

Wenn Sie Operanden aus der Anlage mit Klick übertragen, funktioniert das immer richtig. Da Merker aber nicht in der Anlage sichtbar sind, müssen sie von Hand eingetragen werden.

Da es keinen <EIN>-Taster für Band 2 mehr gibt, ist auch keine Verriegelung am Eingang "S" erforderlich.

Band 1 wird in gleicher Weise verriegelt. Testen Sie jetzt bitte die Einschaltverriegelung, bzw. das automatische Anlaufen.

Soweit ganz gut. Aber wie sieht es mit dem Ausschalten aus?

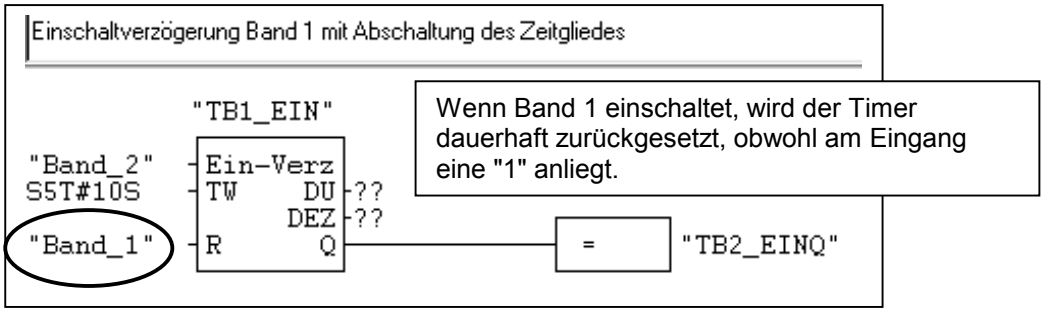
Rücksetzen der Timer

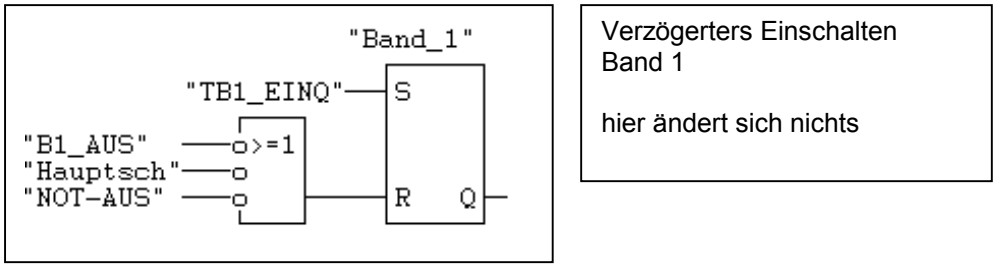
Die Verriegelungslogik wurde bisher nicht geändert. Deshalb kann nur zuerst Band 1 abgeschaltet werden. Aktivieren Sie bitte für die Analyse das Auge-Icon. Obwohl für „Band_1“ Eingang <R> eindeutig mit "1" belegt wird - solange Sie tasten - schaltet das SR-Flipflop nicht dauerhaft ab. Das liegt an der "1" am <S>-Eingang. Wenn die dominierende "1" an <R> nicht mehr anliegt, schaltet <S> wieder ein. Das darf natürlich nicht sein. Wie bekommt man die "1" an <S> weg, wenn diese Abfrage doch beim Einschalten richtig und erforderlich ist? Der Startbefehl des Timers T2 darf nur beim ersten Einschalten wirksam werden.

Eine "Wiedereinschaltsperre" kann man erreichen, wenn man den bisher nicht benutzten <R>-Eingang der Timers verwendet. Dieser war für die Funktionsfähigkeit nicht erforderlich, bringt aber einen wesentlichen Vorteil: Sie hatten bereits aus dem Zeitablaufdiagramm erkannt, dass die Einschaltverzögerung mit dem Übergang von "0" auf "1" am Eingang startet. Und genau auf diesen Übergang kommt es an. Wenn bisher eine Dauer-"1" am Eingang anlag, führte der Ausgang auch - zeitversetzt - dauerhaft "1". Wird jetzt aber mit einem <R>-"1"-Impuls der Baustein zurückgesetzt - ohne am <S>-Eingang den "1"-Pegel zu verändern - gibt es am Eingang keinen neuen "0"-->"1"-Übergang; der Timerbaustein wird nicht erneut aktiviert.

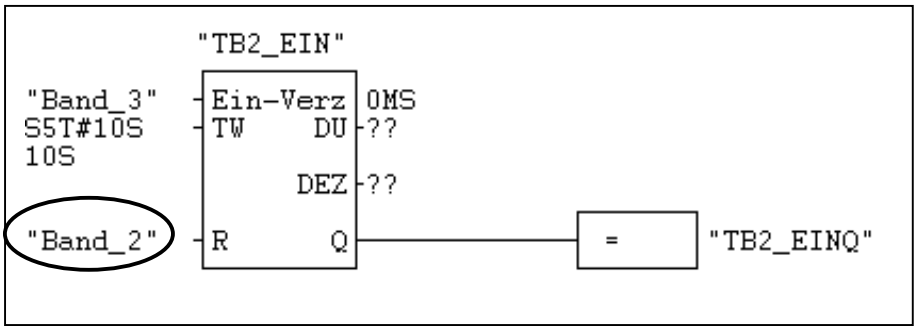
Der Eingang der Timer ist flankengesteuert

Daraus ergibt sich folgende Möglichkeit:





In gleicher Weise wird T1 gesperrt:



Vgl. TrySim-Projekt <F_Band_SR_AUTO>

Sie haben gelernt:

- Ein einschaltverzögerter Timer benötigt eine Dauer-„1“ am Eingang.
- Timer sind flankengesteuert.
- Nach dem Rücksetzen eines Timers über „R“ ist eine noch anstehende „1“ am <S>-Eingang unwirksam.
- Timer sollten ereignisabhängig gesetzt werden und nicht von anderen Timern

Projekt 15: 3 Förderbänder (Automatisches Ein- und Ausschalten)

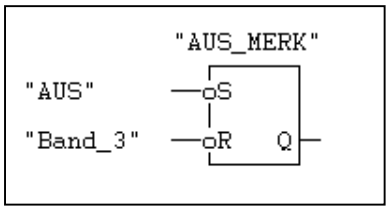
Aufgabe: Vorstehendes Projekt soll so umgebaut werden, dass über einen <AUS>-Taster alle Bänder 1.) leergefahren werden und 2.) zeitversetzt abschalten.

Sie können Ihr vorstehendes Projekt als Vorlage benutzen und davon eine Kopie anlegen, mit der Sie dann weiter arbeiten können.

Bisher blieben beim Ausschalten auf dem Band 1 immer Pakte liegen. Das soll jetzt geändert werden. Eingeschaltet wird das Magazin nach wie vor vom Band 1. Der <AUS>-Befehl wirkt aber nicht mehr auf das Band 1, sondern direkt auf das Magazin. Für diese Bedingung ist eine Verknüpfung erforderlich. Aus dem Tastbefehl mit einer flüchtigen „0“ muss eine dauerhafte Information abgeleitet werden, dass der <AUS>-Taster betätigt wurde.

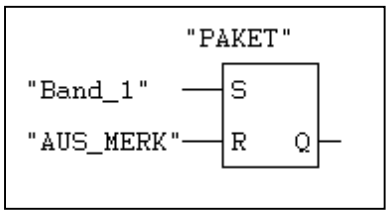
Die Symboltabelle wird um eine Zeile ergänzt:

AUS_MERK	M10.2	BOOL	Merker für <AUS>-Betätigung



Dieses „Gedächtnis“ wird gesetzt, wenn die Anlage ausgeschaltet wird.
Wenn zuletzt Band 3 ausschaltet, wird das FF zurück gesetzt.

Das Netzwerk für den Paketstart wird geändert:

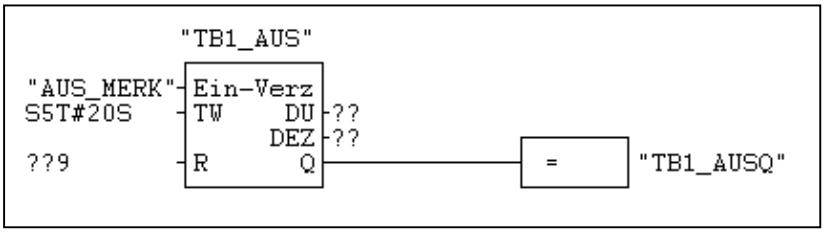


Wenn „AUS_MERK“ einschaltet, liegt am <R>-Eingang für die Paketfreigabe dominierend eine “1“, so dass keine Pakete mehr frei gegeben werden, obwohl „Band_1“ noch eine „1“ an <S> legt.

Das Band 1 soll beim Abschalten leer sein. Deshalb ist es über einen Timer abzuschalten.
Der neue Timer wird wieder in einem neuen Netzwerk angelegt.

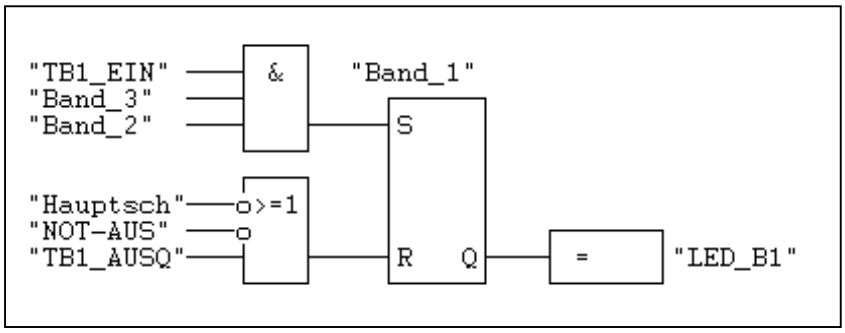
Vorher wird die Symboltabelle erweitert:

TB1_AUS	T 4	Timer	Einschaltverzögerung AUS Band 1
TB1_AUS_Q	M 10.3	BOOL	Ausgang TB1_AUS (Klemme Q)
TB2_AUS	T 5	Timer	Einschaltverzögerung AUS Band 2
TB2_AUS_Q	M 10.4	BOOL	Ausgang TB2_AUS (Klemme Q)
TB3_AUS	T 6	Timer	Einschaltverzögerung AUS Band 3
TB3_AUS_Q	M 10.5	BOOL	Ausgang TB3_AUS (Klemme Q)

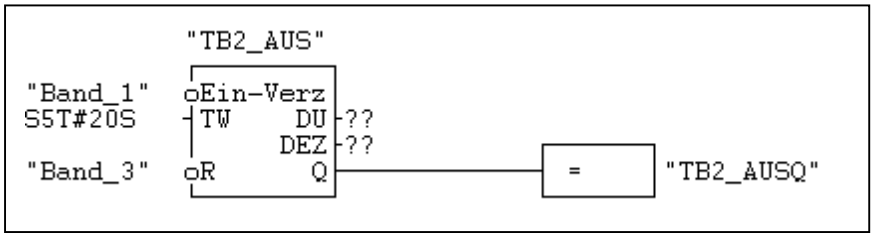


„AUS_MERK“ liefert die Dauer-„1“ für den einschaltverzögerten Timer, mit dem Band 1 ausgeschaltet werden soll.

In dem Netzwerk für den Band_1-Antrieb wird die Abschaltbedingung geändert.



Wenn Band 1 steht, wird der Abschalt-Timer für Band 2 gestartet.

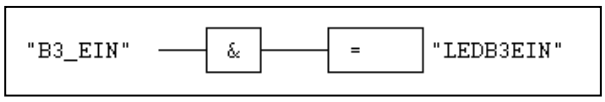


Hier muss dafür gesorgt werden, dass der Timer abgeschaltet wird, wenn er nicht mehr benötigt wird. Sonst würde er dauerhaft „1“ liefern, d.h., im Netzwerk für Band 2 (siehe dort) würde an <R> dominierend „1“ liegen. Band 2 wäre nicht wieder einzuschalten.

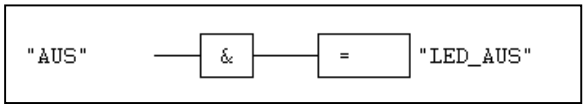
Band 3 wird in gleicher Weise zeitverzögert ausgeschaltet.

Es folgt jetzt die Auflistung der endgültigen Netzwerke.
Vgl. TrySim <Lösungen>|<F_Band_SRT_auto>.

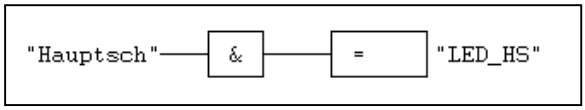
OB1, Netzwerk 1: Zuweisung der Kontrollleuchte



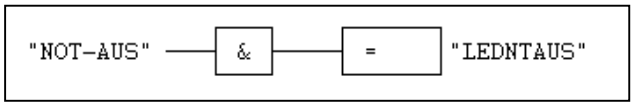
OB1, Netzwerk 2: Zuweisung der Kontrollleuchte



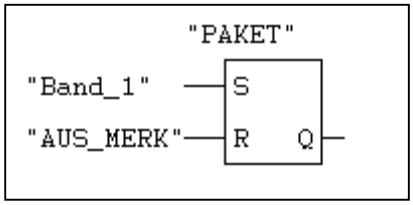
OB1, Netzwerk 3: Zuweisung der Kontrollleuchte



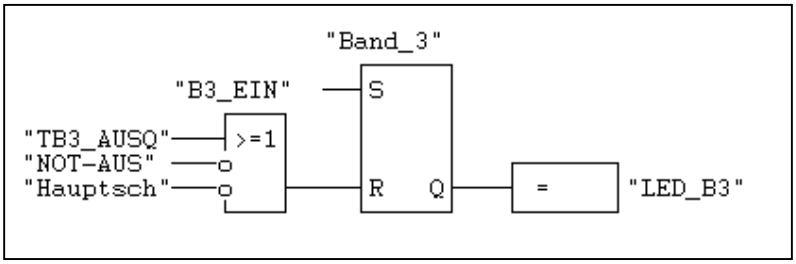
OB1, Netzwerk 4: Zuweisung der Kontrollleuchte



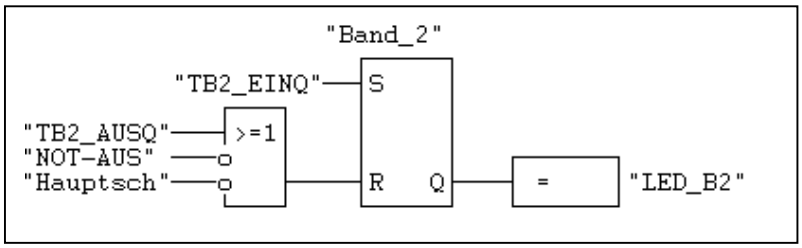
OB1, Netzwerk 5: Abschaltung Magazin



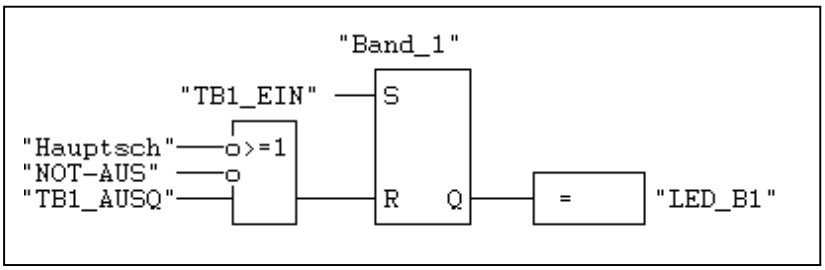
OB1, Netzwerk 6: Ansteuerung Band 3



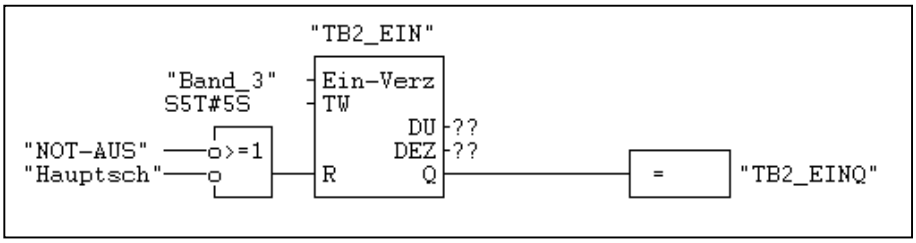
OB1, Netzwerk 7: Ansteuerung Band 2



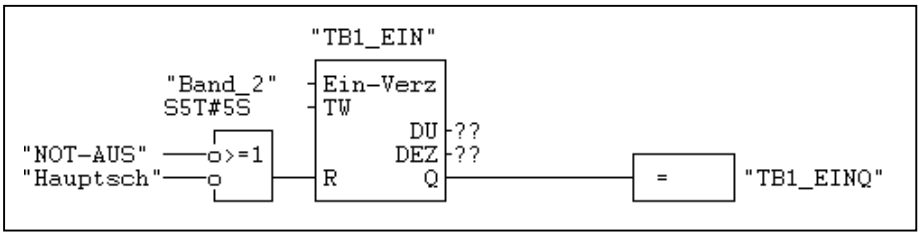
OB1, Netzwerk 8: Ansteuerung Band 1



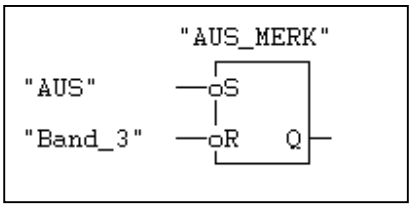
OB1, Netzwerk 9: Timer Band 2 EIN



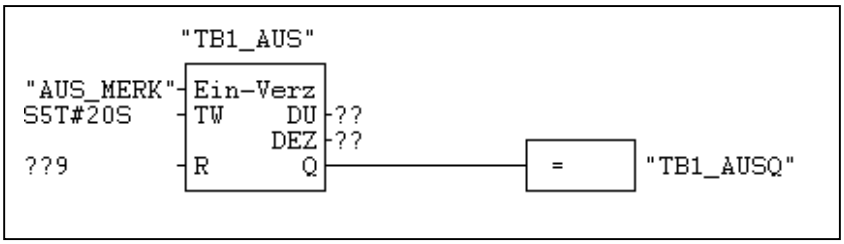
OB1, Netzwerk 10: Timer Band 1 EIN



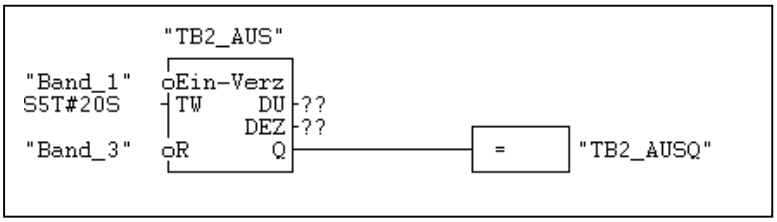
OB1, Netzwerk 11: Merker für Magazin



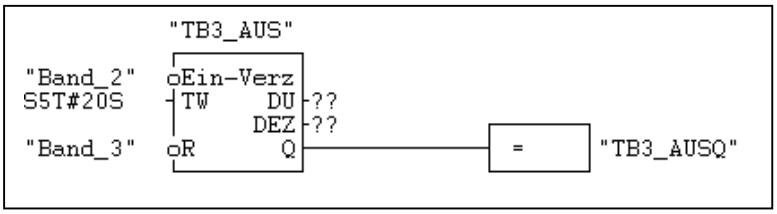
OB1, Netzwerk 12: Abschaltung von Band 1



OB1, Netzwerk 13: Abschaltung Band 2



OB1, Netzwerk 14: Abschaltung Band 3

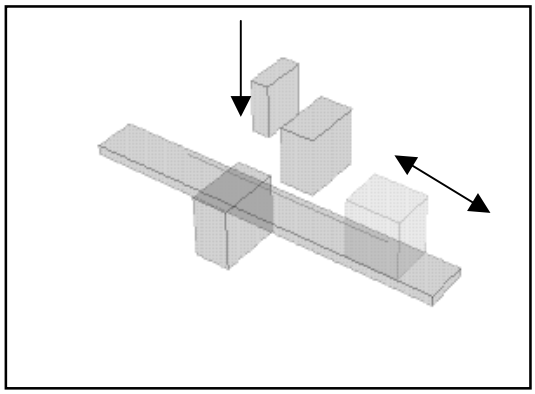


Projekt 16: Durchfahrt / Torsteuerung

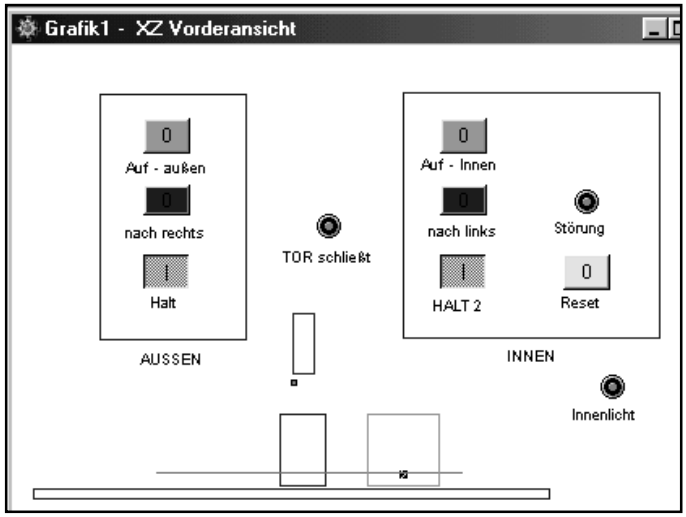
Schwerpunkte: Wendeschützschaltung, Zeitglied, RS-FF,

Torsteuerung mit Sicherheitsleiste

Ein Gegenstand (grüner Kasten) soll auf einem Förderband von rechts nach links - oder umgekehrt - durch kurzen Tasterdruck bewegt werden. Dabei muss ein Rolltor passiert werden. Die Bedienung kann von beiden Seiten aus erfolgen. Es soll ein direkter Richtungswechsel möglich sein. Es ist auszuschließen, dass beide Drehrichtungsschütze gleichzeitig eingeschaltet werden können. Die sogenannte Tasterverrieglung soll nur durch die Abfrage der beiden <EIN>-Taster erreicht werden. Beim Startbefehl muss zuerst das Tor automatisch öffnen. Auf der anderen Seite stoppt der Kasten und das Tor schließt zeitverzögert.
Im Störungsfall können Band und Tor mit den <HALT>-Tastern gestoppt werden. Berührt die Sicherheitsleiste die Kiste, muss das Tor sofort wieder hoch fahren und dort bleiben.
Erst nach Quittierung mit der <RESET>-Taste schließt dann das Tor.
Das Tor kann auch von beiden Seiten geöffnet werden, ohne dass ein Transport durch das Tor erfolgen muss.
Beim Schließen leuchtet die Lampe <TOR schließt>. Nach dem Schließen leuchtet eine gewisse Zeit die Innenbeleuchtung.
Es empfiehlt sich, <Grafik XZ> und <3D> zu öffnen



Grafik <3D>

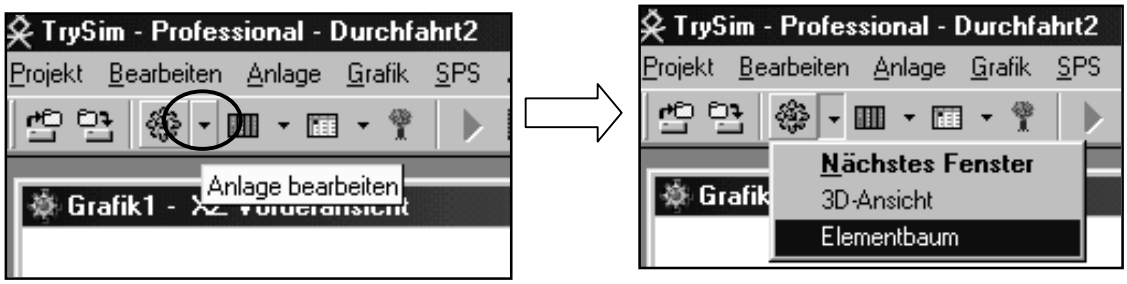


Symboltabelle

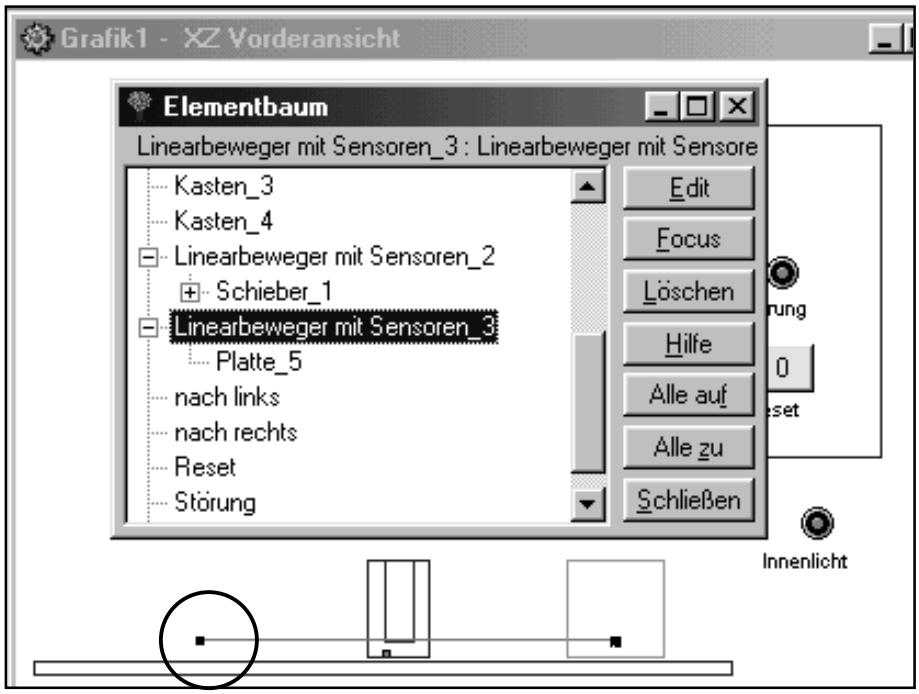
Nummer	Symbol	Adresse	Typ	Kommentar
1	Stoerung	A 0.0	BOOL	Kontrolle Störung
2	InnenL	A 0.1	BOOL	Innenlicht
3	TorZuL	A 0.4	BOOL	Tor schließt (Lampe)
4	TorZu	A 0.5	BOOL	Tor schließt (Antrieb)
5	TorAuf	A 0.6	BOOL	Tor öffnet (Antrieb)
6	rechts	A 0.7	BOOL	Band nach rechts
7	links	A 1.0	BOOL	Band nach links
8	SchL	E 0.0	BOOL	Schutzleiste
9	Halt	E 0.2	BOOL	HALT_Befehl außen (Ö)
10	AufIn	E 0.3	BOOL	Taster (S) Tor öffnen (innen)
11	HaltIn	E 0.4	BOOL	HALT_Befehl innen (Ö)
12	AufAuß	E 0.5	BOOL	Taster (S) Tor öffnen (außen)
13	Reset	E 0.6	BOOL	Rücksetztaster bei Störung
14	BegrL	E 0.7	BOOL	Endschalter links
15	BegrR	E 1.1	BOOL	Endschalter rechts
16	BegrO	E 1.2	BOOL	Endschalter oben
17	BegrU	E 1.3	BOOL	Endschalter unten
18	nachR	E 2.0	BOOL	Taster (S) Band nach rechts
19	nachL	E 2.1	BOOL	Taster (S) Band nach links

Die Anlage und die Symboltabelle sind vorbereitet unter <Vorlagen>|<Durchfahrt_V>. Sie können dieses Projekt öffnen und in Ihr Arbeitsverzeichnis kopieren, um es dort zu programmieren.

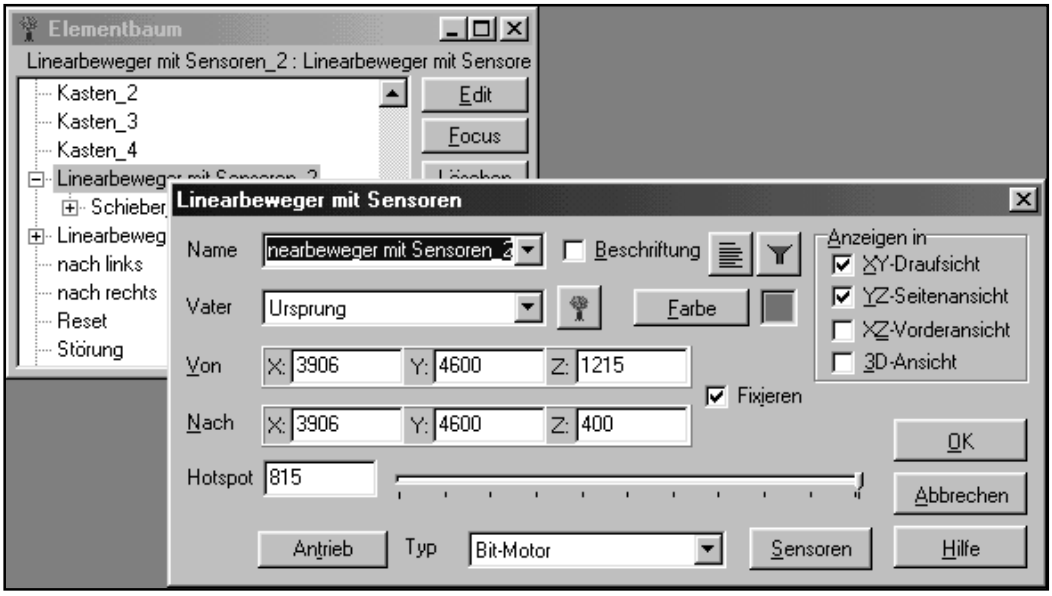
Bevor wir zur eigentlichen Programmierung kommen ist zu klären, wie das Tor bzw. der Kasten angetrieben werden. Als Antriebe wurden "Linearantriebe mit Sensoren" ausgewählt und installiert. In diesem Element sind die Endtaster gleich mit integriert. Wenn Sie dieses Element in der Grafikdarstellung nicht finden, weil z.B. im Editierfenster des Elements unter <Anzeigen in> kein Häkchen gesetzt wurde, können Sie das Icon <Anlage bearbeiten> und dort den Pfeil(!) drücken.



Sie können den Elementbaum öffnen und mit <LM> ein beliebiges Element markieren. Diese Markierung wird in Anlage ebenfalls sichtbar (siehe Abbildung).



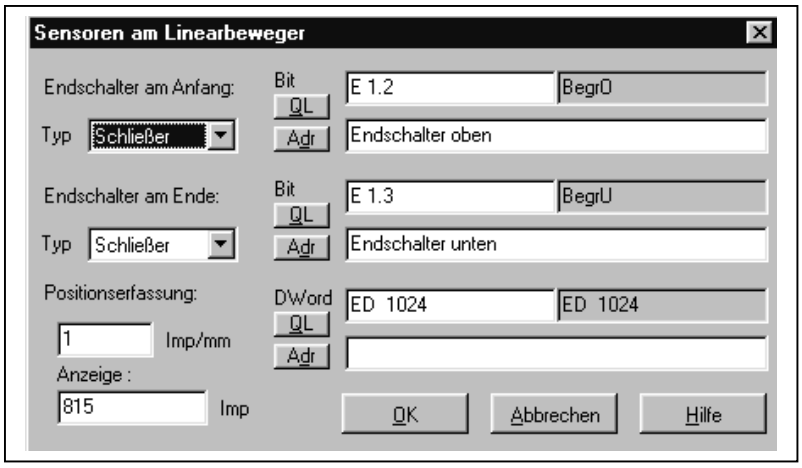
Nun kann es aber sein, dass Sie dort gar nichts finden, wie z.B. beim <Linearbeweger mit Sensoren_2>. Das können Sie schnell ändern, indem Sie **bei ausgeschalteter Anlage** doppelklicken. (Sonst ist im Editierfenster das <OK> grau und damit inaktiv). Sie können jetzt im Editierfenster des Elements unter <Anzeigen in> für die gewünschte Darstellung des Elements ein Häkchen setzten.



Editiermaske des Linearbewegers mit Sensoren

Sie sehen dort unter <Anzeigen in> das Kästchen für <XZ-Vorderansicht> frei. Wenn Sie dort ein Häkchen reinklicken, und <OK> wählen, werden Sie den senkrechten Antrieb für das Rolltor sehen.

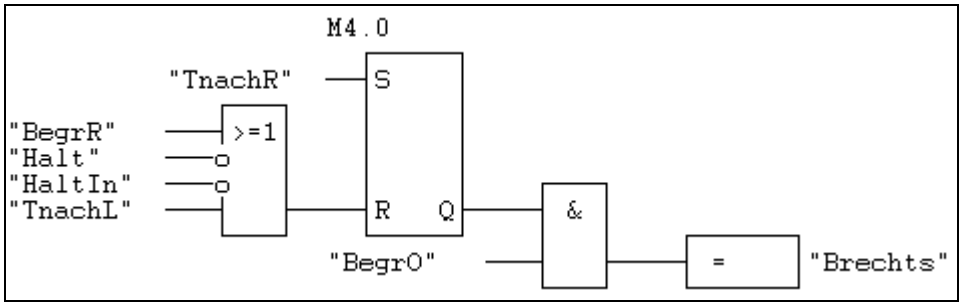
Unter <Sensoren> finden Sie die vorgeschlagenen Endtaster-Operanden, die natürlich veränderbar sind. Ebenfalls erkennen Sie hier die Funktion der Endschalter.



Ebenso finden Sie unter <Antrieb> die Ausgangs-Operanden.
Sie erinnern sich: bei ausgeschalteter Anlage können Sie auch mit der Maus die entsprechenden Operanden nicht nur sichtbar machen, sondern mit <LM> markieren und anschließend in dem SPS-Editierfenster – wiederum mit <LM> - den Operanden zuweisen..

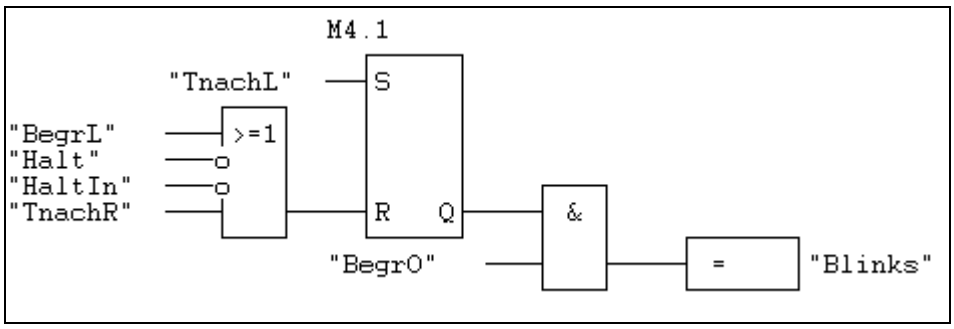
OB 1; Netzwerk 1: Bandsteuerung mit Endabschaltung nach rechts

Wenn der Taster <nach rechts> gedrückt wurde und das Tor geöffnet ist (<BegrO> = "1"), fährt das Band nach rechts, solange nicht der Endschalter <BegrR> erreicht wird, bzw. nicht <Halt> oder <HaltIn> gedrückt wird. Durch die Abfrage des Tasters <nach links> (unbetätigter Schließer = "0") wird gewährleistet, dass M 4.0 zurückgesetzt würde, wenn beide Taster zugleich betätigt würden.
Achten Sie bitte auf das Prinzip: Zum Setzen des SR-FF reicht ein kurzer "1"-Impuls. Danach ist der Signalzustand an <S> bedeutungslos. Aber alle Bedingungen, die zum Ausschalten führen sollen, müssen unabhängig voneinander wirken. Deshalb werden diese Bedingungen vor <R> ODER-verknüpft. Sie erinnern sich bestimmt: <R> wird mit "1" zurückgesetzt.

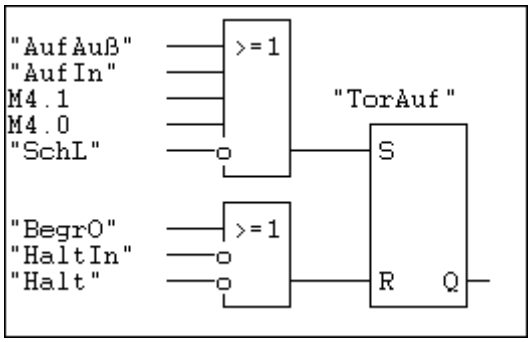


Normalerweise werden alle Bedingungen für einen Ausgang **vor** einem SR-FF programmiert. Warum nicht hier? Das hängt von der Aufgabenstellung ab. Das SR-FF ist als Merker M 4.0 programmiert und nicht direkt als Ausgang. Die Setz- und auch die Rücksetzbedingungen gelten unabhängig vom Rolltor. Erst wenn das Tor oben ist, darf der Kistenantrieb eingeschaltet werden. Der Merker M 4.0 ist also die Vorbereitung für den Antrieb <rechts>.

OB 1; Netzwerk 2: Bandsteuerung mit Endabschaltung nach links



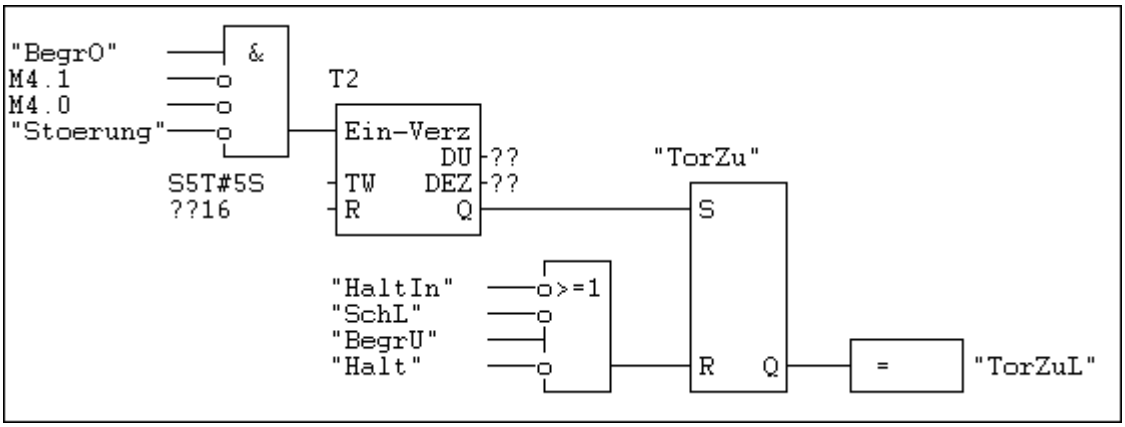
OB 1; Netzwerk 3: Tor öffnen



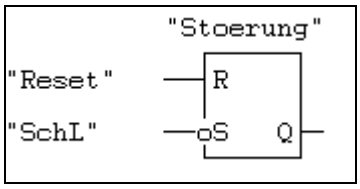
Bevor das Band gestartet wird (M 4.0 oder M 4.1), muss das Tor geöffnet sein. Zusätzlich kann es von Hand geöffnet werden ("AufAuß" oder "AufIn"). Beim Auslösen der Schutzleiste ("SchL") muss das Tor sofort öffnen.
Gestoppt wird der Vorgang durch <Halt> oder beim Erreichen der Endlage ("BegrO").

OB 1; Netzwerk 4: Tor automatisch schließen

Wenn das Tor den oberen Endtaster erreicht hat (BegrO" = "1"), das Band nicht mehr läuft und keine Störung vorliegt, fährt das Tor nach 5 s wieder runter. Gestoppt wird der Vorgang beim Erreichen der unteren Endlage, bei <Halt> oder falls die Schutzleiste anspricht.



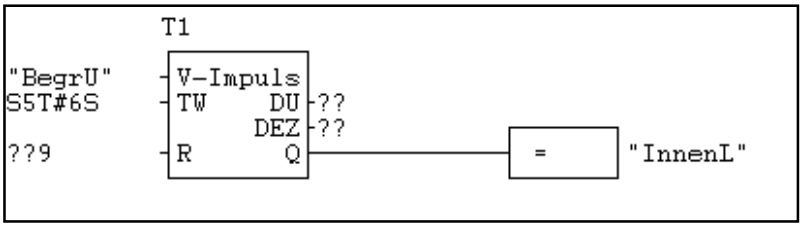
OB 1; Netzwerk 5: Störungsanzeige.



Ist bei der Abwärtsfahrt dem Rolltor ein Hindernis im Wege, liefert die Sicherheitsleiste eine "0".
Die Störungslampe leuchtet, bis sie mit <RESET> quittiert wird.

OB 1; Netzwerk 6: Innenlicht

Nachdem das Tor geschlossen wurde, wird für 6 s das Innenlicht eingeschaltet.



Kritische Würdigung:

Überlegen Sie bitte, ob die Anlage ausreichend sicher und praxisgerecht programmiert wurde. Dieses Buch kann und will nicht in jedem Fall eine vollständige, praxistaugliche Lösung anbieten, sondern fordert vom Leser und Anwender ein gehöriges Maß an Eigenverantwortlichkeit und Sensibilität für möglich Schwäche und Störungen. Es reicht eben nicht aus, die gewünschte Funktion zu erfüllen. Grundsätzlich muss von der ungünstigsten Fehlerlage ausgegangen werden. Ist Ihr Programm dann noch in der Lage, den Fehler zu beherrschen?
Bei diesem Projekt ist zumindest noch an den Not-Aus-Schalter zu denken. Ferner ist zu prüfen, ob die Wahl der Endtaster richtig ist (Schließer oder Öffner?). Bei einem Motorantrieb müsste der Überstromauslöser berücksichtigt werden.

Vgl. TrySim <Lösungen>|<Durchfahrt_>

Projekt 17: Zahlensysteme

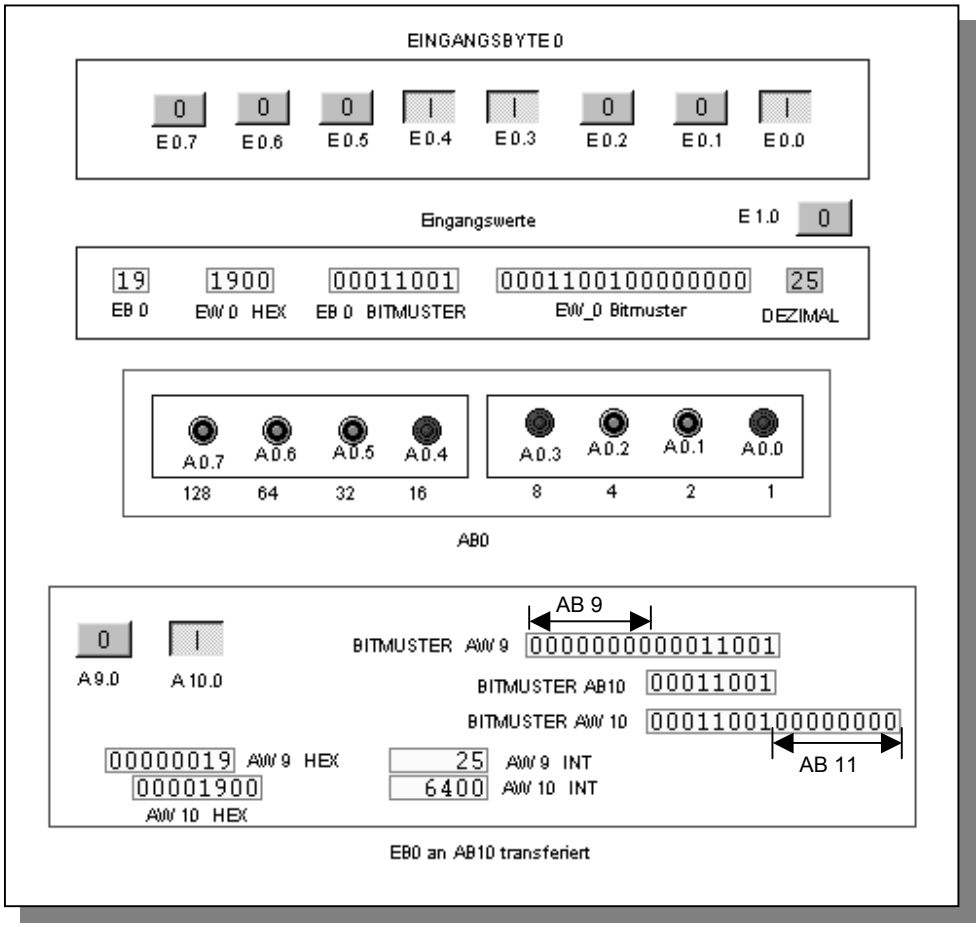
Schwerpunkte: Byte, Word, Zahlensysteme, Darstellung von DUAL- und BCD-Zahlen

Aufgabe:
Diesmal geht es nicht um die Programmierung eines Projektes, sondern um das Hintergrundwissen für die Darstellung und Verarbeitung von Zahlen.
Vergleichen Sie die unterschiedlichen Zahlencodes DUAL und BCD.

Theoretische Erörterung der Aufgabe:
Diese Aufgabe entspricht sicherlich nicht einer praktischen Anwendung. Sie zeigt, worauf z.B. beim Einsatz eines Zählerbausteins zu achten ist.
Wenn man Zähler nicht nur nach "Kochrezept" verwenden will - was nicht empfehlenswert ist, wie wir sehen werden - bleibt Ihnen leider nicht erspart, etwas über verschiedene Zahlendarstellungen (Codes) und die Byte- bzw. Worddarstellung zu erfahren.
Deshalb laden Sie bitte zur Begleitung der weiteren Erläuterung das fertige TrySim-Projekt <Zähler_2> aus dem <Lösungen>-Ordner. Ich verzichte vorläufig auf die Erklärung des Projektes und will anhand der fertigen "Anlage" nur die Begriffe "Bit", "Byte", "Word" und "transferieren" verdeutlichen.

Sie finden bei geöffnetem Projekt folgende Anlage vor:
Hier in der Abbildung sind schon einige Eingangsschalter gesetzt worden. Ihre Pegelverarbeitung soll jetzt erklärt werden.

<Zähler_2> im <Lösungen>-Ordner



Die 8 Schalter sind den Eingangsoperanden E 0.0 bis E 0.7 zugeordnet. Bisher wurden in den Projekten die Operanden jeweils einzeln abgefragt. Für reine Steuerungsaufgaben ist das auch die richtige Lösung.

Byte-Verarbeitung

Bei Steuerungen, in denen zahlenmäßig bewertete Informationen verarbeiten werden sollen, z.B. Temperaturen oder Stückzahlen, bei dem Vergleich zweier Werte oder bei der Berechnung der Differenz (Soll-/ Istwertvergleich), benutzt man Zahlensysteme, die aus einer mehrstelligen Kombination aus "0" und "1" bestehen.

Bei der gemeinsamen Auswertung von 8 Stellen erreicht man $2^8 = 256$ verschiedene Kombinationen, die man je nach Zahlensystem unterschiedlich bewertet.

Allgemein: 2^n = Zahl der Kombinationen, wobei n die Anzahl der Stellen ist.

Es ist ganz wichtig zu wissen, wie (in welchem Zahlensystem) ein bestimmtes Bitmuster zu deuten ist. Hier können aus Platzgründen die Zahlensysteme nicht ausführlich dargestellt werden. An anderer Stelle wurden sie bereits kurz erläutert.

Die Zusammenfassung einer Klemmenreihe mit gleichen Einschubnummern wird **Byte** bezeichnet. Der Computer erkennt das gesamte Byte der Klemmen E 0.0 bis E 0.7 durch den neuen Operanden EB 0.
Das Ausgangsbyte der Klemmen A3.0 bis A3.7 heißt sinngemäß: AB 3. Auch Merkerbytes können verwendet werden. Wenn man 1 Byte benutzt (z.B. MB 2), macht es keinen Sinn, einem einzelnen Bit (z.B. M 2.1) ein Verknüpfungsergebnis (VKE) zuzuweisen, denn dieses Bit würde ja den Wert des Bytes verfälschen.
Die Bits werden von 0 bis 7 nummeriert, wobei Bit 7 das höchstwertige Bit ist (MSB most significant bit).

Beispiel für die Abfrage des Byte EB0:

	MSB							LSB
Operand	E 0.7	E 0.6	E 0.5	E 0.4	E 0.3	E 0.2	E 0.1	E 0.0
Wert	128	64	32	16	8	4	2	1
Zustand	0	1	0	1	1	0	0	1
Wert:	0 +	64 +	0 +	16 +	8 +	0 +	0 +	1

Die Dualzahl 1011001, die als Byte EB0 erfasst wird, entspricht dezimal dem Wert 89.

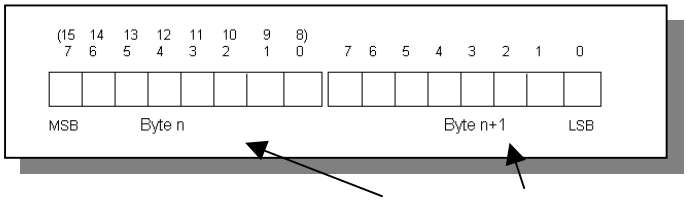
Wenn mehr als 256 verschiedene Zahlen / Zustände unterschieden werden müssen, werden weitere Bit-Stellen benötigt. Der Computer kann in ebenso einfacher Weise wie beim Byte-Befehl gleich 2 Bytes zusammen auswerten, z.B. mit den Operanden

- EW 0 für das Eingangswort 0,
- AW 2 für das Ausgangswort 2,
- MW3 für das Merkerwort 3.

Wortverarbeitung

Die Wortdarstellung ist wegen der größeren Auflösung ($2^{16} = 65536$ Kombinationen) wichtig für die **Verarbeitung von Analogwerten**. Da der Rechner und die SPS nur Kombinationen von "0" und "1" verarbeiten kann, müssen analoge Werte - also z.B. beliebige Werte eines Sensors im Bereich von 0 bis 10 V - mit Hilfe eines A/D-Wandlers aufbereitet werden. Dieser A/D-Wandler könnte prinzipiell außerhalb der SPS liegen und mit seiner Wortbreite von 16 Bits 2 Eingangsbytes blockieren, oder es könnte über einen Analogeingang der SPS intern das Wort gebildet werden. Egal wie - verarbeitet wird nur die digitale Information.
Was ist ein Wort (word), und welche Bits gehören dazu?
Dieser Typ ist aus zwei Bytes zusammengesetzt.
Die oben erläuterte Zählsystematik der Stellenwertigkeit wird fortgeschrieben. Nach links nimmt die Wertigkeit zu. Zum Verständnis ist ganz wichtig: Das Wort bekommt die Nummer des kleineren Bytes.

Das kleinere Byte steht links! Das kleinere Byte ist höherwertig!



EW 0 bedeutet Byte EB **0** + Byte EB1.

Der kleinste Stellenwert im Sinne einer Dualzahl liegt im Bit 0 des Bytes 1 (LSB = least significant byte [niederwertigst]) , der größte Stellenwert im Sinne einer Dualzahl liegt im Bit 7 des Byte 0 (MSB = most significant byte [höchtstwertigst]).

Die Dezimalzahl "1" wird repräsentiert durch eine "1" im Bit 0 des Bytes **1**. (1 x 2^0 = 1)
Die Dezimalzahl "4" wird repräsentiert durch eine "1" im Bit 2 des Bytes **1**. (1 x 2^2 = 4)
Die Dezimalzahl "128" wird repräsentiert durch eine "1" im Bit 7 des Bytes **1**. (1 x 2^7 = 128)
Die Dezimalzahl "256" wird repräsentiert durch eine "1" im Bit 0 des Bytes **0**. (1 x 2^8 = 256)
Die Dezimalzahl "32768" wird repräsentiert durch eine "1" im Bit 7 des Bytes **0**. (1 x 2^{15} = 32768)

Im Byte 0 des Wortes 0 liegen die höheren Stellenwerte der Dualzahl.

Wozu benötigt man das Wissen?
Zum Verständnis dafür, wo welche Information abgelegt wurde. Wenn Sie z.B. das Wort "EW 0" für die Darstellung einer Istwerterfassung verwenden wollen, dürfen Sie die Klemmen "E 1.0" bis "E 1.7" nicht mit weiteren Informationen belegen.. Sie wissen doch: Dieses Byte "EB 1" wird als Bestandteil des Wortes "EW 0" ausgewertet.
Testen Sie diesen Fall mit TrySim, indem Sie den zusätzlichen Schalter <E 1.0> betätigen. Sie müssen also höllisch aufpassen, wohin sie Ihre Bits verschieben!
Sie sehen in der Anzeige <EB 0 Bitmuster> die identische Darstellung der Eingangsleiste E 0.0 bis E 0.7.
Hierzu ist überhaupt kein Programm erforderlich, sondern in der "Anlage" wird die Digitalanzeige direkt auf "EB 0" gelegt, u.z. in der Editiermaske mit <rM> in der Darstellung (Codierung) "Byte - Bitmuster". Rechts daneben wird die Anzeige <EW 0 Bitmuster> mit der Zielbezeichnung (Operand) "EW 0" verbunden. Die Darstellung ist auf "Word - Bitmuster" geschaltet. Betrachten Sie bitte, wo im Wort das obige EB 0 abgelegt wurde und vergleichen Sie mit vorstehender Zuweisungsregel.

Im Wort Nr. x steht immer das Byte mit der nächsthöheren Bytenummer (x+1).
Dieses Byte wird automatisch mit abgefragt.

Dasselbe gilt auch für das davorliegende Wort! Mit den simplen Befehlen...

```
L EB 0      (nicht mehr mit "U" beginnen, sondern mit "L" = Lade das Byte abfragen)
T AB0       (und transferiere (weise zu...) an das Ausgangsbyte AB 0).
```

...wird die Information des Eingangsbytes an die Ausgangsleiste 0 gelegt, die wiederum aus lauter einzelnen Klemmen dargestellt wurde. Die Klemmen lassen sich als Stellen des Dualcodes interpretieren. Der Zahlenwert der AB 0-Klemmenleiste. entspricht der angezeigten Dezimalzahl, die vom AW 12 gebildet wurde. (L EB 0; T AW 12). Die Information vom EB 0 wird **rechtsbündig** ins AW 12 gelegt, kommt wertemäßig dort also richtig an. Die Anzeige ist als "INT" dargestellt.

AB 0 umfasst die Klemmen (Operanden) A 0.0 bis A 0.7. Hätten wir EB 0 nach A**W** 0 transferiert, wäre das Byte ebenfalls **rechtsbündig** in das AW 0 gelegt worden, d.h. ins AB 1! Und an den LEDs A 0.0 bis A 0.7 wäre es dunkel geblieben. Sie können das einfach überprüfen, indem Sie im Netzwerk 1 bei ausgeschaltetem Auge "B" gegen "W" austauschen (und hinterher das Auge wieder einschalten).

Wörter lassen sich "1 zu 1" übertragen. Das EW 0 wurde ins AW 10 transportiert / transferiert. Hüten Sie sich aber davor, die benachbarten Wörter zu benutzen. Wie Sie sehen können, wird z.B. AW 9 automatisch mit der Information aus AW 10 belegt, obwohl AW 9 gar nicht programmiert wurde:

In EW 0 liegt EB 0 links und wird nach AW 10 auch links übernommen als AB 10. AB 10 gehört aber als rechtes Byte auch zu AW 9. Alles klar? Sehen Sie sich die "Anlage" an.

Der Wert von EB 0 wurde in / durch EW 0 verändert, weil sich der Stellenwert geändert hat! (EB 0 liegt links). Da EW 0 nach AW 10 transferiert wurde, kann hier nicht der richtige Eingangswert dargestellt werden, wohl aber in AW 9, da hier die Information rechtsbündig anliegt. Vergleichen Sie bitte die INT-Anzeigen (Dezimalwerte) von AW 9 und AW 10.

Ein weiterer wichtiger Punkt ist die Darstellung des Wertes. Hierfür kann man verschiedene Zahlencodes verwenden.
In Step®7 werden bei Zählerbausteinen im FUP und KOP am Ausgang "DU" der Code "Dualzahl" und am Ausgang "DE" der "BCD"-Code erzeugt. (Angezeigt wird zwar bei beiden Ausgängen die Dezimalzahl).

Erläuterungen zum BCD-Code:
BCD-codiert heißt binär codierte Dezimalzahl, d.h. der Zahlenwert wird ziffernweise erfasst und jeweils als Dualzahl dargestellt.

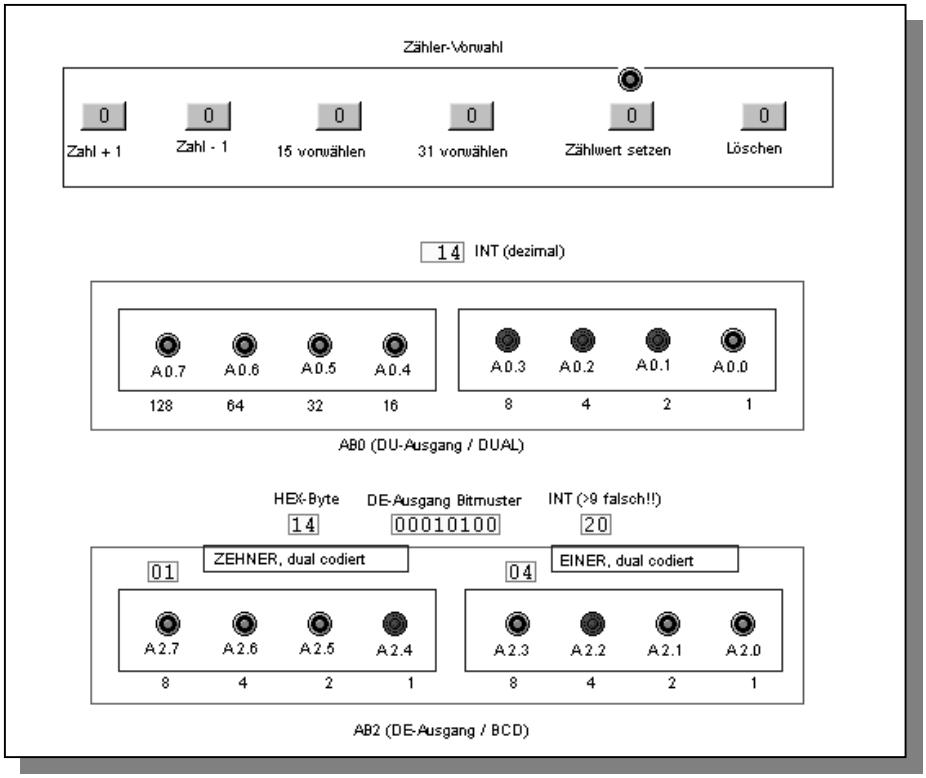
Beispiel: Die Dezimalzahl 36 wird dargestellt als
3 (x Zehn) **und** **6** (x Einer).

Die Zahl 3 allein entspricht dual : 0011
Die Zahl 6 allein entspricht dual : 0110

Somit entspricht die Dezimalzahl 36 der BCD-Zahl 0011 0110, wie Sie leicht mit dem fertigen Programm <Zähler_1> aus dem <Lösungen>-Ordner ausprobieren können. Dieses Programm kann hier ohne Vorkenntnisse direkt benutzt werden, um den Code nachzuvollziehen und zu überprüfen.

Falls Sie die BCD-Zahl 0011 0110 als Dualzahl auswerten würden, kämen Sie zu einem ganz anderen (falschen!) Ergebnis.
Probieren Sie ruhig mit verschiedenen Werten im Eingangsbyte und betrachten Sie die unterschiedlichen Ergebnisse.

<Zähler_1> aus dem Ordner <Lösungen>



Projekt 18: Wo bleiben die Eingangsbits?

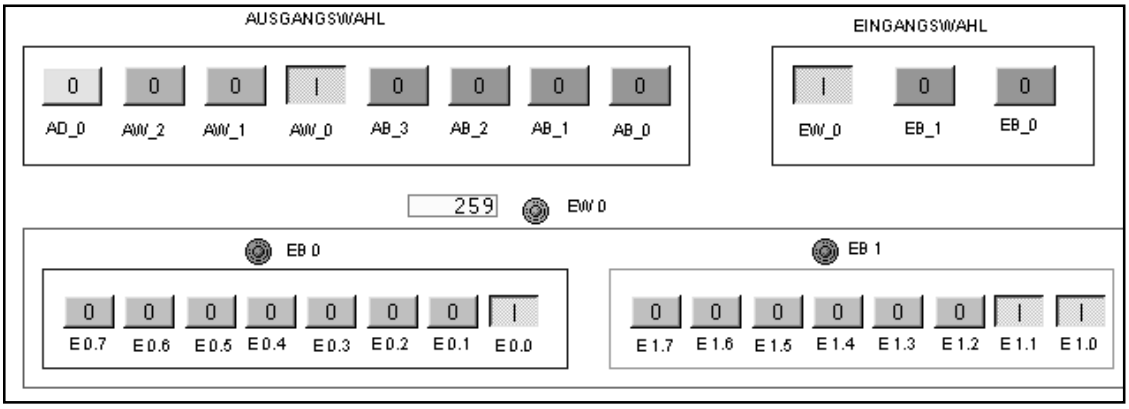
Schwerpunkte: Byte- und Wortverarbeitung, Schiebebefehl SLW.

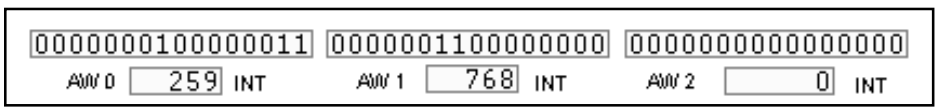
Anmerkungen:
Dieses Projekt dient nicht der Lösung eines tatsächlichen Anwendungsfalles, sondern ist als optische Erklärungshilfe gedacht um nachzuvollziehen, welches die Auswirkung sind, wenn ein Byte oder Wort transferiert wird.
Wahrscheinlich wird es für Sie ausreichen, wenn Sie den Datentransport beobachten und nachvollziehen können, um fehler- oder rätselhafte Datenveränderungen bei Ihrer Programmierung zu vermeiden. Für besonders interessierte Leser werde ich das Programm natürlich auch kommentieren. Laden Sie bitte aus dem Verzeichnis <Lösungen> das fertige Projekt <AKKU>. Im Projekt 17, <Zahlensysteme>, wurde bereits die Systematik der Byte- und Wortverarbeitung erläutert. In dem jetzigen Projekt haben Sie die weitergehende Möglichkeit, verschiedene Bytes und ein Wort zu transferieren.

Aufgabe:
- Eine Eingangsinformation (ein Bit, Byte oder Wort) soll zu einem Ausgangsbereich transferiert werden. Die Information darf dabei nicht verfälscht werden.
- Wählen Sie ein beliebiges Bitmuster an den Schaltern des EB 0 bzw. EB 1.
  (Solange Sie nicht mindestens 1 Bit der Bytes auf „1“ gesetzt haben, blinkt die zugehörige <Wahl>-LED).
- Jetzt müssen Sie entscheiden, ob ein Eingangsbyte oder das Eingangswort EW 0 geladen werden soll. Dazu betätigen Sie bitte einen Schalter der <EINGANGSWAHL>.
  (Auch hier blinkt eine <WAHL>-LED, sofern noch kein Schalter gedrückt wurde). Sie können die Wahl ändern, indem Sie auf einen anderen Schalter dieses Bereiches drücken.
- Anschließend entscheiden Sie durch eine Wahl im Bereich <AUSGANGSWAHL>, wohin die Dateninformation gesendet werden soll.
  (Die zu dem gewählten Bereich gehörenden Klemmen sind in Rahmen zusammengefasst. Eine entsprechende LED blinkt).
- Untersuchen Sie, in welchem Byte die Informationen aus EB 0 bzw. EB 1 abgelegt werden.
- Prüfen Sie, ob die Ausgangsinformation identisch ist mit der Eingangsinformation.

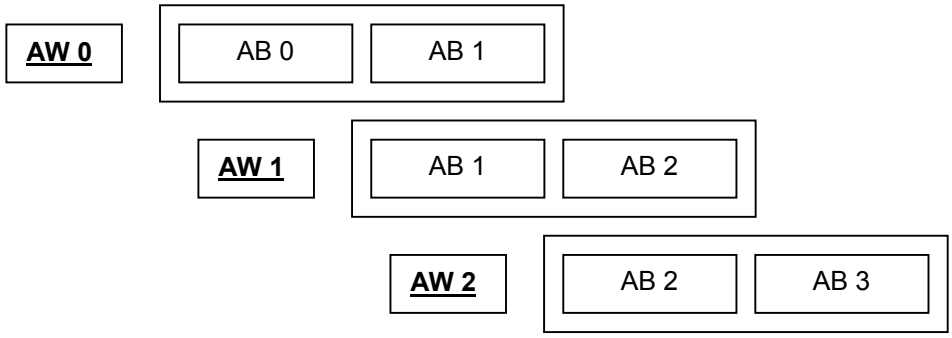
Beispiel: EB 0 allein entspricht dem Wert 1 und EB 1 entspricht dem Wert 3. Es soll das EW 0 an AW 0 transferiert werden.
Das EW 0 setzt sich aus EB 0 und EB 1 zusammen, wobei die Wertigkeit der Stellen durchgehend von ganz rechts (E 1.0, entspricht dem Wert 1) bis ganz links (E 0.6, entspricht dem Wert 16384) ansteigt. E0.7 entscheidet über das Vorzeichen.

Das niedrigere Byte ist höherwertig!

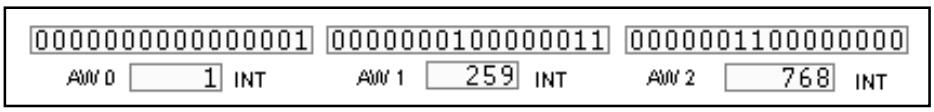




Sie erkennen, dass E 0.0 also eine unterschiedliche Wertigkeit erhält, je nachdem, ob der Operand im Byte allein, oder im Zusammenhang mit EB 1 - als Wort - interpretiert wird. Sie können dafür zwischen <EW_0> und <EB_1> wechseln und das AW 0 ablesen. Betrachten Sie beim Umschalten auch die Anzeige von EW 0, die sich nicht ändert. Der betätigte Schalter <EB_1> bedeutet lediglich, dass ein anderer Bereich (Byte 1) an AW 0 transferiert werden soll.
Sie können erkennen, dass nicht nur im AW 0 ein Wert anliegt, sondern auch im AW 1, obwohl im nachfolgenden Programm kein entsprechender Befehl programmiert wurde. Die Werte sind auch noch unterschiedlich. Wie kommt das?
Nun, die Bytes EB 0 und EB 1 werden in gewollter Weise zum AW 0 transferiert. Das A**W 0** umfasst die Bytes A**B 0** und A**B 1**. Im AB 1 liegt der Wert des EB 1. Zum AW 1 gehören nach gleicher Systematik die Bytes AB 1 und AB 2. Deshalb wird im AW 1 unweigerlich - unprogrammiert und ungewollt - ein (falscher) Wert angezeigt. Sie sehen, dass Sie höllisch aufpassen müssen, dass keine direkt aufeinander folgenden Worte programmiert werden.



Sie finden diese Behauptung bestätigt, wenn Sie anstatt <AW _0> jetzt <AW_1> drücken.



Der richtige Wert steht jetzt in der Anzeige des AW 1. Links (als AB 1) wird EB 0 und rechts (als AB 2) wird EB 1 abgebildet. Im AW 0 liegt rechts ebenfalls AB 1. Deshalb wird als Wert von AW 0 jetzt <1> angezeigt. AB 2 gehört auch zum AW 2. Dadurch erklärt sich, dass in der Anzeige von AW 2 auch ein - falscher - Wert erscheint.

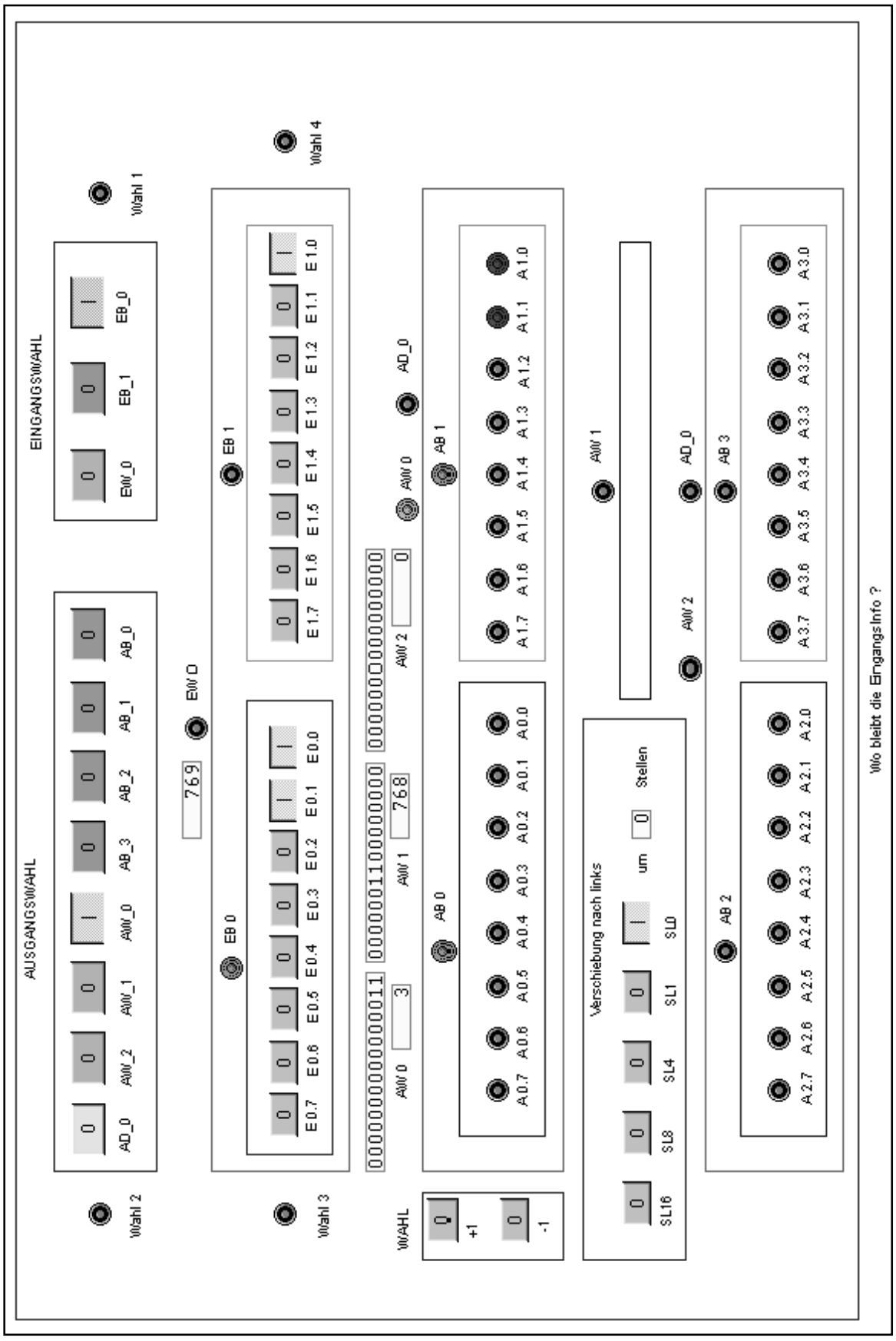
- Sie können auch untersuchen, welch ein Unsinn entsteht, wenn man das EW 0 in das AB0 „pressen“ wollte.

- Sie können das Ausgangsbitmuster mit den Schaltern <SL0> bis <SL16> um die entsprechenden Stellen nach links verschieben, oder mit <+1> oder <-1> um jeweils eine Stelle nach links oder rechts.

Als Ergänzung zu diesen Erläuterungen können Sie auch intensiv die TrySim-Hilfe benutzen. Dieses Buch ist keine Kopie der wichtigen Programm-Erläuterungen, sondern ist als Leitfaden gedacht für eine mir sinnvoll erscheinende Reihenfolge, nach der man seine Kenntnisse systematisch und hoffentlich lustvoll vertiefen kann.

Es folgt das Programm für diese Testanordnung.

Anlage <AKKU>


EINGANGSWAHL
AUSGANGSWAHL
EW_0
EB_1
EB_0
AD_0
AW_2
AW_1
AW_0
AB_3
AB_2
AB_1
AB_0
Wahl 1
Wahl 2
Wahl 3
Wahl 4
EW 0
EB 0
EB 1
AB 0
AB 1
AB 2
AB 3
AW 0
AW 1
AW 2
769
768
3
WAHL
+1
-1
Verschiebung nach links
um
Stellen
SL16
SL8
SL4
SL1
SL0
Wo bleibt die Eingangsinfo ?

Symbolische Adresse	Operand	Datentyp	Kommentar
Wahl_1	A 4.0	BOOL	LED Eingangswahl tätigen
Wahl_2	A 4.1	BOOL	LED Ausgangswahl tätigen
Wahl_3	A 4.2	BOOL	LED Wahl EB0 tätigen
Wahl_4	A 4.3	BOOL	LED Wahl EB1 tätigen
EW0	A 4.4	BOOL	LED Wahl EW_0
EB0	A 4.5	BOOL	LED Wahl EB_0
EB1	A 4.6	BOOL	LED Wahl EB_1
AW0	A 4.7	BOOL	LED Wahl AW_0
AB1	A 5.0	BOOL	LED Wahl AB_1
AB0	A 5.1	BOOL	LED Wahl AB_1
AW1	A 5.2	BOOL	LED Wahl AW_1
AB3	A 5.3	BOOL	LED Wahl AB_3
AB2	A 5.4	BOOL	LED Wahl AB_2
AW2	A 5.5	BOOL	LED Wahl AB_1
AD01	A 5.6	BOOL	LED Wahl AD_0
AD02	A 5.7	BOOL	LED Wahl AD_0
EB_0	E 2.0	BOOL	Schalter Wahl EB_0
EB_1	E 2.1	BOOL	Schalter Wahl EB_1
EW_0	E 2.2	BOOL	Schalter Wahl EW_0
AB_0	E 2.3	BOOL	Schalter Wahl AB_0
AB_1	E 2.4	BOOL	Schalter Wahl AB_1
AW_0	E 2.5	BOOL	Schalter Wahl AW_0
AB_2	E 2.6	BOOL	Schalter Wahl AB_2
AB_3	E 2.7	BOOL	Schalter Wahl AB_3
AW_1	E 3.0	BOOL	Schalter Wahl AW_1
AW_2	E 3.1	BOOL	Schalter Wahl AW_2
SL8	E 3.2	BOOL	Schiebe 8 Stellen nach links
SL4	E 3.3	BOOL	Schiebe 4 Stellen nach links
SL1	E 3.4	BOOL	Schiebe 1 Stellen nach links
SL0	E 3.5	BOOL	Schiebe 0 Stellen nach links
AD_0	E 3.6	BOOL	Schalter Wahl AD_0
SL16	E 3.7	BOOL	Schiebe 16 Stellen nach links
mehr	E 4.0	BOOL	Verschiebung +1
weniger	E 4.1	BOOL	Verschiebung -1
Z_SETZEN	M 4.0	BOOL	Stellenzähler setzen
max	M 6.0	BOOL	max. Verstellung
Flank_M0	M 50.0	BOOL	Flankenmerker
Flank_M1	M 50.1	BOOL	Flankenmerker
Flank_M2	M 50.2	BOOL	Flankenmerker
Flank_M3	M 50.3	BOOL	Flankenmerker
Flank_M4	M 50.4	BOOL	Flankenmerker
VerschS	MW 1	INT	Setzwert der Verschiebung
VerschDU	MW 7	INT	Zahlenwert DEZ-codiert Schieben
VerschDE	MW 10	INT	Zahlenwert BCD-codiert Schieben
AUSWERT0	MW 30	INT	Ausgangswert ohne Verschiebung

OB 1; Netzwerk 1: Zuordnung der Taster zu den LEDs der Bytes und Wörter

```
U  "EB_0"          //Wenn der Schalter <EB_0> betätigt wurde...
O  "EW_0"          //...oder der Schalter <EW_0>, ...
=  "EB 0"          //...dann setzte die LED <EB 0> auf logisch 1.
U  "EB_1"
O  "EW_0"
=  "EB 1"
U  "EW_0"
=  "EW 0"
U  "AB_0"          //Wenn der Schalter <AB_0> betätigt wurde...
O  "AW_0"          //...oder der Schalter <AW_0> ...
```

```
    O  "AD_0"          //...oder der Schalter <AD_0> ...
    =  "AB 0"          //...dann setzte die LED <AB 0> auf logisch 1.
                       (<AB_0> gehört zu all den vorgenannten Ausdrücken)
    U  "AB_1"
    O  "AW_0"
    O  "AW_1"
    O  "AD_0"
    =  "AB 1"
    U  "AB_2"
    O  "AW_1"
    O  "AW_2"
    O  "AD_0"
    =  "AB 2"
    U  "AB_3"
    O  "AW_2"
    O  "AD_0"
    =  "AB 3"
    U  "AW_0"
    O  "AD_0"
    =  "AW 0"
    U  "AW_1"
    =  "AW 1"
    U  "AW_2"
    O  "AD_0"
    =  "AW 2"
    U  "AD_0"
    =  "AD01“
    =  "AD02"
```

OB 1; Netzwerk 2: Anzeige der Stellenverschiebung über MW 1 ("VerschS")

Ohne jede Vorbedingung wird das der Wert 0 zum Merkerwort MW 1 transferiert, sozusagen als Grundstellung. Damit wird in der FC 2 ein Zähler gesetzt. Dort kann der Wert noch verändert werden. Dieser Wert wird dann in der Digitalanzeige dargestellt.

```
    L  0               // Lade die Zahl 0 und ...
    T  "VerschS"       // transferiere sie zum Merkerwort MW 1.
```

Jetzt beginnt die Zuordnung von anderen Zahlen / Werten des MW 1 zu den Schalterstelllungen.

```
    U  "SL0"           // Wenn  der Schalter <SL0> betätigt ist...
    SPBN  _1           // (falls nicht, springe zur Sprungmarke _1:)
    L  0               //... lade die Zahl 0 und ...
    T "VerschS"        //... transferiere sie zum Merkerwort MW 1.
    SPA  _x            // Springe Absolut (ohne Bedingung) zur Sprungmarke _x:.
_1:  U  "SL1"
    SPBN  _4
    L  1
    T  "VerschS"
    SPA  _x
_4:  U  "SL4"
    SPBN  _8
    L  4
    T  "VerschS"
    SPA  _x
_8:  U  "SL8"
    SPBN  _16
    L  8
    T  "VerschS"
_16: U  "SL16"
    SPBN  _x
    L  C# 16
    T  "VerschS"
_x:  CALL FC 2  // Aufruf der Funktion FC 2.
```

OB 1; Netzwerk 3: Blinklicht-Steuerung

Wenn noch keine Bereichstaste gedrückt wurde, soll die entsprechende Blink-LED leuchten.

```
UN E   0.0
UN E   0.1
UN E   0.2
UN E   0.3
UN E   0.4
UN E   0.5
UN E   0.6
UN E   0.7
=  "Wahl_4"

UN E   1.0
UN E   1.1
UN E   1.2
UN E   1.3
UN E   1.4
UN E   1.5
UN E   1.6
UN E   1.7
=  "Wahl_3"

UN "EB_0"
UN "EB_1"
UN "EW_0"
=  "Wahl_1"

UN "AB_0"
UN "AB_1"
UN "AW_0"
UN "AB_2"
UN "AB_3"
UN "AW_1"
UN "AW_2"
UN "AD_0"
=  "Wahl_2"
```

OB 1; Netzwerk 4: Zuordnung von Eingängen zu Ausgängen

```
L   0                  //Grundeinstellung. Alle Ausgangsbits werden auf "0"
                       //gesetzt.
T   AW  0              //Transferiere vorstehenden Wert nach AW 0.
T   AW  2              //Transferiere vorstehenden Wert nach AW 2.
```

AB1 bis AB 3 müssen nicht extra auf "0" gesetzt werden, da sie Bestandteil von AW 0 und AW 2 sind.

```
CALL FC 2              // Aufruf von FC 2. Von dort erfolgt anschließend der
                       // Rücksprung an diese Stelle.
```

Es folgt die Abfrage, welche Eingangskombination gedrückt wurde.

```
ein0: U  "EB_0"        // Wenn der Schalter <EB_0> gedrückt wurde...
      SPBN  ein1       // (Falls nicht, springe zur Marke <ein1:>)
      L   EB  0        // Lade das Eingangsbyte EB 0 und
      U  "AB_0"        // wenn Schalter <AB_0> gedrückt wurde...
      SPBN  aus1       // (Falls nicht, springe zur Marke <aus1:>)
      L   EB  0        // Lade EB 0 und
      T   "AUSWERT0"   // transferiere es zum MW 30.
                       // (Wert der nicht verschobenen Eingangsabfrage).
      CALL FC 1        // Aufruf von FC 1. (Verschiebefunktion)
                       // Von dort erfolgt anschließend der Rücksprung an diese Stelle.
      T   AB  0        // Transferiere den evtl. veränderten Wert an AB 0.
      BEA              // Baustein Ende Absolut (ohne Bedingung)
```

Die Abfrage wird so lange "weitergereicht", bis eine Bedingung erfüllt ist.

```
ein1: U  "EB_1"
      SPBN  ein2          // Falls nicht, springe zur nächsten Eingangsabfrage, usw.
      L   EB  1
      U  "AB_0"
      SPBN  aus1          // Falls nicht, springe zur nächsten Ausgangsabfrage, usw.
      L   EB  1
      T   "AUSWERT0"
     CALL FC 1
      T   AB  0
      BEA                 //Wenn das Programm bis hierher durchgelaufen wäre, ist hier das
                          // Netzwerk abgeschlossen.
```

Falls vorstehende Eingangsabfragen nicht gesetzt waren, bleibt nur noch diese Abfrage übrig.

```
ein2: U  "EW_0"  // Wenn das Eingangswort EW 0 abgefragt werden soll...
      U  "AB_0"          //... und als Ziel AB 0 gewählt wurde...
      L   EW  0          //... wird EW 0 geladen und ...
      SPBN  aus1         // (falls nicht, wird zur Marke <aus1:> gesprungen)
```

(Der eingeschobene Ladebefehl "L EW 0" hat keinen Einfluss auf die "SPBN"-Abfrage).

```
      T   "AUSWERT0"     //... nach MW 30 transferiert.
      CALL FC 1          // Aufruf von FC 1. (Verschiebefunktion)
                         // Von dort erfolgt anschließend der Rücksprung an diese Stelle.
      T   AB  0          // Transferiere den evtl. veränderten Wert an AB 0.
      BEA

aus1: U  "AB_1"          // Abfrage, ob die Eingangskombination nach AB 1 transferiert werden soll.
      SPBN  aus2         // (Falls nicht, springe zur Marke <aus2:>).
      T   "AUSWERT0"     // Transferiere den Eingangswert nach MW 30.
      CALL FC 1          // Aufruf von FC 1 und Rücksprung an diese Stelle.
      T   AB  1          // Transferiere den evtl. veränderten Wert an AB 1.
      BEA
aus2: U  "AB_2"          // usw, usw...
      SPBN  aus3
      T   "AUSWERT0"
      CALL FC 1
      T   AB  2
      BEA
aus3: U  "AB_3"
      SPBN  aus4
      T   "AUSWERT0"
      CALL FC 1
      T   AB  3
      BEA
aus4: U  "AW_0"
      SPBN  aus5
      T   "AUSWERT0"
     CALL FC 1
      T   AW  0
      BEA
aus5: U  "AW_1"
      SPBN  aus6
      T   "AUSWERT0"
      CALL FC 1
      T   AW  1
      BEA
aus6: U  "AW_2"
      SPBN  aus7
      T   "AUSWERT0"
      CALL FC 1
      T   AW  2
      BEA
aus7: U  "AD_0"
      SPBN  nix          // Wenn alles nicht passt - z.B. weil anfangs noch nichts
                         // ausgewählt wurde - wird das Netzwerk ohne FC 1-Aufruf beendet
```

```
    T  "AUSWERT0"
    CALL FC 1
    T  AD  0
nix:  NOP 0              //Das ist lediglich die Sprungmarke, um einen Programmteil zu überspringen.
```

FC 1; Netzwerk 1:

```
L  "VerschDE"        // Lade den Wert am Ausgang <DE> des Zählers Z 10 in der FC 2, in dem die
                     // gewünschte Verschiebung der Stellen festgelegt wird.
   L  0              // Lade die Zahl 0 als "Integer-Zahl".
```

(Vgl. zum Zahlenformat in der TrySim-Hilfe unter <Index > "Konstanten":

Integer-Konstante z.B. -5. Diese Konstante entspricht unseren ganzen Zahlen.
Der Wertebereich geht von -32.768 bis +32.767

Achtung: Dieses Zahlenformat wurde hier gewählt um zu zeigen, dass es bis zur Zahl 9 egal ist, ob eine Zahl als Integerzahl vorgegeben wird, oder BCD-codiert. Sie sind vom Code her ja identisch. "VerschDE" ist immer BCD-codiert!

```
   ==I               // Wenn beide Werte gleich sind...
   SPBN  _1          // (sonst springe zur Marke <_1:>).
   L  "AUSWERT0"     // Lade den Wert, der noch ohne Abfrage der Stellenverschiebung für die
                     // Ausgangsdarstellung bereit steht.
   SLW 0             // Verschiebebefehl zur Verschiebung um 0 Stellen.
```

(Da nicht verschoben werden soll, ist die Zeile eigentlich überflüssig. Sie ist nur wegen der Systematik und Vergleichbarkeit mit den anderen Sprungmarken hier dargestellt).

```
   BEA               // absolutes Baustein-Ende. Rückkehr zum aufrufenden Netzwerk.

_1:  L  "VerschDE"   // Der Vorgang wiederholt sich: Um wieviel Stellen soll verschoben werden?
   L  1              // Lade die Zahl 0 als "Integer-Zahl".
   ==I               // Wenn beide Werte gleich sind...
   SPBN  _2          // (sonst springe zur Marke <_2:>).
   L  "AUSWERT0"     // Lade den Wert, der noch ohne Abfrage der Stellenverschiebung für die
                     // Ausgangsdarstellung bereit steht.
   SLW 1             // Verschiebebefehl zur Wort-Verschiebung um 1 Stelle nach links.
   BEA
```

Sie merken schon: Der Ablauf wiederholt sich für jede mögliche Kombination, bis der Vergleicher "1" liefert.

```
_2:  L  "VerschDE"
   L  2
   ==I
   SPBN  _3
   L  "AUSWERT0"
   SLW 2
   BEA
_3:  L  "VerschDE"
   L  3
   ==I
   SPBN  _4
   L  "AUSWERT0"
   SLW 3
   BEA
_4:  L  "VerschDE"
   L  4
   ==I
   SPBN  _5
   L  "AUSWERT0"
   SLW 4
   BEA
_5:  L  "VerschDE"
   L  5
   ==I
   SPBN  _6
```

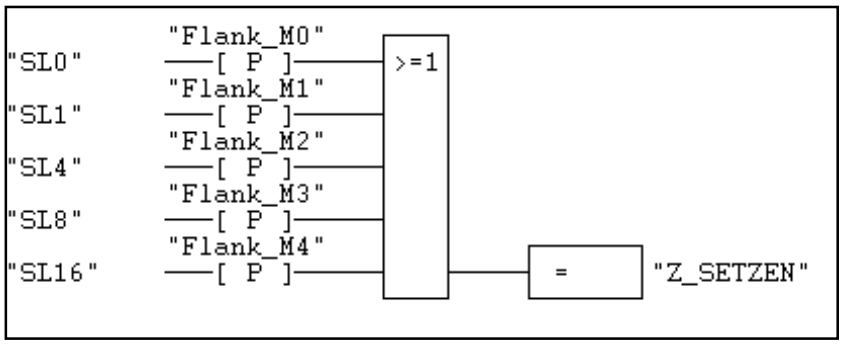
```
    L   "AUSWERT0"
    SLW 5
    BEA
_6:  L   "VerschDE"
    L   6
    ==I
    SPBN  _7
    L   "AUSWERT0"
    SLW 6
    BEA
_7:  L   "VerschDE"
    L   7
    ==I
    SPBN  _8
    L   "AUSWERT0"
    SLW 7
    BEA
_8:  L   "VerschDE"
    L   8
    ==I
    SPBN  _9
    L   "AUSWERT0"
    SLW 8
    BEA
_9:  L   "VerschDE"
    L   9
    ==I
    SPBN  _10
    L   "AUSWERT0"
    SLW 9
    BEA
_10: L   "VerschDE"
    L   C#10          // Achtung: Ab hier wird deutlich, dass der Vergleicher eine BCD-codierte Zahl
                      //erwartet! Vgl. zum Zahlenformat in der TrySim-Hilfe unter <Index > "Konstanten":
```

Zähler-Konstanten z.B. **C#59**. Hiermit laden Sie die angegebene Zahl **BCD -codiert**.
Der Wertebereich geht von C#0 bis C#999.

```
    ==I
    SPBN  _11
    L   "AUSWERT0"
    SLW 10
    BEA
_11: NOP 0
    L   "VerschDE"
    L   C#11
    ==I
    SPBN  _12
    L   "AUSWERT0"
    SLW 11
    BEA
_12: L   "VerschDE"
    L   C#12
    ==I
    SPBN  _13
    L   "AUSWERT0"
    SLW 12
    BEA
_13: L   "VerschDE"
    L   C#13
    ==I
    SPBN  _14
    L   "AUSWERT0"
    SLW 13
    BEA
_14: L   "VerschDE"
    L   C#14
    ==I
    SPBN  _15
```

```
    L   "AUSWERT0"
    SLW 14
    BEA
_15: L   "VerschDE"
    L   C#15
    ==I
    SPBN  _16
    L   "AUSWERT0"
    SLW 15
    BEA
_16: L   "VerschDE"
    L   C#16
    ==I
    SPBN  _x
    L   "AUSWERT0"
    SLD 16
    BEA
_x:  NOP 0
```

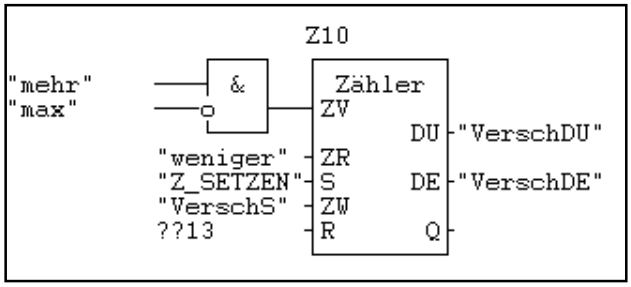
FC 2; Netzwerk 1: Flankenauswertung zum Setzen des Zählers, der die Verschiebung um X Stellen bestimmt.



Gesetzt wird der Zähler mit einer positiven Flanke. Damit beim Umschalten keine Überlappung der Schaltzustände entsteht und so dauernd eine "1" am ODER-Ausgang anliegen würde, sind Flankenmerker eingebaut worden. (Aktives Edit-Fenster | FUP-Elemente | <Steigende Flanke>). Dieser Verknüpfung muss jeweils ein sonst nicht benutzter Merker zugewiesen werden.

Vielleicht finden Sie ja eine andere Möglichkeit der Simulation.

FC 2; Netzwerk 2: Erzeugung der Schiebezahl



Das Wort <VerschDU> zeigt die Stellenzahl als Dualzahl an, um die der Ausgangswert nach links verschoben werden soll. Die Zahl wird als INT in der Digitalanzeige dargestellt.
<VerschS> ist die Vorgabe der Stellenzahl-Schalter. Mit <Z_SETZEN> wird <VerschS> an den Zähler übertragen. Die Stellenzahl kann mit <+1> über <mehr> um 1 erhöht werden, sofern nicht die Zahl 16 (<max>) erreicht wurde. Mit <-1> über <weniger> kann die Zahl um 1 reduziert werden.

Der Rücksetzeingang wird nicht benötigt. <VerschDE> ist der Ausgangswert im BCD-Format. Er wird zum Vergleich in der FC 1 benötigt.

FC 2; Netzwerk 3: Erkennen der max. Verschiebezahl 16

```
L   "VerschDE"
L   C#16
==I
=  M   "max"              // Das Bit wird im vorherigen Netzwerk benötigt.
```

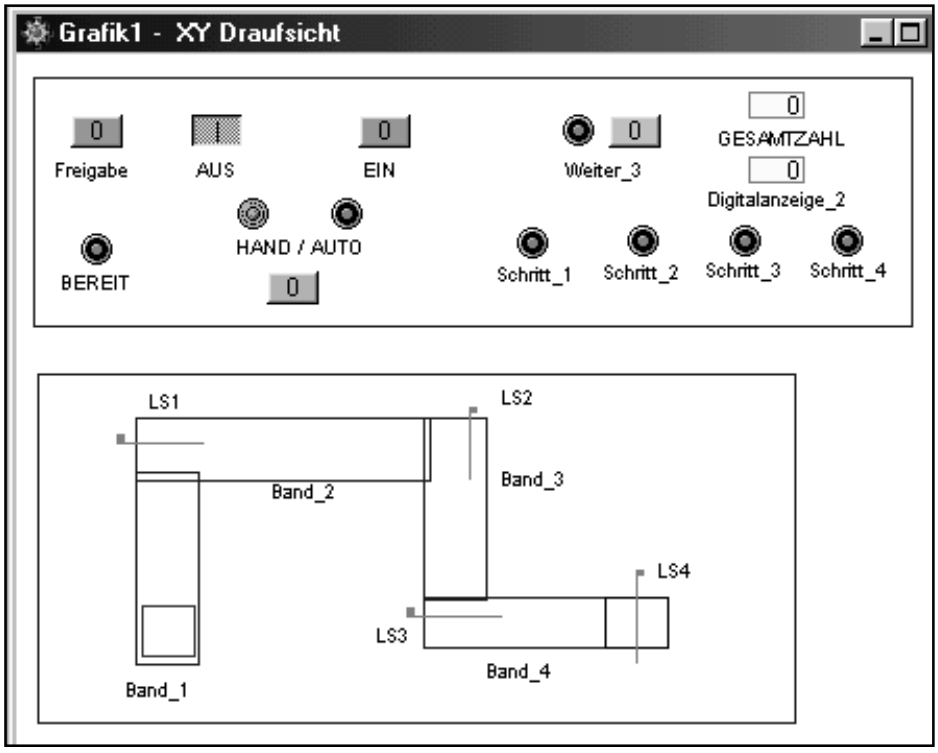
Projekt 19 : Zähleranwendung

Inhalte: Nachbildung des Prinzips der Ablaufsteuerung, getrennte speicherbare Ausgänge, S und R; Schalten mit Zählerbausteinen.

Aufgabe:
Sie werden in der Praxis wohl nicht nur neue Anlagen konzipieren, sondern vorhandene modifizieren. Das soll hier auch Grundlage der Übung sein.
Das bisherige Projekt <AS1_VAR1> soll etwas verbessert werden. Dazu sollten Sie sich eine Kopie Ihres entsprechenden Projektes anlegen und diese neu benennen.

- Bei Automatikbetrieb soll das 3. Band einschalten, ohne dass die zusätzliche Freigabe über <Weiter_3> erforderlich ist.
- Die Gesamtzahl der Pakete seit dem Einschalten mit <EIN> bis zum Ausschalten an <Freigabe> soll gezählt werden.
- Nach jeweils x Paketen soll die Anlage bei Automatikbetrieb von allein anhalten.

Im Pult sind 2 neue Digitalanzeigen "einzubauen".
Wählen Sie bitte bei aktivem Grafikfenster unter <Anlage>|<neues Element>|<Bedienung> die <Digitalanzeige> aus und ziehen sie mit gedrückter <LM> in das Pult im Grafikfenster. Tragen Sie dort nach <rM>-Klick den gewünschten Namen ein. Ein freier Operand wird in der Editiermaske vorgeschlagen. Sie können ihn übernehmen.



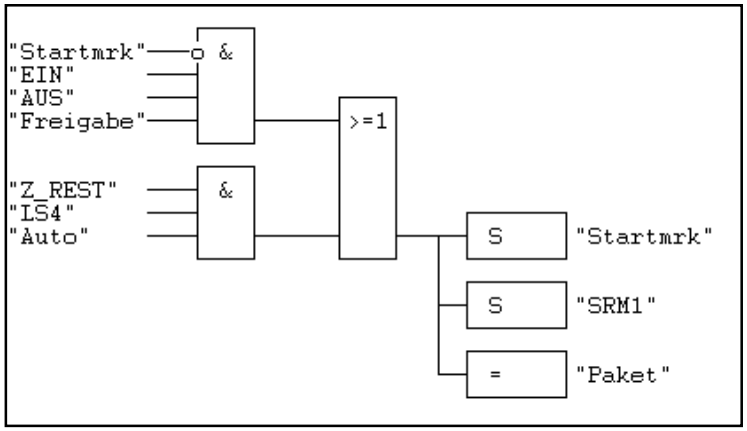
Damit sind die konstruktiven Voraussetzungen erfüllt.

Die Symboltabelle sollte noch um die neuen Adressen erweitert werden.

Symbol. Adresse	Operand	Datentyp	Kommentar
INIT	A 0.0	BOOL	Freigabe-Anzeige
LAusg_1	A 0.1	BOOL	LED Ausgang 1
LAusg_2	A 0.2	BOOL	LED Ausgang 2
LAusg_3	A 0.3	BOOL	LED Ausgang 3
LAusg_4	A 0.4	BOOL	LED Ausgang 4
LED_Hand	A 0.5	BOOL	LED Hand / 1 Durchgang
LED_Auto	A 0.6	BOOL	LED Automatik / Dauerbetrieb
Band_1	A 0.7	BOOL	Bandantrieb 1
Band_2	A 1.1	BOOL	Bandantrieb 2
Band_3	A 1.3	BOOL	Bandantrieb 3
Band_4	A 1.5	BOOL	Bandantrieb 4
Paket	A 1.7	BOOL	Generator / Paketspender
Freigabe	E 0.0	BOOL	Freigabeschalter
AUS	E 0.1	BOOL	AUS-Taster (Ö)
Weiter_3	E 0.3	BOOL	zus. Weiterschaltbedingung Band 3
EIN	E 0.6	BOOL	EIN-Taster (S)
Auto	E 0.7	BOOL	Wahlschalter HAND / AUTOMATIK
LS1	E 1.0	BOOL	Lichtschranke Ende Band 1
LS2	E 1.1	BOOL	Lichtschranke Ende Band 2
LS3	E 1.2	BOOL	Lichtschranke Ende Band 3
LS4	E 1.3	BOOL	Lichtschranke Ende Band 4
SRM1	M 0.1	BOOL	Schrittmerker 1
SRM2	M 0.2	BOOL	Schrittmerker 2
SRM3	M 0.3	BOOL	Schrittmerker 3
SRM4	M 0.4	BOOL	Schrittmerker 4
Startmrk	M 0.5	BOOL	Startmerker
GES_ZAHL	**AW 514**	**WORD**	**Gesamtzahl der Pakete**
REST	**AW 516**	**WORD**	**Restzahl der Pakete bis Stopp**
Z_GES	**Z1**	**COUNTER**	**Zähler Gesamtzahl**
Z_REST	**Z2**	**COUNTER**	**Zähler Restzahl**

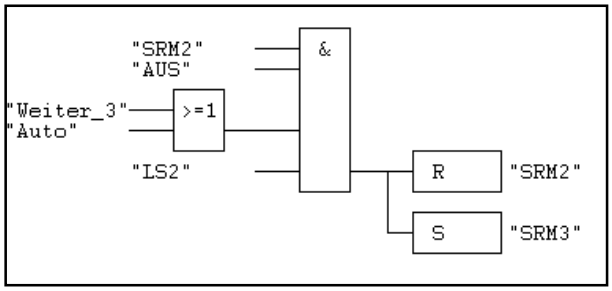
Nur die hier aufgeführten Netzwerke sind zu ändern.

Netzwerk 2: Schrittmerker 1 - Band 1
Nur beim ersten Takt, wenn der Startmerker nicht gesetzt ist, kann über "EIN" gestartet werden.



Wenn die gewählte Paketzahl noch nicht durchgelaufen ist, ist das VKE des Rückwärtszählers <Z_REST> "1".
So lange kann also bei Automatikbetrieb der Zyklus durch die Lichtschranke >LS4> neu begonnen werden.

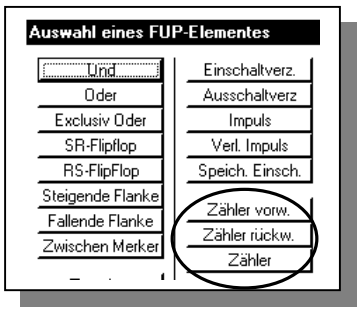
Netzwerk 4: Schrittmerker 3 - Band 3



Bei Handbetrieb kommt die zusätzliche Weiterschaltbedingung vom Taster <Weiter_3> und bei Automatikbetrieb steht die "1" vom Schalter <AUTO> dauernd an.
Wenn die anderen Bedingungen der UND-Verknüpfung gegeben sind, wird Band 3 über den <SRM2> eingeschaltet.

Neu ist das folgende Netzwerk. Dazu wird in dem letzten, leeren Netzwerk ein Zählerbaustein platziert.

Die erste Frage ist vielleicht, welche Zählertypen es gibt. Man findet die verschiedenen Zähler - wie schon bei den Timern besprochen - im Auswahlfenster (Icon Nr. 27 für FUP oder Nr. 33 für KOP).



Mit <LM> klicken Sie den Zählertyp <Zähler vorwärts> in das vorher ausgewählte Netzwerk und bezeichnen ihn im Kopf mit „Z1“. Bei der Wahl der symbolischen Darstellung wechselt die Bezeichnung in „Z_GES“.

Achten Sie bitte darauf, dass dabei der Cursor nicht im Kommentarfeld des Netzwerkes steht. Ein <LM>-Klick ins Editierfenster genügt.

Eine nähere Beschreibung der Zähler finden Sie im TrySim-Programm in der Menüzeile unter <Hilfe> <Index>|“Zähler“. Ausschnitte sind hier wiedergegeben:

Es gibt in der SPS 512 „Zähler“, die zum zählen von Ereignissen verwendet werden können. Die Zähler sind nummeriert von Z0 bis Z511, selbstverständlich kann ihnen über die Symboltabelle auch ein vernünftiger Name gegeben werden.
Zähler können als Aufwärtszähler, Abwärtszähler und als Auf- und Abwärtszähler verwendet werden. Der Zählbereich geht von 0 bis 999.
Wenn am Eingang ZV eine steigende Flanke auftritt, wird der Zähl-Wert um 1 erhöht. Wenn am Eingang ZR eine steigende Flanke auftritt, wird der Zähl-Wert um 1 erniedrigt. Wenn am Eingang S eine steigende Flanke auftritt, wird der Zähl-Wert auf den am Eingang ZW anliegenden Wert gesetzt. Wenn am Eingang R eine „1“ anliegt, wird der Zähler statisch auf 0 gesetzt, d.h. nicht nur bei einer Flanke. Am Ausgang DU wird der aktuelle Zähl-Wert dualcodiert in eine beliebige Word-Variable geschrieben. Am Ausgang DE wird der Zählwert BCD-codiert in eine beliebige Word-Variable geschrieben. In dieser Form kann er direkt wiederverwendet werden, um einem weiteren Zähler als ZW-Wert zu dienen.

Der Ausgang <Q> führt eine „1“, wenn der Zählwert > 0 ist. Das bedeutet, für Steuerfunktionen bei bestimmten Zählerständen kann nur der Rückwärtszähler verwendet werden, denn nur beim Zählerstand "0" führt der Ausgang <Q> ein anderes Signal als bei allen anderen Zählerständen.

Negative Zählwerte oder Werte >999 gibt es nicht. Falls beim Zählwert 0 eine Flanke am <ZR>-Eingang auftritt, geschieht nichts, genauso wenn bei Stand von 999 eine Flanke am <ZV>-Eingang auftritt.
<LM> auf den Link "Eingang ZW" auf dieser Seite führt zu:
Eingang <ZW> eines Zählers
Auf diesen Wert wird der Zähler gesetzt, wenn am S-Eingang eine steigende Flanke anliegt. Dieser Eingang muss nicht beschaltet werden, wenn Sie diese Funktion nicht benötigen. Der reine Abwärtszähler ist aber bei unbeschaltetem S- oder ZW-Eingang ziemlich nutzlos. Der Zähler kann nur auf Werte von 0 bis 999 gesetzt werden.
<LM> auf den Link "Ausgang DU" auf dieser Seite führt zu:
Ausgang <DU> eines Zählers
An diesem Ausgang wird der aktuelle Zählwert dualcodiert in eine beliebige Word-Variable geschrieben. Wenn Sie diesen Wert nicht benötigen, können Sie den Ausgang auch unbeschaltet lassen. Der Wertebereich der hier ausgegebenen Zahl geht von 0 bis 999. Sie können diesen Wert **nicht** als <ZW>-Wert eines anderen Zählers verwenden, da dieser BCD-codiert sein muss, verwenden Sie dazu den am <DE>-Ausgang ausgegebenen Wert.
<LM> auf den Link "Ausgang DE" auf dieser Seite führt zu:
Ausgang <DE> eines Zählers
An diesem Ausgang wird der aktuelle Zählwert BCD codiert in eine beliebige Word-Variable geschrieben. Wenn Sie diesen Wert nicht benötigen, können Sie den Ausgang auch unbeschaltet lassen. Der Wertebereich der hier ausgegebenen Zahl geht von 0 bis 999. Sie können diesen Wert als <ZW>-Wert eines anderen Zählers verwenden. diesen Wert jedoch **nicht** mit den +I, -I, *I und /I - Operationen weiterverarbeiten.

Wenn Sie mit dem Zählwert rechnen wollen, sollten Sie den am <DU> Ausgang ausgegebenen dualcodierten Wert verwenden. Wenn Sie den am <DE>-Ausgang ausgegebenen Wert mit anderen Zahlen vergleichen wollen, müssen Sie darauf achten, dass diese ebenfalls BCD-codiert sind.

Erläuterungen zum Vorwärtszähler oder zum Eingang <ZV> finden Sie unter
<Index>| „**ZV**“:
Bei einer steigenden Flanke des VKE’s bei der Bearbeitung dieses Befehls wird der Zählerwert um Eins erhöht. Wenn der Zählerwert bereits 999 beträgt, ändert er sich nicht.

Erläuterungen zum Rückwärtszähler oder zum Eingang <ZR> finden Sie unter
<Index>| „**ZR**“:
Bei einer steigenden Flanke des VKE’s bei der Bearbeitung dieses Befehls wird der Zählerwert um Eins erniedrigt. Wenn der Zählerwert bereits Null ist, ändert er sich nicht.

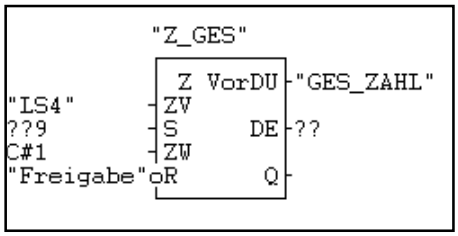
Unter **<Liste der Operationen>** sind alle verfügbaren Operationen aufgelistet, z.B. auch alle der Zähler.

Dort: **<Setze Zähler>** oder auch über <Index>| „**S**“:
Wenn der <S>-Eingang eines Zählers von „0“ nach „1“ geht, wird der Zähler auf den am <ZW>-Eingang angegeben Wert gesetzt. Die Zahl am <ZW>-Eingang muss BCD-codiert sein, das wird sie automatisch, wenn sie als C#.. Konstante angegeben wird, also für die Dezimalzahl 12 tragen Sie „C#12“ ein.
Zusätzlicher Hinweis: Wenn Sie variable Werte am <ZW>-Eingang übertragen wollen, z.B. mit einem MW(x), müssen Sie auch darauf achten, dass ein BCD-Code vorliegt. Falls im Wort ein Dualcode vorliegt, müssen Sie ihn mit "ITB" wandeln:

```
L MW 5      // Lade das Dualwort MW 5
ITB         // transformiere von INT TO BCD
T MW 7      // das soll das Eingangswort an ZW sein.
```

Dort: **<Rücksetze Zähler>** oder auch über <Index>| „**R**“:
Der angegebene Zähler wird auf den Wert 0 gesetzt. Abfragen des Boolschen Zustandes des Zählers am Ausgang <Q> ergeben so lange „falsch“ = "0", bis der Zähler wieder hochgezählt wird.

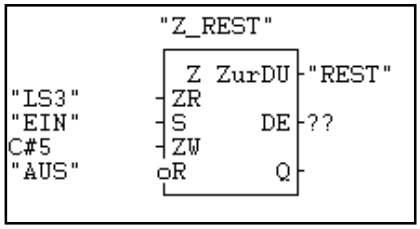
Netzwerk 11: Zählen der Gesamtpakete



Mit den Impulsen der letzten Lichtschranke <LS4> werden die Pakete gezählt. Wenn <Freigabe> zurückgesetzt wird, springt der Zähler auf den Wert 0.
Das Ausgangswort "GES_ZAHL" wird als Int-Wert an der Digitalanzeige dargestellt. Klicken Sie dazu bei ausgeschalteter Anlage im Pult mit <LM> auf die Anzeige und anschließend an den Ausgang <DU>.

Verfahren Sie beim nächsten Zähler in ähnlicher Weise. Wählen Sie dafür jedoch den Rückwärtszähler aus.

Netzwerk 12: Zahl der restlichen Pakete bis zum Abschalten.



Beim Einschalten wird der Rückwärtszähler über <S> auf den vorgewählten ZW-Wert gesetzt. Die dritte Lichtschranke <LS3> liefert die Impulse, mit denen <Z_REST> rückwärts zählt. Der Ausgang Q muss nicht mit einem Zuweisungsblock verbunden werden. Dieser Bitwert kann auch symbolisch mit "Z_REST" oder direkt mit "Z2" in anderen Netzwerken verarbeitet werden.

Warum wurde hier die Lichtschranke <LS3> verwendet? Wenn der Zählerstand "0" ist, liegt an Ausgang Q eine "0". Bei allen anderen Zählerständen läge eine "1" an. Würde <LS4> verwendet, würde <Q> zwar auch auf "0" gesetzt, aber gleichzeitig könnte der Zyklus noch einmal neu beginnen, weil im Netzwerk 2 <Z_REST> noch für einen Zyklus auf "1" liegen würde, wenn dort <LS4> ein erneutes Startsignal liefern würde.

Bei den restlichen Netzwerken treten keine Änderungen auf.

Vertiefung:
Löschen Sie bitte im Netzwerk 11 an <DU> das Ausgangswort und setzen es im selben Zählerbaustein an den <DE>-Ausgang.
Starten Sie die Anlage über <Freigabe> neu und verfolgen Sie die angezeigten Zahlen der <Gesamtzahl>. Betätigen Sie <EIN> so oft, bis die Anzeige der <Gesamtzahl> von 9 auf den nächsten Wert wechselt. Sie vermuten sicherlich, dass die „10“ angezeigt wird. Vielleicht sind Sie etwas verdutzt, wenn Sie statt dessen „16“ lesen. Falls Sie nicht verdutzt sind, haben Sie anscheinend beim Kapitel BCD-Zahlen gut aufgepasst, denn am <DE>-Ausgang wird intern der Wert BCD-codiert, aber wegen der Bequemlichkeit des Betrachters als Dezimalzahl dargestellt. Da die Anzeige für die INT-Darstellung programmiert wurde, kann das „falsche“ Bitmuster nur entsprechend falsch dargestellt werden.

Wichtig ist, an welchem Ausgang (<DU> oder >DE>) der Code abgegriffen wird.

Beachten Sie bei der Weiterverarbeitung die Codierung des Ausgangswortes

Damit wieder alles reibungslos funktioniert, sollten Sie die Änderung an dem Zählerausgang wieder rückgängig machen.

vgl. TrySim-Projekt <AS_ZAHL>

Projekt 20: Richtungsabhängiges Zählen 1

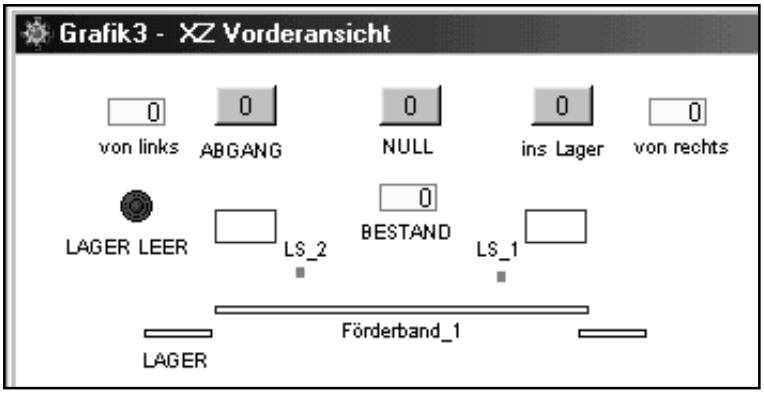
Schwerpunkte: Zähleranwendung

Aufgabe: Richtungsabhängiges Zählen
An einer einspurigen Wegstrecke sollen richtungsabhängig die bewegten Pakete gezählt werden. Außerdem soll der aktuelle Bestand im Lager (links) erfasst werden.
Als Eingabesignale sind hierfür 2 Lichtschranken erforderlich, die richtungsabhängig zählen.

Die Freigabe der Pakte wird in beiden Richtungen über die entsprechenden Taster ausgelöst. Es darf auf dem Band nicht zu Kollisionen kommen. Jede Vorwahl muss automatisch zu Ende geführt werden. Wenn das Lager leer ist, ist eine weitere Entnahme unmöglich, es leuchtet zusätzlich eine Kontroll-LED.
Die durchgelaufenen Pakete werden optisch nicht gelagert, sondern mittels bereits projektierter Vernichter "entsorgt". Die Entnahme aus dem Lager wird über den linken Generator simuliert.

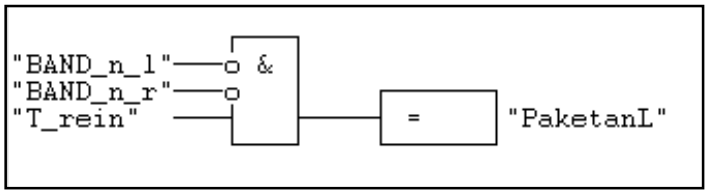
Durchführung des Projektes mit TrySim:
Im Verzeichnis <Vorlagen> ist das Projekt unter <LI_RE_ZAHL_1_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Anlage und die Symboltabelle sind bereits vorbereitet.



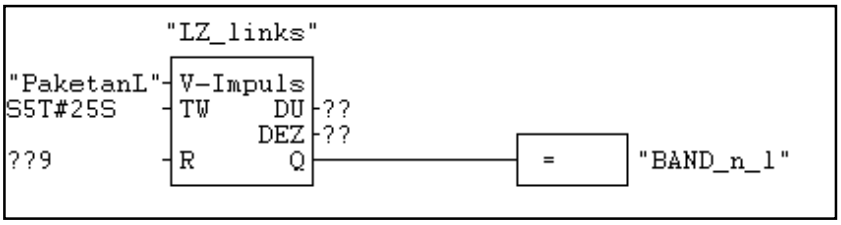
Symbolische Adresse	Operand	Datentyp	Kommentar
BAND_n_l	A 0.0	BOOL	Band nach links (Lager)
BAND_n_r	A 0.1	BOOL	Band nach rechts (Abgang)
PaketanL	A 0.2	BOOL	Paket ins Lager
Paket_vL	A 0.3	BOOL	Paket von Lager (Abgang)
T_rein	E 0.0	BOOL	Taster Paket ins Lager (S)
T_raus	E 0.1	BOOL	Taster Paket aus Lager (S)
T_NULL	E 0.2	BOOL	Zähler auf Null setzen
LS_re	E 0.3	BOOL	Lichtschranke 1 rechts
LS_li	E 0.4	BOOL	Lichtschranke 2links
EINLAG	Z 0	COUNTER	Zähler Einlagerung
AUSLAG	Z 1	COUNTER	Zähler Auslagerung
BESTAND	Z 2	COUNTER	Zähler Lagerbestand
LZ_links	T 0	TIMER	Laufzeit Band nach links
LZ_re	T 1	TIMER	Laufzeit Band nach rechts
M_LS_re	M 1.0	BOOL	Merker 1. Imp. an LS_re
M_LS_li	M 1.1	BOOL	Merker 1. Imp. an LS_li
P_rein	M 1.2	BOOL	Zählimp. Pakete ins Lager
P_raus	M 1.3	BOOL	Zählimp. Pakete raus aus Lager
L_LEER	A 1.0	BOOL	Lager ist leer

OB 1; Netzwerk 1: Paket nach links



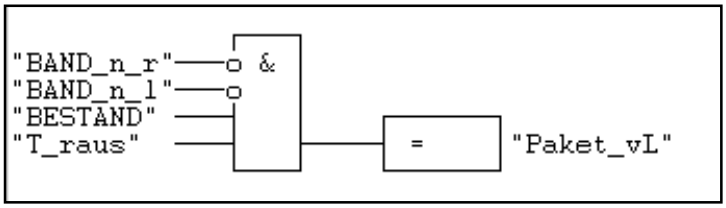
Erst wenn das Band steht –beide Richtungen sind aus – kann mit dem Taster ein neues Paket angefordert werden (siehe nächstes Netzwerk).

OB 1; Netzwerk 2: Antrieb nach links



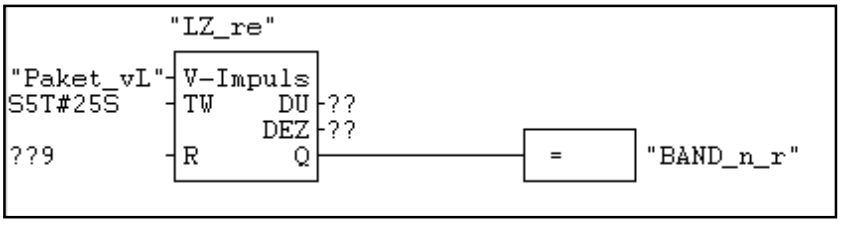
Aus der Tasteranforderung wird ein Impuls für die Dauer eines Transportweges.

OB 1; Netzwerk 3: Paket nach rechts

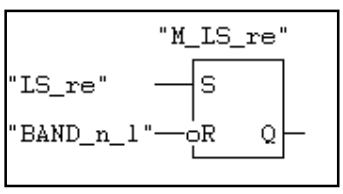


Nur wenn der Bestandszähler einen Vorrat im Lager anzeigt, kann – wie in der anderen Richtung – ein Paket abgerufen werden.

OB 1; Netzwerk 4: Antrieb nach rechts

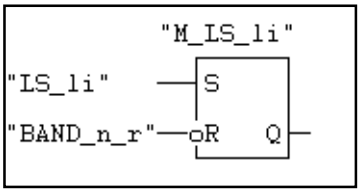


OB 1; Netzwerk 5: Speichern Lichtschranke 1 (rechts)

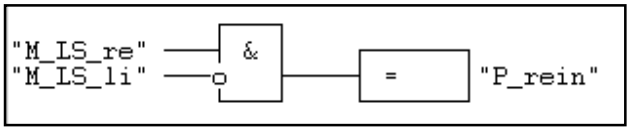


Durch diesen Merker wird aus dem Impuls der rechten Lichtschranken eine Dauer-„1“. Dieser Merker ermöglicht das richtungsabhängige Zählen (s.u.).
Der Merker verliert seine Information, wenn das Band (nach links) ausgeschaltet wird.

OB 1; Netzwerk 6: Speichern Linkschranke 2 (links)

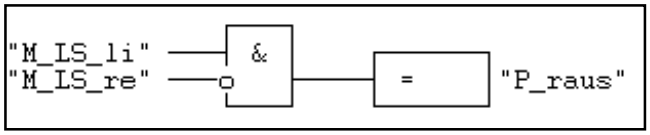


OB 1; Netzwerk 7: Zählen von rechts



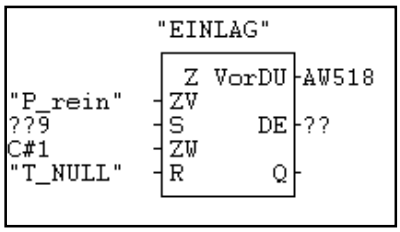
Wenn der Merker der rechten Lichtschranke zuerst gesetzt wurde, wird im NW 9mit <P_rein> der Zähler für die zugehenden Pakete raufgezählt. Wenn die zweite – linke– Lichtschranke auf „1“ gesetzt wird, hat die Kombination keine Zählwirkung.

OB 1; Netzwerk 8: Zählen von links



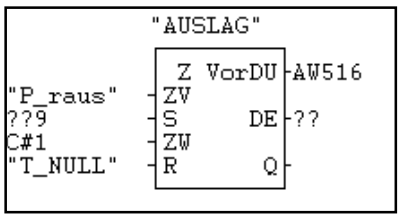
Wenn der Merker der linken Lichtschranke zuerst gesetzt wurde, wird im NW 10 mit <P_raus> der Zähler für die abgehenden Pakete raufgezählt. Wenn die zweite – rechte – Lichtschranke auf „1“ gesetzt wird, hat die Kombination keine Zählwirkung.

OB 1; Netzwerk 9: Zähler von rechts



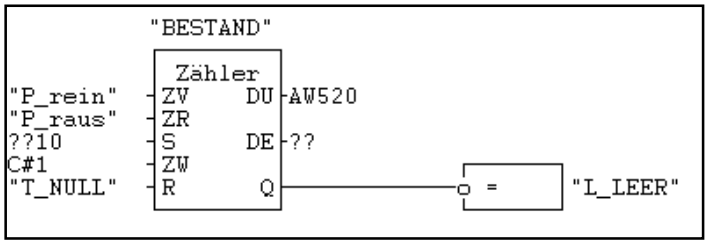
Hier werden die einzulagernden Pakete summarisch gezählt.
Mit <NULL> werden alle Zähler zurückgesetzt.
Die Anzeige erfolgt über die Digitalanzeige.

OB 1; Netzwerk 10: Zählen von links



Hier werden die auszulagernden Pakete summarisch gezählt.
Mit <NULL> werden alle Zähler zurückgesetzt.
Die Anzeige erfolgt über die Digitalanzeige.

OB 1; Netzwerk 11: Bestandszähler



Der Vor-/Rückwärtszähler erfasst den aktuellen Lagerbestand.
Der Leerstand wird an der LED angezeigt und verhindert die weitere Entnahme und den Bandstart für den Lauf nach rechts.

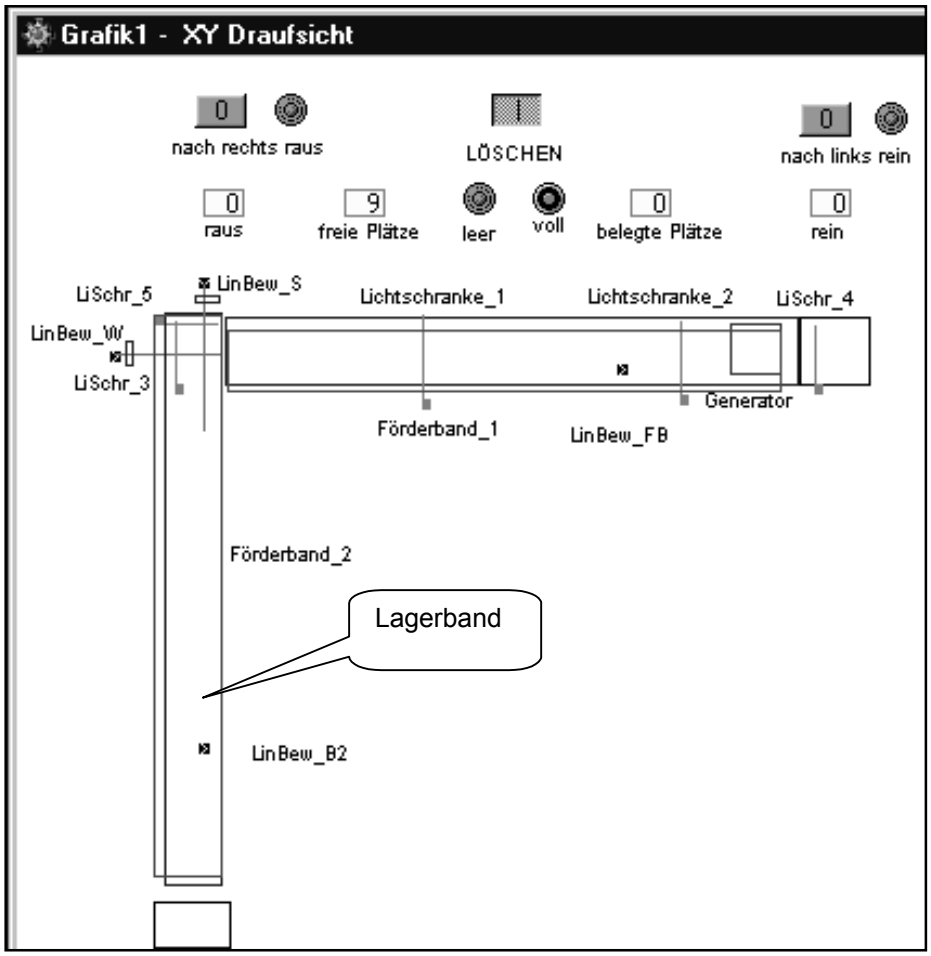
Projekt 21: Richtungsabhängiges Zählen 2

Schwerpunkte: Zähleranwendungen

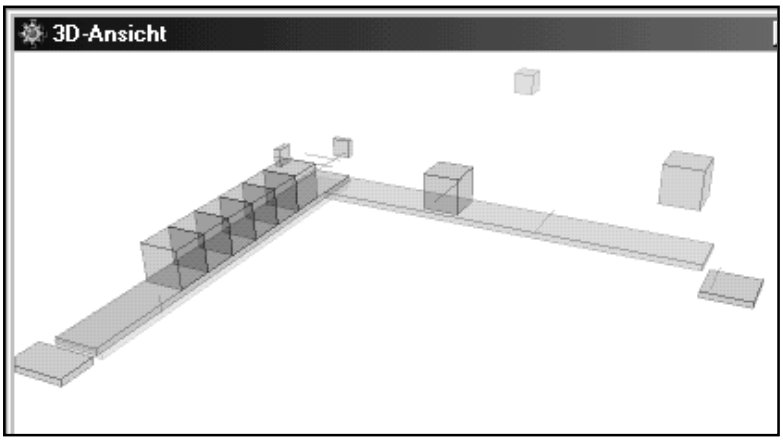
Aufgabe: Lager- / Produktionszähler
Dieses ist die Fortsetzung der Zählaufgabe <LI_RE_ZAHL_1> in einem Lager von begrenzter Größe, dessen Kapazität (9 Teile) realistisch nachgebaut wurde.
Bei Tasterbetätigung wird einem Magazin (Generator) ein Paket entnommen, welches über ein Förderband (Zählstrecke) auf ein zweites Förderband (Lager) transportiert wird. Die Bänder stehen rechtwinklig zu einander. Die gesamte, vom Magazin geholte Stückzahl wird gezählt. Lichtschranken geben die Zähl- und Weiterschaltimpulse. Aus dem Lager können Teile abgerufen werden, die dann mit
Band 1 nach rechts bis zur Lichtschranke 4 transportiert werden. Der Füllstand des Lagers und der Gesamtabgang werden auch gezählt. Beim Einlagern werden die Pakete auf das Band 2 geschoben; Band 2 läuft nicht. Beim Auslagern wird Band 2 bewegt, bis die Verschiebeposition auf Band 1 erreicht ist.
Natürlich darf es durch die verschiedenen Kommandos und Bahnbewegungen nicht zu Kollisionen kommen. Bei vollem Lager kann nur noch ausgelagert werden.
Beim Druck auf <Löschen> werden die Zähler zurückgesetzt und das Lager wird "geräumt", indem mit 2 bereits installierten Linearbewegern senkrecht bewegte Vernichter auf Bandhöhe angehoben werden.

Durchführung des Projektes mit TrySim:
Im Verzeichnis <Vorlagen> ist das Projekt unter <LI_RE_Zahl_2_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Anlage und die Symboltabelle sind bereits vorbereitet.



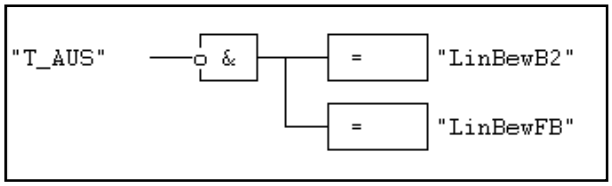
Anlage:



Symboltabelle:

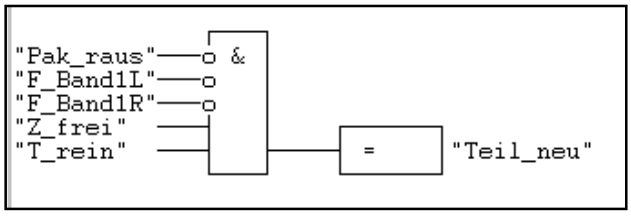
Symbolische Adresse	Operand	Datentyp	Kommentar
F_Band1R	A 0.4	BOOL	F_Band waagerecht, Antrieb n.rechts
F_Band1L	A 0.5	BOOL	F_Band waagerecht, Antrieb n. links
LinBew_S	A 0.6	BOOL	Linearbeweger senkrecht, Ventil
Teil_neu	A 0.7	BOOL	Generator für neue Teile
MAG_voll	A 1.0	BOOL	Magazin ist voll (Anzeige)
MAG_leer	A 1.1	BOOL	Magazin ist leer (Anzeige)
F_Band2o	A 1.3	BOOL	F_Band senkrecht, Antrieb nach oben
LinBewWR	A 1.4	BOOL	Linearbeweger waager.; nach rechts
LinBewWL	A 1.5	BOOL	Linearbeweger waager.; nach links
LinBewB2	A 2.0	BOOL	Linearbeweger zum Löschen der Teile
AUSLAG_M	A 2.1	BOOL	LED Auslagerung möglich
LinBewFB	A 2.3	BOOL	Linearbew. Löschen d. Pakete auf FB
EINLAG_M	A 2.4	BOOL	LED Einlagerung möglich
ANZ_rein	AW 516	WORD	Anzahl / Anzeige ALLE eingelagert
ANZ_raus	AW 518	WORD	Anzahl / Anzeige Teile ausgeliefert
ANZ_blgt	AW 520	WORD	Anzahl / Anzeige Teile im Magazin
ANZ_frei	AW 522	WORD	Anzahl / Anzeige Plätze frei im Magazin
T_rein	E 0.0	BOOL	Taster "Einlagern"(S)
T_raus	E 0.1	BOOL	Taster "Auslagern" (S)
LiSchr_1	E 0.2	BOOL	Lichtschranke 1 (links) (S)
LiSchr_2	E 0.3	BOOL	Lichtschranke 2 (rechts) (S)
LiSchr_3	E 0.4	BOOL	Lichtschranke 3 (Stop links) (S)
LiSchr_4	E 0.5	BOOL	Lichtschranke 4 (Ausgang rechts) (S)
T_AUS	E 0.6	BOOL	Taster Aus (Ö)
LiSchr_5	E 0.7	BOOL	Lichtschranke 5 (F_Band_2 oben) (S)
ETlinks	E 1.0	BOOL	Endtaster Lin.Beweger waager. links (S)
ETrechts	E 1.1	BOOL	Endtaster Lin.Beweg. waager. rechts (S)
LvorR	M 0.0	BOOL	Zählt wenn LiSchr_1 vor LiSchr_2 "1" ist
RvorL	M 0.1	BOOL	Zählt wenn LiSchr_2 vor LiSchr_1 "1" ist
Pak_raus	M 50.0	BOOL	Merker 1 Paket aus Lager holen
Z_raus	Z 0	COUNTER	Anzahl Teile raus
Z_rein	Z 1	COUNTER	Anzahl Teile rein
Z_belegt	Z 2	COUNTER	Anzahl im Lager
Z_frei	Z 3	COUNTER	Anzahl frei im Lager

OB 1; Netzwerk 1: Bänder durch Vernichter leeren



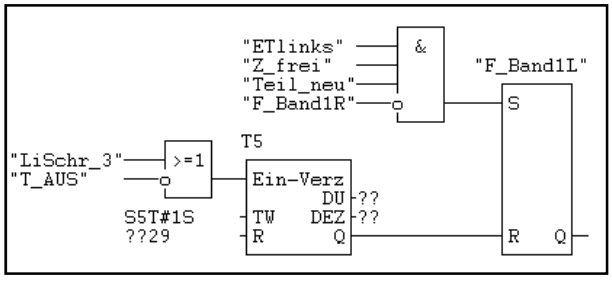
Dieses Netzwerk ist bereits vorbereitet. Es hat mit der eigentlichen Aufgabe nichts zu tun, sondern dient lediglich dazu, die Pakete „abzuräumen“, indem durch die Linearbeweger Vernichter senkrecht bewegt werden.

OB 1; Netzwerk 2: neue Dynamiks nach links



Wenn kein Speicherbefehl anliegt, dass ein Paket ausgelagert werden soll, Band 1 nicht läuft, im Lager noch Platz ist, dann kann mit Tasterdruck auf <nach links rein> ein neues Paket vom Generator angefordert werden.

OB 1; Netzwerk 3: Band nach links (rein)

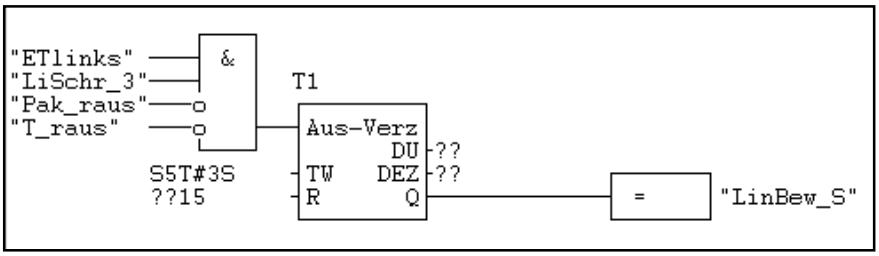


Band 1 fährt nach links (Paket wird eingelagert), wenn

- <LinBew_W> links steht („ETlinks“ = 1)
- im Lager noch Platz ist
- der Generator ein neues Paket liefert und
- Band 1 nicht läuft

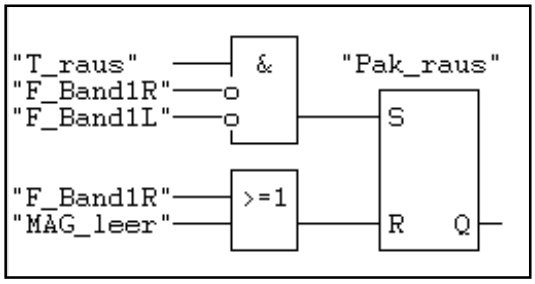
Gestoppt wird Band 1 bei Linkslauf, wenn das Paket die Lichtschranke 3 erreicht hat, oder auf <LÖSCHEN> gedrückt wird. Beim Erreichen der Lichtschranke darf das Band nicht sofort stehen, weil das Paket noch etwas über die Lichtschrankenposition hinaus transportiert werden soll. Deshalb ist ein Verzögerungsglied dazwischen geschaltet. (Natürlich könnte „T_AUS“ auch direkt auf <R> wirken).

OB 1; Netzwerk 4: Linearbeweger senkrecht (Impuls)



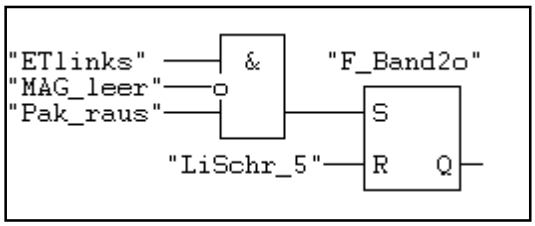
Wenn der waagerechte Schieber in Grundposition (links) steht, die Lichtschranke 3 durch ein Paket betätigt ist und kein Paket ausgelagert werden soll ist das VKE des UND-Gliedes „1“. Dadurch wird das Paket senkrecht auf das Band 2 verschoben. Dabei wird „LiSchr_3“ wieder „0“. Das Paket soll aber noch ein bisschen weiter geschoben werden. Deshalb wird die Spannung am Ventil „LinBew_S“ erst verzögert abgeschaltet. (Durch eine Feder wird das spannungslose Ventil umgeschaltet).

OB 1; Netzwerk 5: Merker; 1 Paket aus Lager holen



Mit dem Taster-Impuls <nach rechts raus> wird der Merker "Pak_raus" gesetzt, wenn bei dem Tasterbefehl das Band 1 steht.
Wenn dann etwas später das Band 1 ein Paket nach rechts transportiert, wird dieser Merker nicht mehr als Information benötigt und zurückgesetzt. Wenn allerdings das Lager leer ist, wird der Setz-Befehl durch den dominanten Rücksetzbefehl von „MAG_leer" ignoriert.

OB 1; Netzwerk 6: Förderband _2

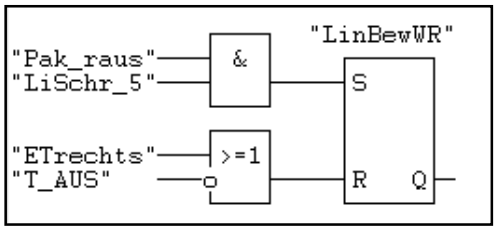


Das Band 2 fährt nach oben, um ein Paket vor Band 1 zu schieben, wenn

- der waagerechte Schieber nicht aktiv ist
- das Lager nicht leer ist und
- der Befehl zum Auslagern gespeichert ist.

Durch die Lichtschranke 5 wird der Antrieb ausgeschaltet.

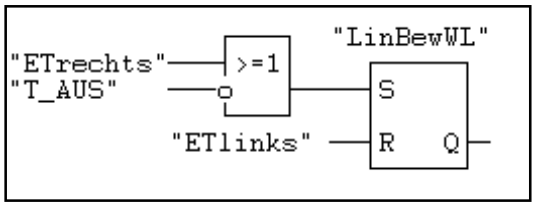
OB 1; Netzwerk 7: Linearbeweger waagerecht nach rechts



Wenn die Lichtschranke 5 von einem hochgeschobenen Paket eingeschaltet wird, schiebt der waagerechte Schieber das Paket auf das Band 1, sofern vorher der Befehl zum Auslagern gespeichert wurde
Zurückgesetzt wird der Baustein (nicht der Schieber), wenn dessen rechter Endtaster erreicht wurde, oder auf <LÖSCHEN> gedrückt wird.

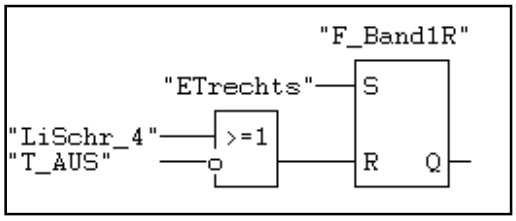
Der Rückweg für diesen Linearbeweger wäre als 4/2-Wege-Ventil (Feder) einfacher zu programmieren. Als Abwechselung zum Schieber in NW 4 wurde hier aber eine Steuerung mit 2 Richtungsspulen gewählt. Deshalb folgt das...

OB 1; Netzwerk 8: Linearbeweger waagerecht nach links zurück



Wenn der rechte Endtaster erreicht ist, oder wenn <LÖSCHEN> betätigt wird, wird die Spule für den Rückweg des Schiebers aktiviert.
In der Grundstellung schaltet der Endtaster ab.

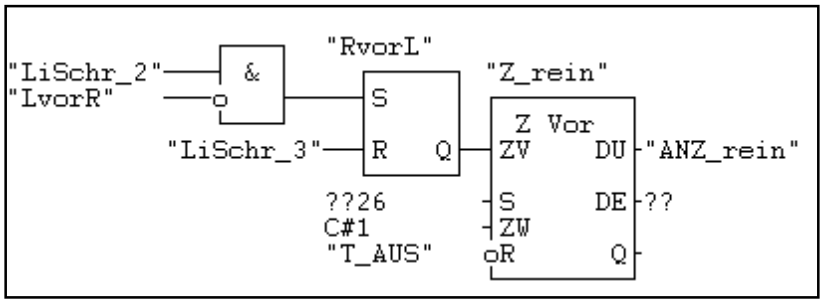
OB 1; Netzwerk 9: Band nach rechts (raus)



Wenn ein Paket durch den waagerechten Schieber auf das Band 1 geschoben hat, ist der rechte Endtaster des Schiebers betätigt.. Dadurch wird das Band 1 für Rechtslauf eingeschaltet.
Wenn das Paket rechts die Lichtschranke 4 erreicht, oder wenn <LÖSCHEN> betätigt wird, schaltet das Band 1 ab.

Der technische Ablauf ist hiermit programmiert. Jetzt noch das Zählprogramm.

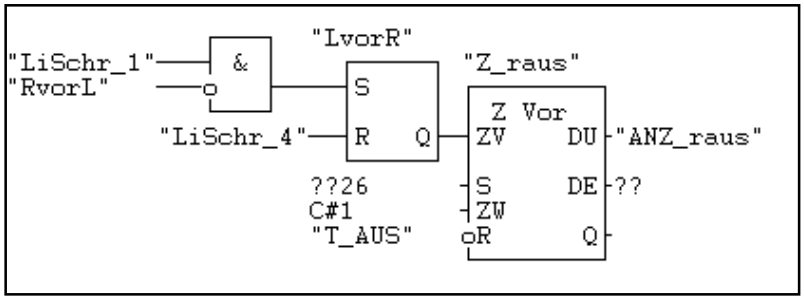
OB 1; Netzwerk 10: Anzahl Teile rein (mit Richtungserkennung)



Wenn die Lichtschranke 2 eine „1“ liefert, und der noch nicht gesetzte Merker „LvorR“ eine „0“, wird der Merker „RvorL“ gesetzt und gleichzeitig der Zähler für die einzulagernden Pakete um 1 raufgesetzt. Der Zählerstand wird als WORT an die Anzeige „ANZ_rein“ gelegt.

Wenn das Paket die Lichtschranke 3 erreicht, wird der Richtungsmerker zurückgesetzt. Der Zählerbaustein wird mit <LÖSCHEN> auf 0 zurückgesetzt.

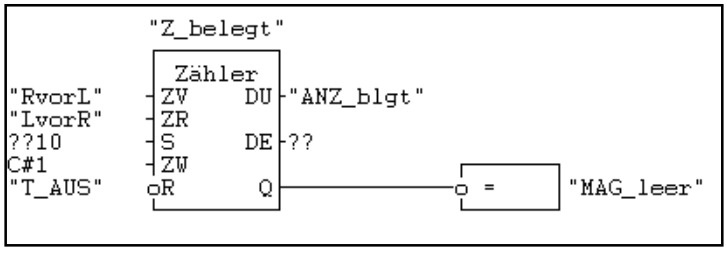
OB 1; Netzwerk 11: Anzahl Teile raus (mit Richtungserkennung)



Wenn die Lichtschranke 1 eine „1“ liefert, und der noch nicht gesetzte Merker „RvorL“ eine „0“, wird der Merker „LvorR“ gesetzt und gleichzeitig der Zähler für die auszulagernden Pakete um 1 raufgesetzt. Es wird die Gesamtzahl aller aus dem Lager geholten Pakete gezählt.

Wenn das Paket die Lichtschranke 4erreicht, wird der Richtungsmerker zurückgesetzt. Der Zählerbaustein wird mit <LÖSCHEN> auf 0 zurückgesetzt.

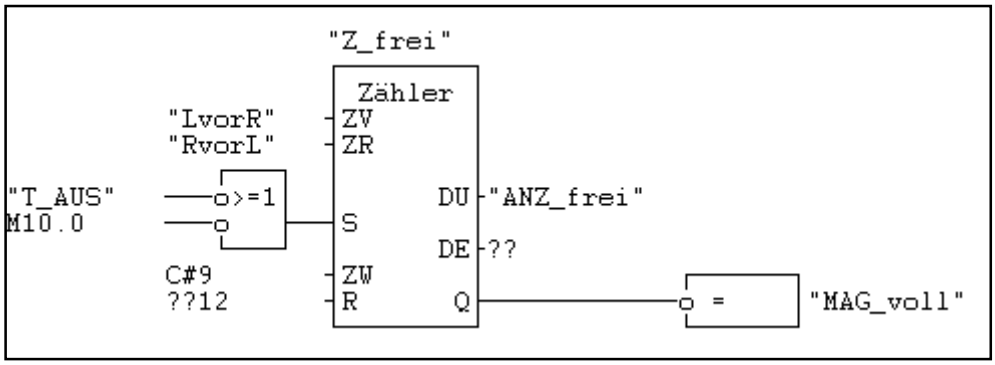
OB 1; Netzwerk 12: belegte Plätze



Dieser Vorwärts- / Rückwärtszähler zählt die Anzahl der belegten Plätze. Bei Zählerstand 0 zeigt „MAG_leer“ diesen Zustand an.
Der Zählerstand wird als WORT an die Anzeige „ANZ_blgt“ gelegt.

Sowie der Zähler die Zahl >0 anzeigt, springt der Ausgang <Q> auf „1“. „MAG_leer“ führt dann immer „0“.

OB 1; Netzwerk 13: Voll-Anzeige / freie Plätze



Für die Anzeige er freien Plätze kann man von der Gleichung ausgehen:

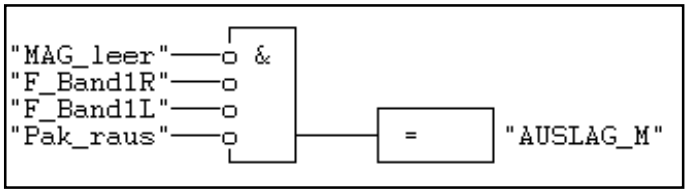
Freie Plätze = max. Plätze – belegte Plätze.

Dafür muss der Zähler beim Starten des Programms (beim Einschalten) ohne weitere Einwirkung auf dem max. Wert gesetzt werden. Das soll für diese Anlage „9“ sein. Gleichzusetzen mit dem Start des Programms ist das Beenden durch <LÖSCHEN>. Dieser Befehl ist nicht gleichbedeutend mit dem Einschalten! Deshalb ist für beide Fälle eine Setzbedingung anzugeben.
Im (folgenden!) Netzwerk sehen Sie, wie die Einschaltsituation über „M10.0“ zum einmaligen Setzen des Zählers führt.
Die einzulagernden Pakete zählen den Zähler zurück. Die auszulagernden Pakete erhöht die Zahl der freien Plätze.

OB 1; Netzwerk 14: Setzimpuls "Zähler freie Plätze"

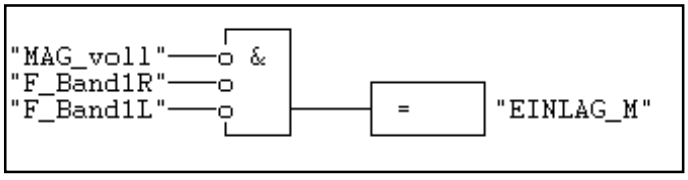
```
UN M   10.0      //Beim ersten Einschalten ist dieser Merker durch nichts gesetzt.
S  M   10.0      //unter der vorstehenden Bedingung wird der Merker gesetzt und bleibt es.
```

OB 1; Netzwerk 15: Bedienung (auslagern)



Wenn das Lager nicht leer ist, das Band 1 steht und das Auslagern nicht bereits vorgewählt wurde, zeigt die LED, dass ein Auslagern gewählt werden kann.

OB 1; Netzwerk 16: Bedienung (einlagern)



Wenn das Lager nicht voll ist und das Band 1 steht, zeigt die LED, dass ein Einlagern gewählt werden kann.

Vgl. TrySim-Projekt <LI_RE_Zahl_2> im Verzeichnis <Lösungen>.

Projekt: 22: Sollwerteinsteller

Schwerpunkte: Integerwerte, +I, -I, ==I, (Zählen ohne Zähler), Vergleichen, Sprungbefehle, FC

Aufgabe: Durch Tasterdruck ist ein Zahlenwert in den Grenzen von -10 bis +10 zu verändern. Dabei können 2 Werte (-5 und +6) direkt vorgewählt werden und der ganze Bereich ist durch Dauerdruck auf die Taster <mehr> bzw. <weniger> kontinuierlich zu verändern.

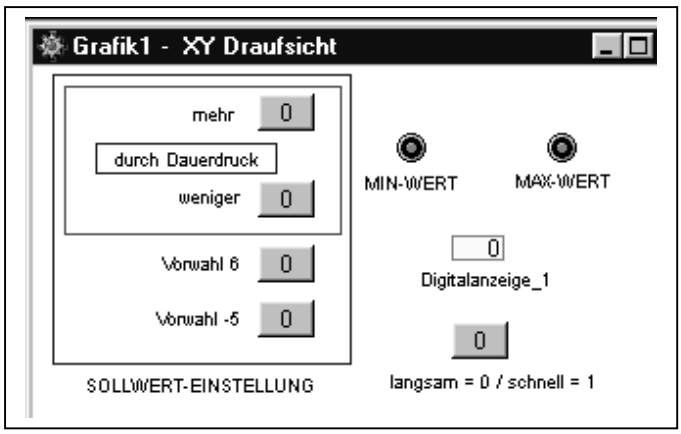
Theorie: Dieses Projekt allein macht keinen Sinn und ist eher als Teil einer Anlage (Unterprogramm) zu verstehen. Es bietet - isoliert betrachtet - aber die Möglichkeit, neue Operationen als Softwareanwendung zu testen. Auch dafür bietet TrySim hervorragende Möglichkeiten

Zahlenwerte lassen sich gut mit Zählerbausteinen verändern, aber nur im positiven Bereich. Negative Zahlen können Sie damit nicht darstellen. Es geht aber auch anders:
Für diese Aufgabe werden wir Integerbausteine verwenden.
Im Vergleich zu den Zählerbausteinen zeigt sich hier jedoch ein wesentlicher Unterschied in der "Gewinnung" der Zahlen:
Die Eingänge der Zähler sind flankengesteuert, d.h., es werden nur Impulse (mit einem Wechsel von "0" auf "1") verarbeitet. Eine Dauer-"1" führt auch nur zu einer Änderung des Wertes um 1.
Integerbausteine führen ihren Auftrag in jedem Zyklus erneut durch. Sie führen die Anweisung ohne Vorbedingung durch, d.h. Verknüpfungen z.B. mit U .... gibt es nicht. Auch Befehle wie "L ..." (Lade) oder "T ..." (Transferiere, übertrage) werden ohne Vorbedingungen ausgeführt.
Wenn man nun will, dass nur unter bestimmten Bedingungen gezählt werden soll / darf, muss man mit Sprungadressen arbeiten. Die hier verwendeten Sprungadressen erlauben das Abarbeiten der Programmzeilen in beliebig (geplanter!) Weise; nicht wie üblich, von oben nach unten, sondern direkt zum Sprungziel, der **Sprungmarke.**
Die Auswahl der möglichen Sprünge finden Sie unter <Hilfe>, >Index>, "SP".

Sprungmarken müssen im selben Baustein liegen. Sie dürfen max. 4 Zeichen lang sein und nur aus Buchstaben, Zahlen und dem Unterstrich bestehen. Das Programm erkennt eine Sprungmarke an dem Doppelpunkt, gefolgt von der ersten Anweisung.

Wie Sie wissen, darf im FUP oder KOP - abgesehen von Parallelschaltungen - immer nur 1 Ausgang je Netzwerk zugewiesen werden. In AWL können mehrere dieser "Kleinanweisungen" hintereinanderweg in einem Netzwerk angelegt werden. Wegen der Sprünge kommt für diese Aufgabe auch nur AWL in Frage.

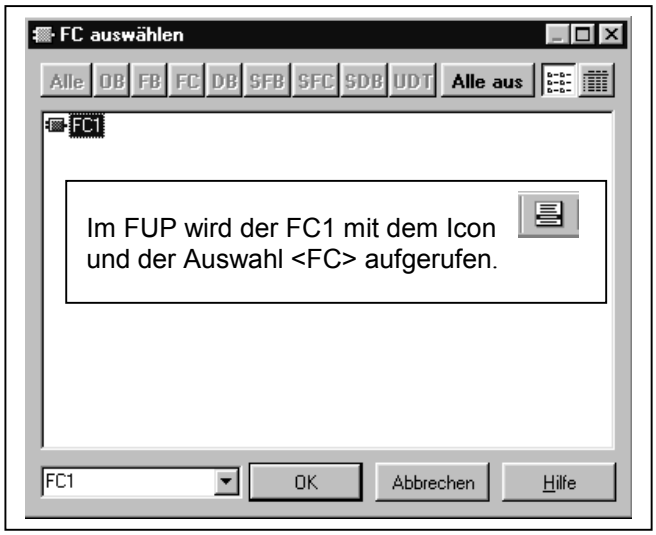
Zur Praxis mit TrySim
Wählen Sie bitte aus dem Ordner **<Vorlagen>** das Projekt **<Sollwert_V>**, speichern Sie es als neues Projekt in Ihrem eigenen Verzeichnis und öffnen das Rumpfprojekt. Die Anlage und die Symboltabelle sind bereits angelegt.



Symbolische Adresse	Operand	Datentyp	Kommentar
Anzeige	AW 512	WORD	Sollwertanzeige
L_MIN_W	A 0.0	BOOL	Leuchter Minimalwert (-10) erreicht
L_MAX_W	A 0.1	BOOL	Leuchter Maximalwert (+10) erreicht
T_(-5)	E 0.3	BOOL	Taster Vorwahl (- 5)
T_(6)	E 0.2	BOOL	Taster Vorwahl (+ 6)
T_mehr	E 0.0	BOOL	Taster höherer Wert
Tweniger	E 0.1	BOOL	Taster Wert erniedrigen

Auch wenn es für die Logik des Programms nicht erforderlich ist, werden wir das "eigentliche" Programm in ein Unterprogramm legen - einfach um die "Technik" kennen zu lernen, um größere Programme strukturiert und damit übersichtlicher zu gestalten.
Aufgerufen wird das Programm immer über den OB1, den wir aber erst im 2. Schritt programmieren werden.
Zuerst muss das Unterprogramm, das ja aufgerufen werden soll, angelegt werden. Denn was es noch nicht gibt, kann man auch nicht aufrufen. Deshalb wenden wir uns zuerst dem Unterprogramm - einer Funktion (FC) - zu.

Unter <SPS>|<NEU> markieren Sie <FC>. Rechts im Fenster steht "1", da es der erste FC-Baustein

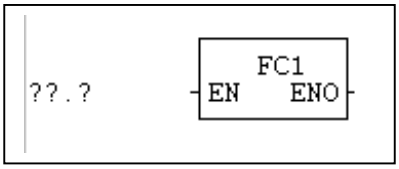


ist.

Nachdem Sie den neuen "FC1"-Baustein angelegt haben, müssen Sie diesen neuen Stand der Projektierung speichern, damit das restliche Programm "weiß", dass es den neuen Baustein gibt.
Jetzt kehren wir zurück zum OB1 (oder öffnen ihn mit <SPS>|<Öffnen> und Doppelklick auf <OB1>).
Nach einem Klick in das Editierfenster (nicht in das Kommentarfenster!) placieren Sie - wie vorstehend gezeigt - den FC1-Aufruf.

OB1-Netzwerk

NETZWERK 1: Aufruf des Unterprogramms FC1 (Sollwertsteller)



In AWL, aus FUP übersetzt:

```
CALL FC 1
NOP 0
```

Mehr ist für den Wechsel vom OB1-Hauptprogramm zum (hier: alleinigen) Unterprogramm FC1 nicht erforderlich.

Anschließend wechseln wir in das Fenster FC1 und tragen dort im Kopf die Variablen ein. Da von außen keine Variablen abgefragt oder an andere Bausteine übergeben werden müssen, sind nur solche Variablen zu deklarieren, die intern (in diesem FC1-Baustein) verwendet werden. Diese werden als "temp"-Werte deklariert.

FC 1 Netzwerk 1 Maximalwert-Kontrolle

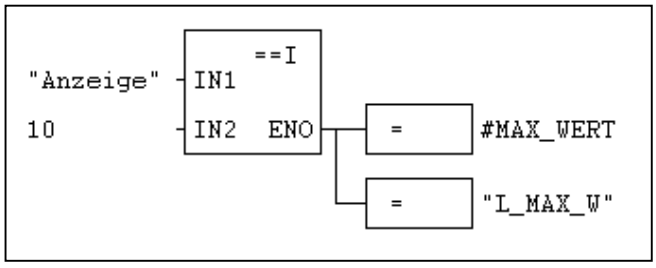
Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
	in				
	out				
	in_out				
0.0	temp	MAX_WERT	BOOL		Maximalwert ist erreicht
0.1	temp	MIN_WERT	BOOL		Minimalwert ist erreicht

Wir wollen den Maximal- und den Minimalwert erfassen. In die erste "temp"-Zeile schreiben Sie bitte als Name „MAX_WERT“. Der Datentyp ist „BOOL“. Geben Sie auch einen sinnvollen Kommentar ein und drücken dann die <ENTER>-Taste. Dadurch öffnet sich die nächste Zeile des selben Deklarationstyps. Legen Sie bitte eine Zeile für den Minimalwert an. (Sie können das Deklarationsfenster im Kopf wie alle Teilfenster unter Windows vergrößern oder verkleinern. Das ist vor allem wichtig, falls Sie es einmal unbedacht ganz geschlossen haben und bei einer späteren Sitzung grübeln, wo das Fenster geblieben ist. Sie können den Rahmen durch vorsichtiges Suchen unterhalb der Kopfleiste finden und nach unten ziehen).

Das Programm wird jetzt ganz normal - also wie Sie es gewöhnt sind - in verschiedenen Netzwerken innerhalb des FC1-Bausteins geschrieben. Am Ende dieses Bausteins kehrt das Programm automatisch zum aufrufenden Baustein (hier: OB1) zurück.

FC-Netzwerke:

NETZWERK 1: Maximalwert-Kontrolle



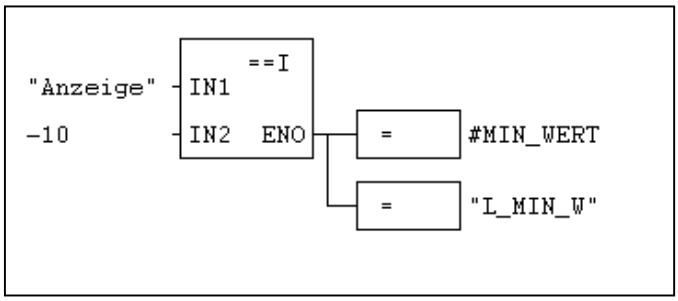
Im AW 512 steht der aktuelle Wert. Er wird verglichen mit der Zahl "10".

Grundsätzlich gilt:
Die 16 niederwertigen Bits von Akku 1 und Akku 2 werden miteinander verglichen, und wenn sie gleich sind, wird das VKE auf „1“ gesetzt, andernfalls auf „0“.

Vgl. TrySim <Hilfe>|<Index>| "Operationen", < ==I >.

Sie sehen, dass 2 parallele Ausgangsoperanden programmiert wurden.
Beträgt der aktuelle Sollwert im AW 512 den Wert "+10", ist der "L_MAX_W" -Pegel (A 0.1) "1". Aus dem FC heraus können also direkt Ausgänge angesprochen werden.
Der weitere "Ausgang" "#MAX_WERT" ist der Wert, den Sie im Deklarationskopf "geschaffen" haben.
Durch das vorgestellte "#" erkennt das Programm, dass Sie diese interne Variable meinen.
Dieser Pegel wird im Netzwerk 3 zur Begrenzung verwendet.

NETZWERK 2: Minimalwert-Kontrolle



Int-Werte können Zahlen zwischen - 32768 bis +32767 sein. Deshalb eignet sich der Baustein auch zum Vergleich negativer Zahlen.

Beträgt der aktuelle Sollwert im AW 512 den Wert "-10", ist der "L_MIN_W"-Pegel (A 0.0) "1".

Netzwerk 1 und 2 werden vorangestellt, damit die Werte aus dem letzten Zyklus richtig ausgewertet werden und im Netzwerk 3 die aktuellen Verknüpfungsergebnisse vorliegen.

Das folgende Programm wurde mit symbolischen Adressen geschrieben. Sie können auch hier den Operanden mit Mausklick zuweisen: Schreiben Sie in der 1. Zeile die Operation (hier: "U"), markieren bei ausgeschalteter Anlage den entsprechenden Taster und öffnen im Editierfenster mit <rM> das Kontextmenü. In der letzten Zeile steht die vorher ausgewählte Adresse. Mit <LM> placieren Sie diese hinter "U".
Wenn Sie "von Hand" die symbolische Adresse schreiben, müssen Sie genau auf die Schreibweise in der Symboltabelle achten. Es wird zwischen Groß- und Kleinbuchstaben unterschieden. (Sie können auch die symbolische Darstellung bei aktivem Editierfenster mit <LM> auf <Ansicht> <Symbolische Darstellung> ausschalten).

Netzwerk 3: Das eigentliche Programm

```
// Auswahl der Sprungmarken
      U  "T_( 6 )"          // Wenn die Taste <Vorwahl 6> gedrückt wird, ...
      SPB   w6              // (ohne Doppelpunkt!) ...spring bedingt zur Sprungsmarke <w6:>.
      U  "T_(-5)"                   // Wenn die Taste <Vorwahl -5> gedrückt wird, ...
      SPB   w_5             // ...spring bedingt zur Sprungmarke <w_5:>.

      U  "T_mehr"           //Wenn der Taster <mehr> betätigt wird...
      UN #MAX_WERT          // und  der Maximalwert nicht erreicht wurde ...
                             (Dieser Wert wurde im Netzwerk 1 bestimmt).
       SPB   add            //...springe bedingt zur Marke <add:>.
      U  "Tweniger"         //Wenn der Taster <mehr> betätigt wird...
      UN #MIN_WERT          // ...und der Minimalwert nicht anliegt ...
      SPB   sub             //...springe bedingt zur Marke <sub:>.
      SPA   end             //Sonst springe absolut zur Marke <end:>.

// Erzeugung der Zahl 6 ("w6:" ist die Sprungmarke)
w6:   L   6                 // Lade den Wert 6 (Dezimalzahl) ...
      SPA   aw              // ... und springe unbedingt zur Marke <aw:>

// Erzeugung der Zahl -5
w_5:  L   -5                // Lade den Wert -5 ...
      SPA   aw              // ... und springe absolut zur Marke <aw:>.

// Zunahme der Zahl um den Wert 1
add:  L   "Anzeige"         // Lade der Wert, der angezeigt wird
      L   1                 // Lade den Wert 1
      +I                    // addiere die Werte ...
      SPA   aw              // ... und springe absolut zur Marke <aw:>.
```

In jedem Zyklus wird der Wert des Ausgangswortes um 1 erhöht. Der Sollwertes wird also mit der Zyklus- bzw. Rechnergeschwindigkeit verstellt..

```
// Abnahme der Zahl um den Wert 1
sub:    L  "Anzeige"            // Lade der Wert, der angezeigt wird
        L  1                    // Lade den Wert 1
        -I                      // subtrahiere den letzten Wert (1) vom ersten.
                                // Hier ist kein Sprung mehr erforderlich, weil sowieso die nächste
                                // Programmzeile bearbeitet werden soll.

// Digitalanzeige
aw:     T  "Anzeige"            // Der Akkuinhalt wird an den Ausgang <Anzeige> geleitet.

// Sprungziel, falls nicht anderes mehr zu erledigen ist
end:    NOP 0                   // Die NullOperation besagt, dass hier nichts mehr passiert.
```

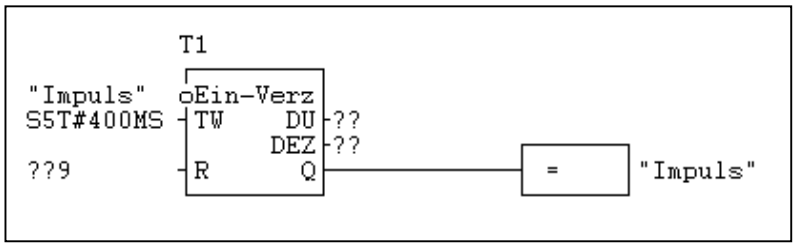
Wenn man in AWL programmiert, benutzt man den Befehl "NOP 0" meistens deswegen, weil in einer Zeile mit Sprungmarke ein Befehl stehen muss.

In diesem Programm wird bisher die Änderung des Ausgangswertes nur durch die Zykluszeit begrenzt. In der Praxis könnte sich bei schnellen Rechnern die Sollwerteinstellung als problematisch erweisen, weil sich die Werte mit der Zykluszeit zu schnell ändern. Deshalb planen wir (als Übung und Vertiefung) zusätzlich eine "Weiterschaltbremse" ein.

Dazu erweitern wir die Symboltabelle

Symbolische Adresse	Operand	Datentyp	Kommentar
Schnell	E 0.4	BOOL	Schalter langsam = 0 / schnell = 1
Impuls	M 2.0	BOOL	Alle 0,4 sec für ein Zyklus "1"

Netzwerk 4: Zwangspause beim Zählen



Der Merker "Impuls" blitzt automatisch alle 0,4 s für einen Zyklus lang auf. Da beim ungesteuerten Einschalten der Anlage "Impuls" "0" führt, schaltet T1 zeitverzögert ein, was aber in einer Art Selbstmordschaltung das Zeitglied sofort abschaltet, bis nach weiteren 0.4 s das Spielchen ungesteuert wieder beginnt.

Es bleibt die Aufgabe, das vorstehende Netzwerk 3 so zu erweitern, dass die Wahl zwischen dem schnellen Durchlauf in Zykluszeit oder dem verzögerten Weiterschalten möglich wird.

Versuchen Sie es bitte selbst. Wenn es gar nicht klappt - hier ein Vorschlag:

Änderungen im Netzwerk 3:

Netzwerk 3: Das eigentliche Programm

```
// Auswahl der Sprungmarken
      U  "T_( 6 )"          // Wenn die Taste <Vorwahl 6> gedrückt wird, ...
      SPB   w6              // (ohne Doppelpunkt!) ...springe bedingt zur Sprungsmarke <w6:>.
      U  "T_(-5)"                   // Wenn die Taste <Vorwahl -5> gedrückt wird, ...
      SPB   w_5             // ...springe bedingt zur Sprungmarke <w_5:>.
```

Hier beginnt der Einschub:

```
      O  "Schnell"      // entweder schnell
      O  "Impuls"       // oder Impuls
      SPBN  end         // sonst passiert nichts;
```

Nur wenn der Taster <Schnell> betätigt wurde oder ein Impuls von T1 kommt, wird das Programm zeilenweise weiter abgearbeitet. Sonst wird der Rest übersprungen und der Anzeigewert ändert sich nicht.
Das ist schon alles. Das restliche Programm bleibt erhalten.

vgl. TrySim-Projekt <Lösungen>|<Sollwert>

Sie haben gelernt:

- Sprungmarken setzen
- Sprungziel programmieren
- Mehrere Anweisungen in einem Netzwerk in AWL programmieren
- FC erstellen, speichern und aufrufen
- Variablen in FC deklarieren
- Vergleicher programmieren
- Sprungbefehle programmieren
- Laden und Transferieren von Werten
- „NOP 0“ anwenden
- Zykluszeiten ändern

Projekt 23: Zahlendarstellung

Schwerpunkte: Byte, Word, Zahlensysteme, Darstellung von DUAL- und BCD-Zahlen

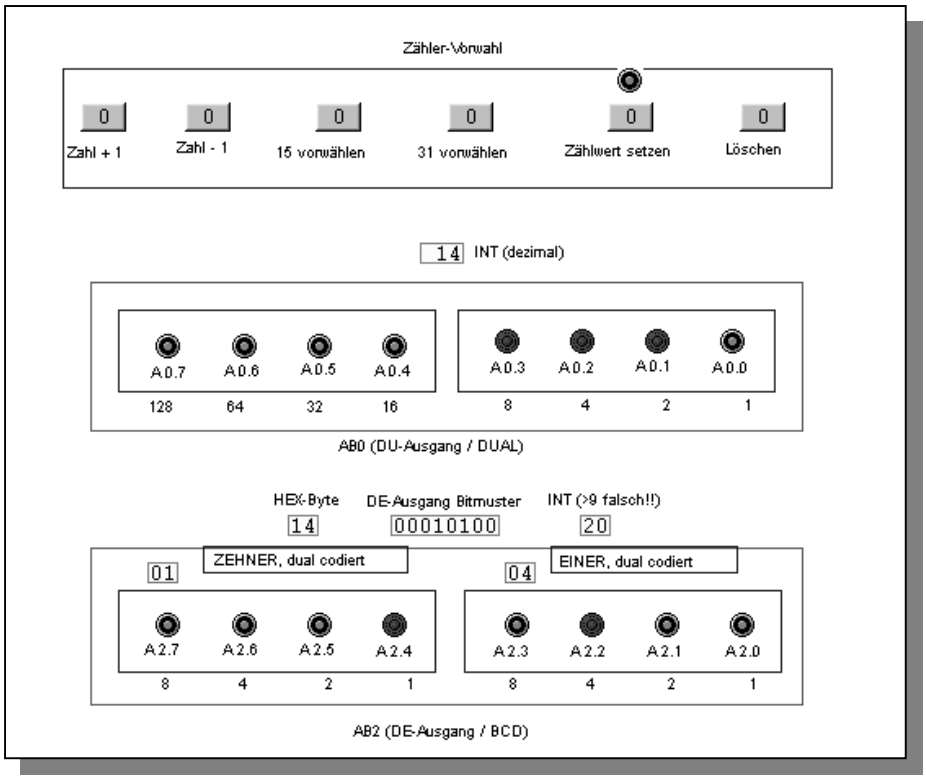
Aufgabe:
Ein Vor- u. Rückwärtszähler soll vollständig angeschlossen werden. Sein Zählwert soll über beide Ausgänge <DU> und <DE> richtig auf Anzeigen gelegt werden.
Über Tasterbefehle können 2 unterschiedliche Zahlen durch Sprungbefehle vorgewählt werden.
Diese Zahlen können in Einzelschritten um 1 erhöht oder erniedrigt werden.
Der jeweilige Zählerstand wird als Wort gespeichert und an die Digitalanzeige ausgegeben.
Vergleichen Sie die unterschiedlichen Zahlencodes DUAL und BCD.

Zur Umsetzung der Aufgabe mit TrySim:
Dieses Projekt wurde bereits in der fertigen Version im vorherigen Abschnitt zur Anschauung des BCD-Codes verwendet. Jetzt soll dieses Projekt erläutert werden.
Wählen Sie bitte aus dem Ordner **<Vorlagen>** das Projekt **<Zähler_1_V>**, speichern Sie es als neues Projekt in Ihrem eigenen Verzeichnis und öffnen Sie es dort in gewohnter Weise. Die "Anlage" ist wie folgt vorbereitet.

Symbolische Adresse	Operand	Datentyp	Kommentar
+ 1	E 0.5	BOOL	zähle 1 hinzu
- 1	E 0.4	BOOL	ziehe 1 ab
DUAL	AW 512	INT	Zählerwert dual codiert
Wähle 31	E 0.0	BOOL	Vorwahl des Zählwertes auf 31
Lösch	E 0.3	BOOL	Rücksetzen des Zählers auf 0
Wähle 15	E 0.6	BOOL	Vorwahl des Zählerwertes auf 15
DEZI	AW 516	WORD	Zählerwert BCD codiert
Z.setzen	E 0.1	BOOL	Zählervorwahl setzen
ZW_WAHL	AW 20	WORD	vorgewählter ZW-Wert

Nur der Vollständigkeit halber aufgeführt, damit sie nicht aus Versehen doppelt belegt werden:

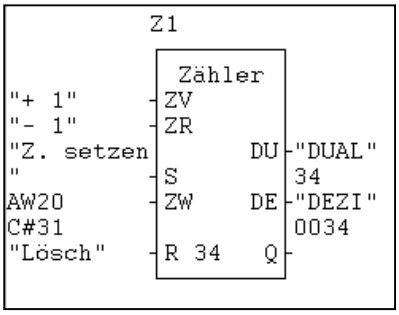
AB 6	AB 6	BYTE	Zehner-Anzeige, HEX codiert
AB 4	AB 4	BYTE	Einer-Anzeige, HEX codiert



Netzwerk 1: Auswahl der Zählervorwahl ZW. Über Sprungfunktionen wird entschieden, ob bzw. welche Zahl vorgewählt werden soll.

```
      U  "Wähle 15"        //Wenn Taster <15 vorwählen> betätigt wird,
      S  A   1.0           //…setze den Ausgang A 1.0
      SPBN  _001           //Wenn nicht, springe zur Marke <_001>.
      L   C#15             //Wenn ja, setze hier fort: Lade BCD-codiert die Zahl 15
      T   "ZW_WAHL"        //...und transformiere sie ins AW 20.
      SPB   ende           //Springe (bedingt) zur Marke <ende>.
_001: U  "Wähle 31"
      S  A   1.0
      SPBN  ende
      L   C#31
      T   "ZW_WAHL"
ende: NOP 0
```

Netzwerk 2: Anschluss des Zählerbausteins



Schalten Sie nebenstehendes Netzwerk auf die AWL-Ansicht um. Dann können Sie die tatsächlich verarbeiteten Werte nachvollziehen. Die nebenstehenden Anzeigewerte sind immer Dezimalzahlen.
Aber Vorsicht: Falls Sie den Wert z.B. als Merkerwort weiter verwenden wollen, müssen Sie höllisch aufpassen, in welchem Format der Wert tatsächlich vorliegt.

Vergessen Sie nicht, den Zähler zu "instanziieren", d.h. in diesem Beispiel, als "Z1" zu deklarieren.

```
U  "+ 1"          // Wenn die Taste gedrückt wird ("1"-Signal)...
ZV Z   1          // ...zähle +1
U  "- 1"          // Wenn die Taste gedrückt wird ("1"-Signal)...
ZR Z   1          // ...zähle -1
U  "Z. setzen"    // Wenn Taste gedrückt wird ("1")...        AKKU-WERTE
L   AW  20        // ...lade den Wert vom AW 20 ...                31 (BCD-codiert, C#...)
S  Z   1          // ...und setzte den Zähler Z1 auf diesen Wert.
U  "Lösch"        // Wenn Taste gedrückt wird ("1")...
R  Z   1          // ...setze den Zähler Z1 auf 0 zurück
L   Z   1         // Der Zählerwert ...                           22 (hier Anzeige als HEX)
T   "DUAL"        // ...wird als Dualzahl am Ausgang DU ausgegeben.   22
LC  Z   1         // Der Zählerwert ...                           34 (hier Anzeige als Dezimalzahl)
T   "DEZI"        // ... wird als BCD-Wort an DE ausgegeben.          34
NOP 0
```

Netzwerk 1 wurde hier als AWL umgeschaltet. Die letzte "NOP 0"-Zeile erscheint bei der Übersetzung, weil der Ausgang Q nicht belegt wurde.
Hier endet der eigentliche Projektauftrag für den Zähleranschluss. Es folgen die Anzeige und die Zahlwahl.

Netzwerk 2:
Zuordnung der Zahlen zu Ausgangswörtern. Die LEDs sind die Anzeigen dieser Wörter.

```
L   "DUAL"
T   AB  0
L   "DEZI"
T   AB  2
```

Lade das dual codierte Ausgangswort und transferiere den Wert zum Ausgangs-BYTE 0.
Lade das BCD-codierte Ausgangswort und Transferiere den Wert zum Ausgangs-BYTE 2.
(Es wird jeweils nur das rechte Byte des Wortes übertragen).

```
L   AW  512
T   AB  0
L   AW  516
T   AB  2
```

Symb. Adressen reale Adressen

AW 512 und AW 516 sind ziemlich willkürlich. Man könnte hier auch z. B. Merkerwörter (MW ...) zuordnen.
Sie wurden genommen, weil TrySim diese Wörter für die Digitalanzeigen vorgeschlagen hat. Sowohl am Zähler, als auch an der Anzeige können die Wörter geändert werden.

Netzwerk 3: Zerlegung des AB2 (BCD-codiert) in 2 neue Bytes.
Hier werden die **niederen** Bits zum neuen Halb-Byte 4 (nibble) definiert. (BCD-Code für die EINER-Anzeige)

```
U  A   2.0
=  A   4.0
U  A   2.1
=  A   4.1
U  A   2.2
=  A   4.2
U  A   2.3
=  A   4.3
```

Netzwerk 4: Zerlegung des AB2 (BCD-codiert) in 2 neue Bytes.
Hier werden die **höheren** Bits zum neuen Halb-Byte 6 (nibble)definiert. (BCD-Code für die Zehner-Anzeige)

```
U  A   2.4
=  A   6.0
U  A   2.5
=  A   6.1
U  A   2.6
=  A   6.2
U  A   2.7
=  A   6.3
```

Durch diese Zerlegung in 2 neue Bytes ist es möglich, beide Anteile getrennt zur Anzeige zu bringen. In beiden Bytes werden die rechten (unteren) Bits belegt. Die höheren Bits bleiben "0". Somit geben AB 4 die Einer und AB 6 die Zehner dual codiert wieder. Diese Werte lassen sich einfach den Digitalanzeigen zuordnen.

Netzwerk 5: Auswahl der Zählervorwahl ZW
Über Sprungfunktionen wird entschieden, ob bzw. welche Zahl vorgewählt werden soll.

```
      U  "Wähle 15"    // Wenn der Taster gedrückt wird (TRUE = 1) ...
      S  A   1.0       // ...setze den Ausgang (Blinker) dauerhaft auf TRUE
      SPBN  _001       // Wenn die Bedingung nicht erfüllt wurde, springe zur Marke <_001:>
      L   C#15         // sonst lade den Wert 15 ... (als BCD-Wert: C#...)
      T   AW  20       // ...und transferiere ihn zum Ausgangswort AW 20.
      SPB   ende       // Springe nach vorstehender Anweisung zur Marke <ende>
                         (vgl. Befehl SPA, BEA)
_001: U  "Wähle 31"    // Marke <_001>; Wenn der Taster gedrückt wird (TRUE = 1) ...
       S  A   1.0      // ...setze den Ausgang (Blinker) dauerhaft auf TRUE
      SPBN  ende       // Wenn die Bedingung nicht erfüllt wurde, springe zur Marke <ende:>
      L   C#31         // sonst lade die Konstante 31 ...
      T   AW  20       // ...und transferiere sie zum Ausgangswort AW 20.
                         (also eine alternative Wahl).
ende: NOP 0            // hier passiert nichts mehr. Hinter einer Sprungadresse muss eine Anweisung
                          stehen, notfalls eine Leeranweisung (Null-Operation).
```

Die hier verwendeten Sprungadressen erlauben das Abarbeiten der Programmzeilen in beliebig (geplanter!) Weise; nicht wie üblich, von oben nach unten, sondern direkt zum Sprungziel, der **Sprungmarke.**
Die Auswahl der möglichen Sprünge finden Sie unter <Hilfe>|<Index>| "SP".

Sprungmarken müssen im selben Baustein liegen. Sie dürfen max. 4 Zeichen lang sein und nur aus Buchstaben, Zahlen und dem Unterstrich bestehen. Das Programm erkennt eine Sprungmarke an dem Doppelpunkt, gefolgt von der ersten Anweisung.

Netzwerk 6: (Zähler wird gesetzt), Erinnerungsblinker wird zurückgesetzt

```
U  "Z. setzen"     //Das ist der Setzbefehl für die Übernahme der vorgewählten Zahl
R  A   1.0         // gleichzeitig wird der Blinker zurückgesetzt.
```

Vgl. TrySim_Projekt <Zähler_1> im Verzeichnis <Lösungen>.

Sie haben gelernt:

- Arbeiten mit Sprungfunktionen
- Vor- / Rückwärtszähler programmieren
- <DU>- und <DE>-Abfragen
- AKKU-Werte interpretieren
- Konstante „C#“ laden
- Zerlegung eines Bytes in Nibbles

Projekt 24: **<AbfüllAnlage>**
vgl. <Lösungen>|<AbfüllAnlage>

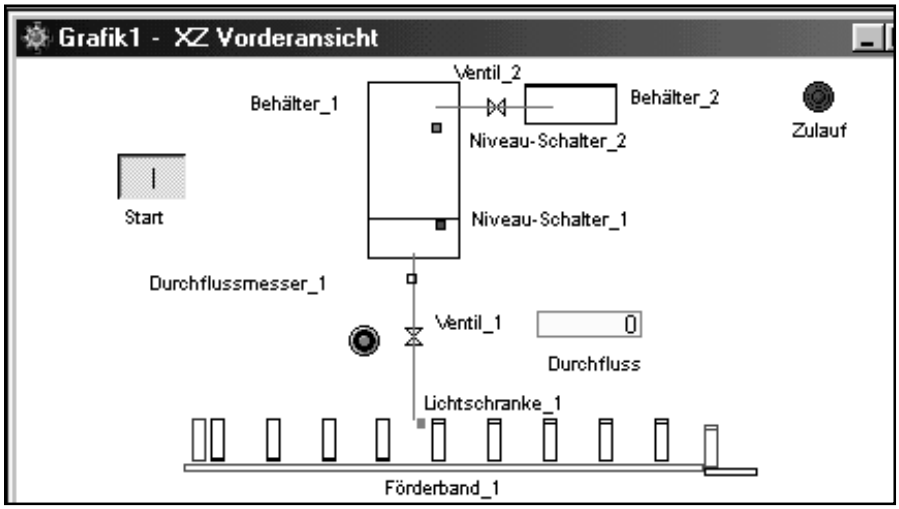
Schwerpunkte: Funktionen, SR_FF, Impuls

Aufgabe:

1.) Wenn Behälter_1 leer ist, wird er über das Ventil_2 wieder gefüllt
2.) Wenn die Lichtschranke bedeckt ist, wird für 2,5 sec das Dosierventil geöffnet
3.) Das Förderband läuft, wenn die Lichtschranke frei ist, oder wenn der Dosiervorgang beendet ist.

Natürlich ist eine solche Steuerung für eine wirkliche Anlage nicht im entferntesten ausreichend. Wenn Sie Lust haben, machen Sie es besser !

Die Anlage und die Symboltabelle sind bereits vorbereitet unter <Vorlagen>|<AbfüllAnlage_V>.
Sie können dieses Projekt öffnen und in Ihr Arbeitsverzeichnis kopieren, um es dort zu programmieren.
Hinweis für Selbstbauer:
Damit Flüssigkeit eingefüllt werden kann, muss das Rohrende mit dem Dynamik überlappen.
Nicht nur in der einen Ansicht müssen sich die Anlagenteile "treffen". Egal, in welcher Ansicht Sie ein Rohr etc. einbauen, jedes Teil "fällt" bis auf den Unter- / Hintergrund und muss in einer weiteren Ansicht räumlich richtig angeordnet werden. Dazu müssen Sie das Teil bei ausgeschalteter Anlage mit <LM> "greifen" und verschieben, bis es in allen Ebenen passt.



Symboltabelle

Nummer	Symbol	Adresse	Typ	Kommentar
1	Start	E 0.3	BOOL	Schalter Start
2	LS	E 0.2	BOOL	Lichtschranke
3	NivMin	E 0.0	BOOL	Niveau Min
4	NivMax	E 0.1	BOOL	Niveau Max
5	Y NachF	A 0.6	BOOL	Nachfüllventil
6	Y Dosier	A 0.5	BOOL	Dosierventil
7	G Flasche	A 0.0	BOOL	Flasche erzeugen
8	T Füll	T 1	TIMER	Füllzeit

OB 2; Netzwerk 1: Flaschen-Generator starten

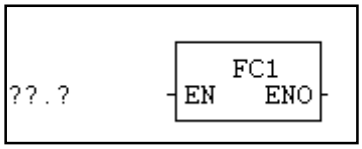
Der OB 2 ist für simulationsbedingten Code vorgesehen.
Er wird **vor** dem OB 1 aufgerufen. (vgl. <TrySim> <Hilfe> <Index> "OB")

```
U  "Start"          //Wenn der <START>-Schalter betätigt wird...
=  "G Flasche"      //...werden im Generator über A 0.0 im dort eingestellten Zeittakt ...
                    //..."Flaschen" erzeugt.
```

Obwohl es für ein so kleines Programm eigentlich nicht erforderlich ist, wurde das "eigentliche" Programm aus Übungszwecken in FCs "verpackt". (Zum Sinn bzw. Unsinn, diese Verpackung zum Dogma zu erheben, möchte ich mich hier nicht auslassen).

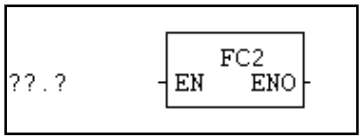
Aus früheren Projekten wissen Sie, dass zuerst die FCs neu erstellt und abgespeichert werden müssen mit <SPS>|<NEU>|<FC>. Erst danach können diese aufgerufen werden.

OB 1; Netzwerk 1: Nachfüllung aufrufen



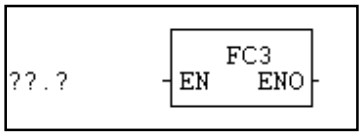
Sprungbefehl zur FC 1

OB 1; Netzwerk 2: Bandsteuerung aufrufen



Sprungbefehl zur FC 2

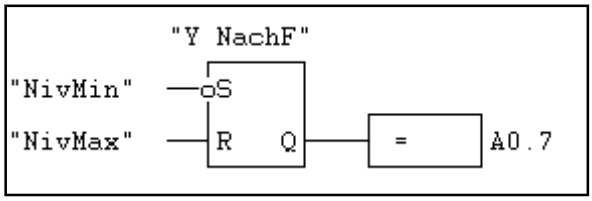
OB 1; Netzwerk 3: Bandsteuerung



Sprungbefehl zur FC 3

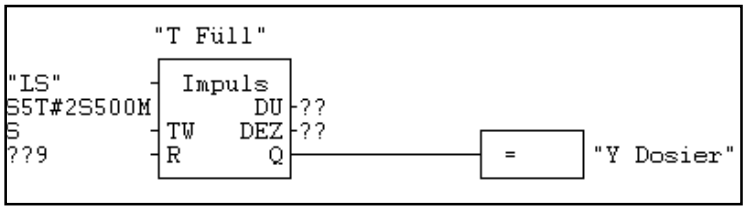
Jetzt folgen die Programme.

FC 1; Netzwerk 1: Nachfüllen



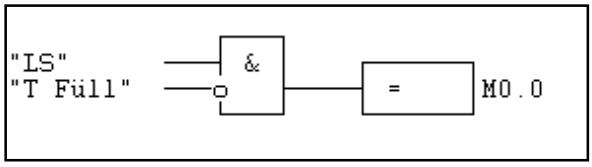
Es wird nachgefüllt, wenn der untere Niveauschalter nicht mehr bedeckt ist.
Das Nachfüllen wird beendet, wenn der obere Niveauschalter bedeckt ist.
Das Ventil (A 0.6; "Y NachF") kann direkt mit dem SR-FF verbunden werden. Es könnte auch parallel zur Kontroll-LED (A 0.7) parallel gelgt werden. Dann muss allerdings das SR-FF mit einem Merkeroperanden versehen werden.

FC 2; Netzwerk 1: Füll-Vorgang



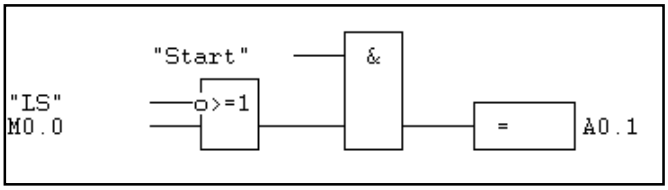
Wenn die Lichtschranke von einer Flasche angefahren wird, öffnet sich für eine eingestellte Impulszeit das Dosierventil.

FC 2; Netzwerk 2: Füllvorgang abgeschlossen



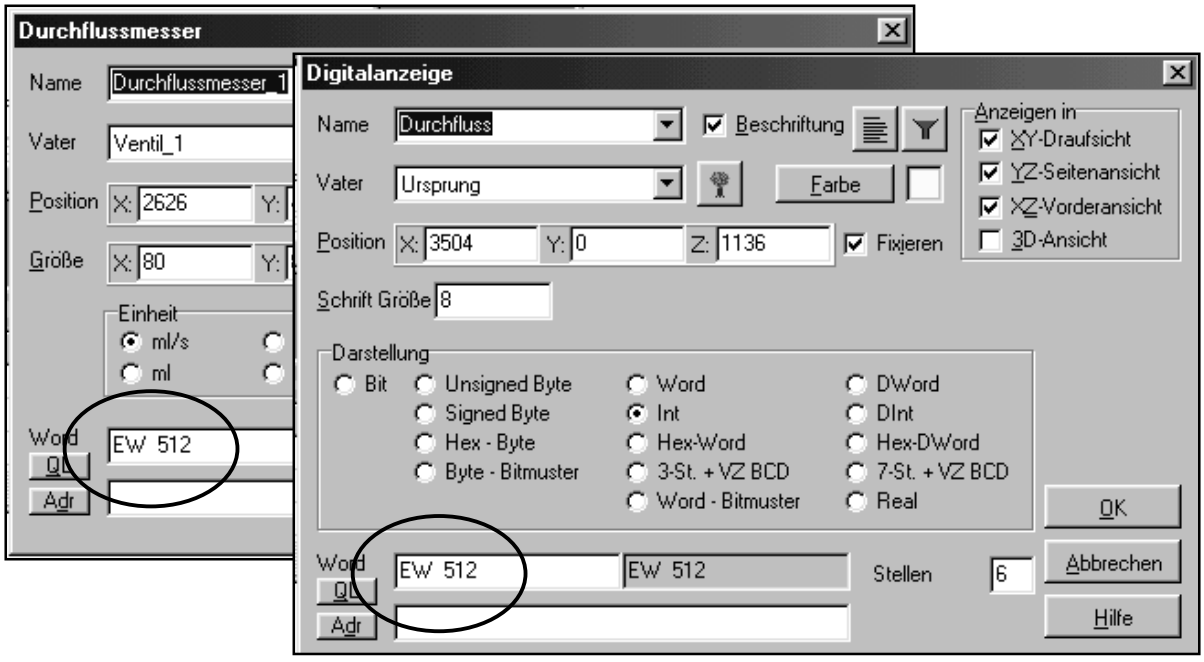
Wenn eine Flasche in der Füllstation steht ("LS" = "1") und die Füllzeit abgelaufen ist, springt der Merker auf "1".

FC 3; Netzwerk 1: Band-Steuerung



Wenn der <START>-Schalter betätigt und vorstehende Füllvorgang abgeschlossen wurde, läuft das Band. Nachdem die Flasche die Füllstation verlassen hat, liegt an <LS> logisch "0" an. Dadurch bleibt das Band eingeschaltet, auch wenn *M 0.0* auf "0" wechselt. (Die "1"-Bedingung im Netzwerk 2, FC 2 ist nicht mehr gegeben).

Die Durchflussanzeige taucht im Programm gar nicht auf, weil das Wort des Durchflussmessers (EW 512) direkt an die Digitalanzeige "gesteckt" wurde.



Projekt 25: Mischung im Flüssigkeitsbehälter

Schwerpunkte: Vergleicher <I, >I, S, R

Hinweis:
Dies ist ein Beispiel, das Behälter, Rohre, Ventile, einen Füllstandsmesser und einen Analysator enthält. Bitte lesen Sie in der TrySim-Hilfe über die weiteren Elemente für Flüssigkeiten. Das Beispiel stellt keine ernst zu nehmende Regelungsanlage dar, sondern soll Sie über weitere Möglichkeiten des Anlagenbaus von TrySim informieren.

Aufgabe: Mischung von Flüssigkeiten
In einem Mischbehälter soll Wasser mit Farbe in einem bestimmten Verhältnis – mit bestimmten Toleranzen - gemischt werden. Dazu wird der Zufluss über zwei Pumpen (Ventile) geregelt. **Über die Steuerung des Wasser-Ventils wird der Prozess in Gang gesetzt.** Ist der Farbanteil zu niedrig, wird das Farb-Ventil geöffnet. Ist er zu hoch, wird das Farb-Ventil geschlossen.
Falls der Mischbehälter die gewünschte maximale Menge enthält, soll sich das Ablauf-Ventil öffnen. Dabei kann der Zufluss geregelt bestehen bleiben, oder das Zulauf-Wasser wird abgeschaltet. Bei abgeschaltetem Wasser sinkt der Flüssigkeitspegel, da mehr abfließt, als über den Farbanteil zufließt. Das Mischungsverhältnis darf sich nur in den vorgegeben Toleranzen ändern, d.h., es darf nicht nur Farbe nachlaufen. Als Folge wird also auch das Farbventil geschlossen.

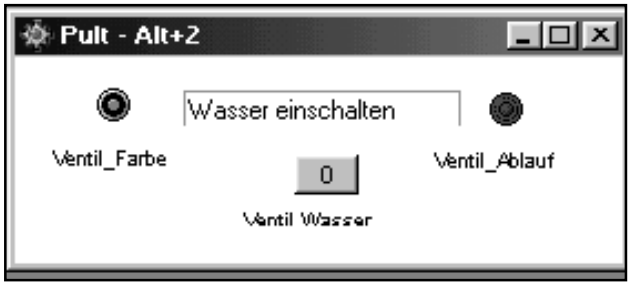
Wenn das Wasser eingeschaltet bleibt, pendelt der Stand der Flüssigkeit, weil die Farbpumpe mehr fördert, als die Wasserpumpe. Beide zusammen lassen mehr zufließen als abfließt. Auch wenn in dieser Situation die Farbkonzentration zu groß wird, schaltet das Farb-Ventil ab. Die momentanen Durchflussmengen werden angezeigt.

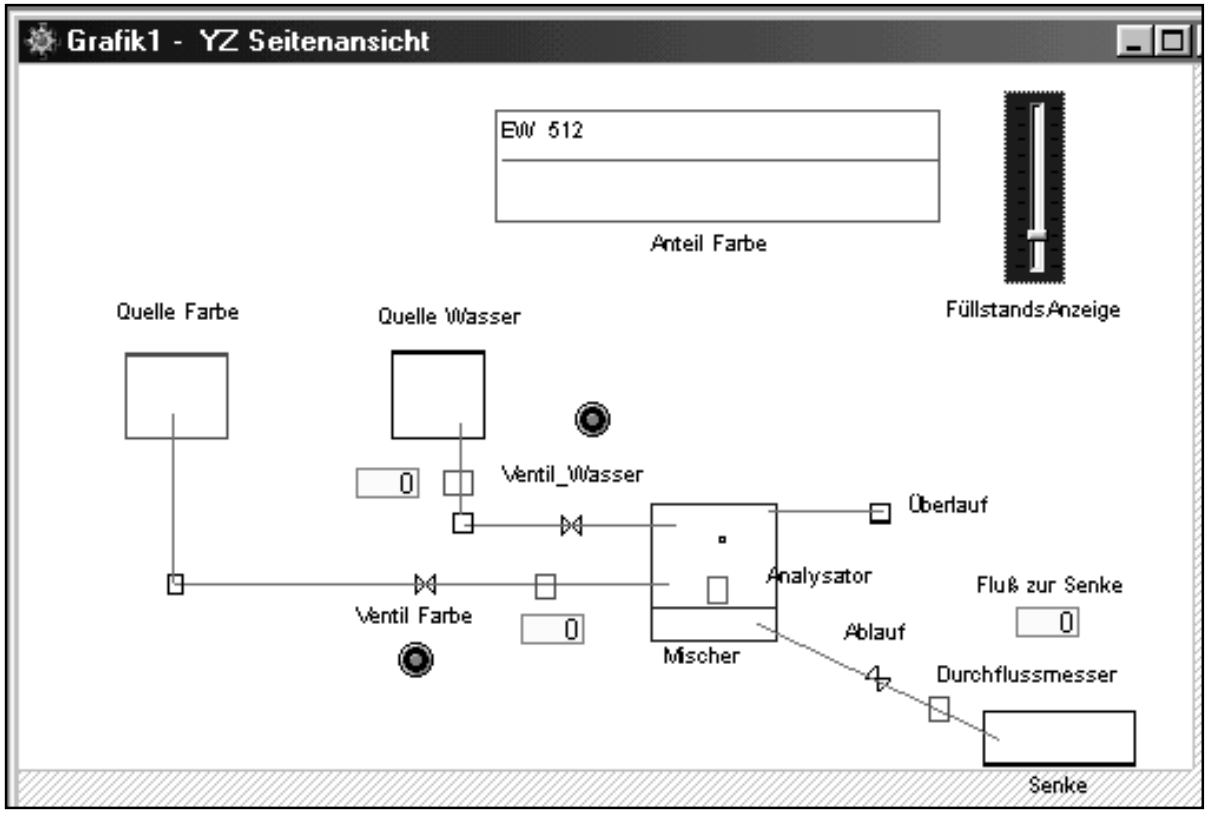
Im Verzeichnis <Vorlagen> ist das Projekt unter <Fluids_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Symboltabelle und die Anlage sind bereits vorbereitet.

Symbolische Adresse	Operand	Datentyp	Kommentar
FarbVent	A 0.0	BOOL	Ventil Farbe
WasserV	A 0.1	BOOL	Ventil Wasser
AblaufV	A 0.2	BOOL	AblaufVentil
LWasserV	A 0.3	BOOL	LED WasserVentil
LFarbV	A 0.4	BOOL	LED FarbVentil
Anteil_F	EW 512	WORD	Anteil Farbe
Senke_FL	EW 514	WORD	Anzeige Fluss zur Senke
Füllung	EW 516	WORD	Füllstand
TEXT	M 0.0	BOOL	Textanzeige

Es sind 2 Grafiken als Bedienung und als technische Anlage konzipiert:





Wichtig ist für eine Flüssigkeitsanlage, dass sie in der Seitenansicht erstellt wird. Alle Anlagenteile unterliegen beim „Einbau“ der Schwerkraft. Die Flüssigkeit soll nach unten sinken und dieser Vorgang soll im Mischer zu beobachten sein. Deshalb ist die Seitenansicht erforderlich. Andererseits „fallen“ alle Anlagenteile in der Ansicht, in der sie plaziert werden, bis an die rückseitige Begrenzung. Für die Rohrleitungen, Ventile, etc. reicht es nicht aus, sie in der Seitenansicht zu verbinden. Sie müssen dafür sorgen, dass sie sich auch räumlich, d.h. in der Tiefe, treffen. Das erreicht man, indem man die bisherige Konstruktion in einem anderen Grafikfenster – in einer anderen Betrachterebene – öffnet. Dort sieht die übersichtliche Anlage aus der Hauptkonstruktionsebene (hier: Seitenansicht) evtl. erschreckend unübersichtlich aus. Mit etwas räumlichem Vorstellungsvermögen findet man aber ein Einzelteil in den anderen Ansichten, wenn man es z.B. in der Seitenansicht markiert. Dieses Teil wird dann auch in den anderen Ansichten markiert. In dieser anderen Ansicht „greift“ man sich jetzt das Teil und verschiebt es soweit, dass es auch in dieser Ansicht zu den anderen Teilen passt.

In diesem Buch geht es aus Platzgründen aber nur am Rande um den Anlagenbau. Für weitere Konstruktionsfragen empfehle ich, die Bauteile mit <rM> anzuklicken und die Editierfenster zu betrachten, bzw. die <Hilfe> zu benutzen.

Der TrySim-eigene Analysator liefert ein Wort als %-Wert der Farbe in der Flüssigkeit. Es ist nicht erforderlich, dass Sie verstehen, wie dieser Wert gebildet wird. In der Praxis würde Ihnen auch nur ein Datenwort aus einem geeigneten Gerät geliefert, das es zu verarbeiten gilt.

OB 1; Netzwerk 1: Regelung der Mischung im Mischer

Bei unter 45 % wird die Pumpe (Ventil Farbe) eingeschaltet,
bei über 55 % wird sie wieder ausgeschaltet

```
L  45              //Lade den Wert 45
L  "Anteil_F"      //Lade das Eingangswort EW 512 (Analysator)
>I                 // wenn 45 > Wert von EW 512 ...
S  "FarbVent"      // ...dann setze den Ausgang A 0.0 (Farbe)

L  55              //Lade den Wert 55
>I                 // wenn 55 < Wert von EW 512 ...
                   // (Zeile "L EW 512" steht vor "L 55". Die Rechenoperation bezieht sich immer
                   // auf die letzten beiden Lade-Aufrufe )
R  "FarbVent"      // ...dann setze Ausgang A 0.0 zurück.
```

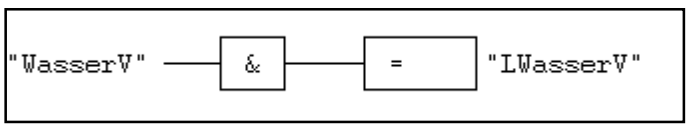
OB 1; Netzwerk 2: Steuerung des Zu- bzw. Ablaufes

Zu steuern ist die Anlage über den Wasserzulauf. Die Mischung wird automatisch im vorstehenden Netzwerk geregelt. Bei maximalem Flüssigkeitsstand wird das Ablaufventil geöffnet. Das Wasserventil kann dabei aus- bzw. wieder zugeschaltet werden.
Bei minimalem Flüssigkeitsstand wird das Ablaufventil geschlossen.
Die Regelung der Mischung führt zum Zu- bzw. Abschalten des Farbe-Ventils.
Die Pumpe der Farbe leistet dabei mehr als die des Wassers.

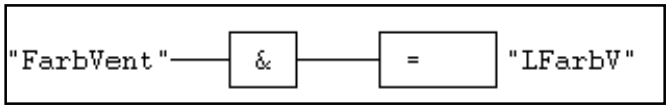
```
L  "Füllung"       //Lade das Eingangswort EW 516 (Füllstand)
L  140             //Lade den Wert 140
>I                 //wenn EW 516 > 140 ...
S  "AblaufV"       //... dann setzte den Ausgang A 0.2 (Ablauf)
L  "Füllung"       //Lade das Eingangswort 516
L  40              //Lade den Wert 40
<I                 //wenn EW 516 < 40 ...
R  "AblaufV"       //dann rücksetze A 0.2

UN "WasserV"       // Wenn das Wasser-Ventil ausgeschaltet ist
=  "TEXT"          // ... wird das Textfeld eingeblendet
```

M 0.0 („TEXT“) ist in der Editiermaske des Textfeldes (Grafikfenster <PULT>) die Adresse, um den Text anzuzeigen. Sie sehen in der Editiermaske, dass der Text gelöscht wird mit der Quittier-Adresse A 0.1 (Ventil Wasser EIN).

OB 1; Netzwerk 3: LED-Anzeige Wasserventil ist offen



OB 1; Netzwerk 4: LED-Anzeige Farbventil ist offen



vgl. <Lösungen>|<Fluids>

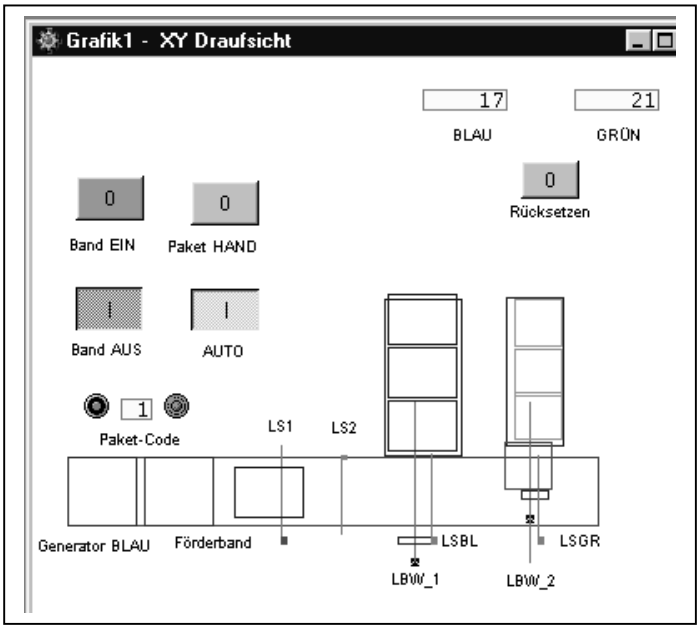
Projekt 26: SortierStrecke 1

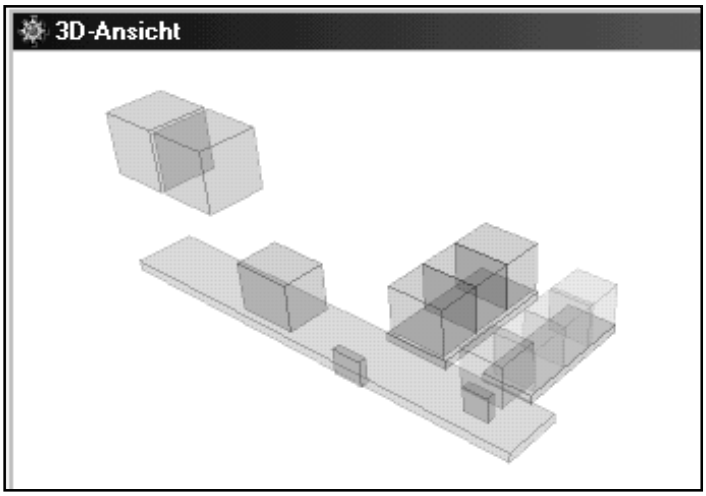
Schwerpunkte: Sprungfunktion SPB, Zufallsgenerator in TrySim, Lade L, Transferiere T, Vergleiche ==I, Impulsbildung mit SE

Aufgabe: Sortierstrecke
Auf einem Förderband sollen unterschiedlich große Pakete (2Typen) sortiert werden. Die Pakete kommen in unterschiedlicher Reihenfolge an und werden nach dem Zufallsprinzip von TrySim erzeugt.
(Die eigentliche Aufgabe ist nicht das Erzeugen, sondern das Sortieren der Pakete).
Die Pakete sind verschieden groß. Durch 2 Lichtschranken wird ihre Länge erfasst und dem entsprechend werden die Weichen angesteuert.
Im Automatikbetrieb wird das nächste Paket durch die Lichtschranke freigegeben, d.h., es muss zuerst ein Paket von Hand gesetzt werden.

Im Verzeichnis <Vorlagen> ist das Projekt unter <SortierStrecke_1_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

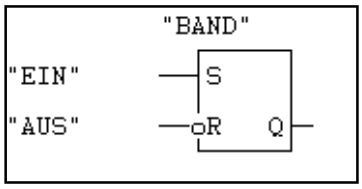
Die folgende Anlage und die Symboltabelle sind bereits vorbereite





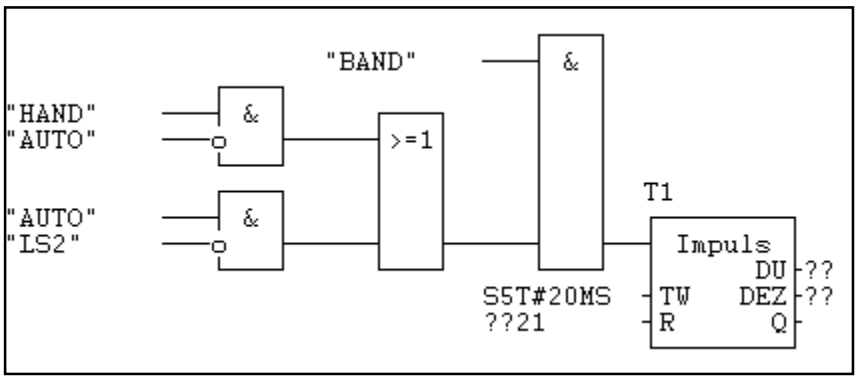
Symbolische Adresse	Operand	Datentyp	Kommentar
SchBL	A 0.0	BOOL	Ausstoßschieber BLAU <LBW_1>
BAND	A 0.1	BOOL	Förderband
MAGBL	A 1.0	BOOL	Magazin BLAU
MAGGR	A 1.1	BOOL	Magazin GRÜN
SchGRUN	A 1.5	BOOL	Ausstoßschieber GRÜN <LBW_2>
CodBL	AW 500	WORD	Code Magazin BLAU
CodGR	AW 502	WORD	Code Magazin GRÜN
PakCode	AW 514	WORD	Paket Code
ANZBL	AW 516	WORD	Anzeige Paketzahl BLAU
ANZGR	AW 518	WORD	Anzeige Paketzahl GRÜN
EIN	E 0.0	BOOL	Band EIN
HAND	E 0.1	BOOL	Paket Hand-(Einzel-)Steuerung
AUS	E 0.2	BOOL	Band AUS (Ö)
LS2	E 0.3	BOOL	Lichtschranke Längenmessung 2
AUTO	E 0.4	BOOL	Automatikbetrieb
ZR	E 0.5	BOOL	Zähler rücksetzten
LSBL	E 0.6	BOOL	Lichtschranke Station BLAU
LS1	E 0.7	BOOL	Lichtschranke Längenmessung 1
LSGR	E 1.0	BOOL	Lichtschranke Station GRÜN
LEDBL	M 1.0	BOOL	LED BLAU
LEDGR	M 1.1	BOOL	LED GRÜN

OB 1; Netzwerk 1: Bandsteuerung von Hand



Das Schütz für den Bandantrieb wird mit einem Tasterdruck auf <Band EIN> eingeschaltet. Das FlipFlop „BAND“ bleibt gesetzt, bis der Taster <AUS> (Öffner) betätigt wird.

OB 1; Netzwerk 2: Impulsbildug für den Zufallsgenerator



Wenn das Band läuft, wird entweder von Hand oder bei Automatikbetrieb ein Impuls erzeugt, der im nächsten Netzwerk für die Erzeugung der Zufallszahl benutzt wird.
<Paket Hand> darf nicht wirksam werden, wenn <AUTO> gewählt wurde, damit nicht unkoordiniert ein zusätzliches Paket den Ablauf stört. Bei <AUTO>-Betrieb wird der Impuls von der Lichtschranke erzeugt, wenn das vorherige Paket die Lichtschranke <LS 2> verlassen hat.

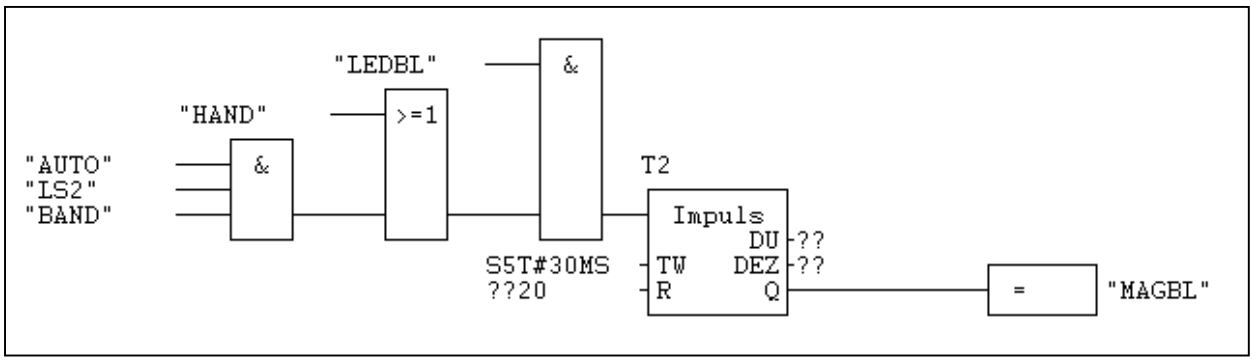
OB 1; Netzwerk 3: Zuordnung des Zufallswertes zu den Magazinen, bzw. zu den LEDs.

Um den Zusammenhang in einem Netzwerk darzustellen, wurde AWL gewählt.

```
    UN T   1          // Wenn der Timer nicht gesetzt ist...
    SPB   weit        // ...springe bedingt zur Marke <weit:>
    L   2             // sonst lade den Wert 2
    FUNC 100          // Interne Funktion: Zufallszahlen von 0 und 1
    T   „PakCode“     // Transferiere den Zufallswert ins Ausgangswort 514

weit: L   „PakCode“   // Lade den Wert des AW 514
    L   0             // Lade Wert 0
    ==I               // Vergleiche. Wenn gleich, dann...
    =  "LEDBL"        // ...setzte LED Blau auf TRUE (= „1“)
    L   “PakCode”
    L   1
    ==I               // Wenn Wert AW514 = 1, dann...
    =  "LEDGR"        // ...setzte LED Grün auf TRUE.
```

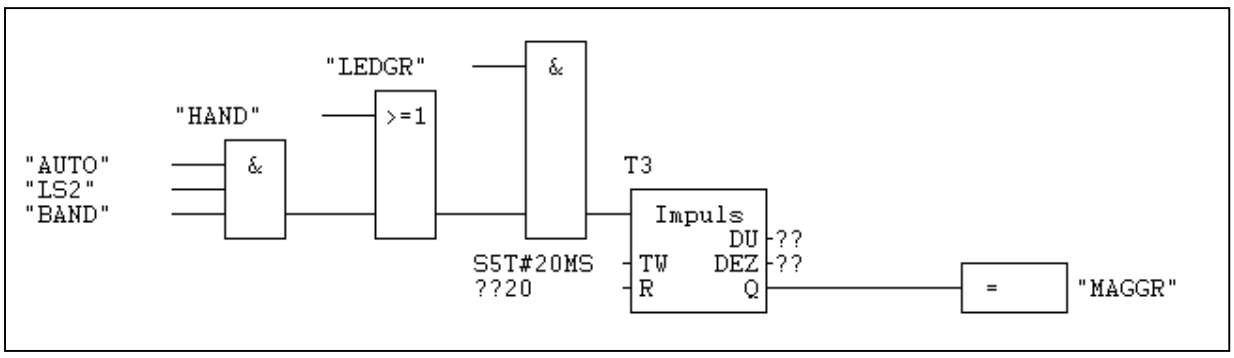
Anmerkung zu „FUNC 100“: Erläuterungen zu dieser Funktion erhalten Sie unter TrySim-Hilfe, <Index>| „FUNC“.

OB 1; Netzwerk 4: Magazin-Ansteuerung (BLAU) (Einzelfreigabe durch Zeitbegrenzung)

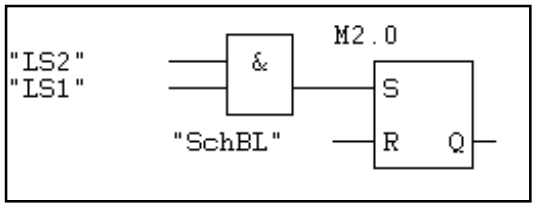


Die Dynamiks / „Pakete“ fallen aus dem Generator / „Magazin“, wenn am Generator-Operanden eine „1“ als Impuls anliegt. Bei Dauer-„1“ würden ständig neue Pakete mit der eingestellten Verzögerung (siehe Editierfenster des Generators) entstehen.

OB 1; Netzwerk 5: Magazin-Ansteuerung (GRÜN) (Einzelfreigabe durch Zeitbegrenzung)

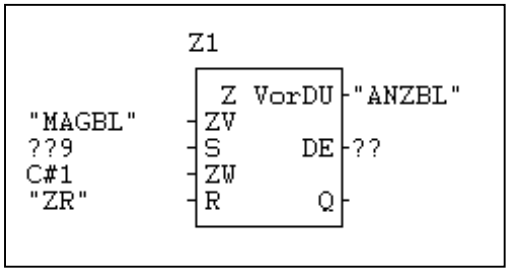


OB 1; Netzwerk 6: Erfassung der langen Teile durch beide Lichtschranken



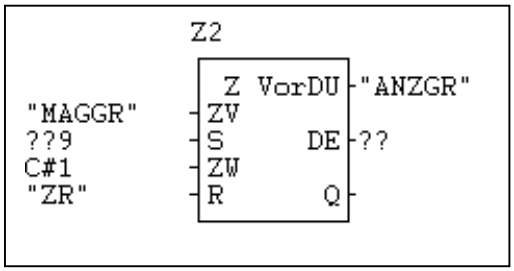
Nur wenn beide Lichtschranken eine „1“ liefern, d.h., gleichzeitig abgedeckt sind, wird Merker M 2.0 gesetzt und hält als Gedächtnis fest, dass sich ein langes Teil der ersten Weiche nähert. Nachdem diese Position erreicht wurde, wird der Merker zurückgesetzt.

OB 1; Netzwerk 7: Zähler für die langen (blauen) Teile

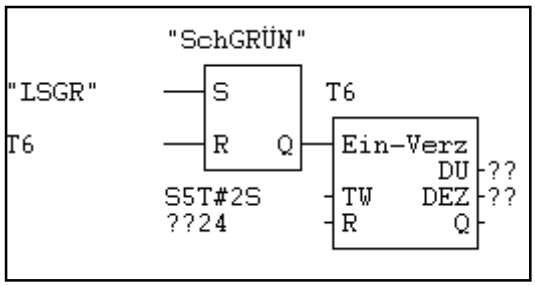


Wenn das Magazin durch einen Impuls ein Paket freigibt, zählt der Zähler um eine Einheit rauf.
Mit <Rücksetzen> können die Zähler auf „0“ zurückgesetzt werden.
Die Eingänge „S“ und „ZW“ werden hier nicht benötigt.

OB 1; Netzwerk 8: Zähler für die kurzen (grünen) Teile

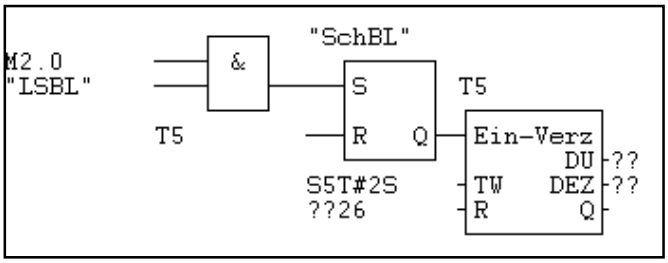


OB 1; Netzwerk 9: Paketschieber GRÜN (zeitgesteuert)



Wenn die Lichtschranke „LSGR“ „1“ führt, d.h., abgedeckt ist, wird das grüne / kurze Paket mit einem 4/2-Wegeventil (Feder) zur Seite geschoben. Es muss keine Auswahl berücksichtigt werden, da an dieser Station „nur der Rest“ ankommt, der zu diesem Lager gehört.
Mit einer kurzen „1“ von T6 wird das Ventil-FlipFlop zurück gesetzt.

OB 1; Netzwerk 10: Paketschieber BLAU (zeitgesteuert)

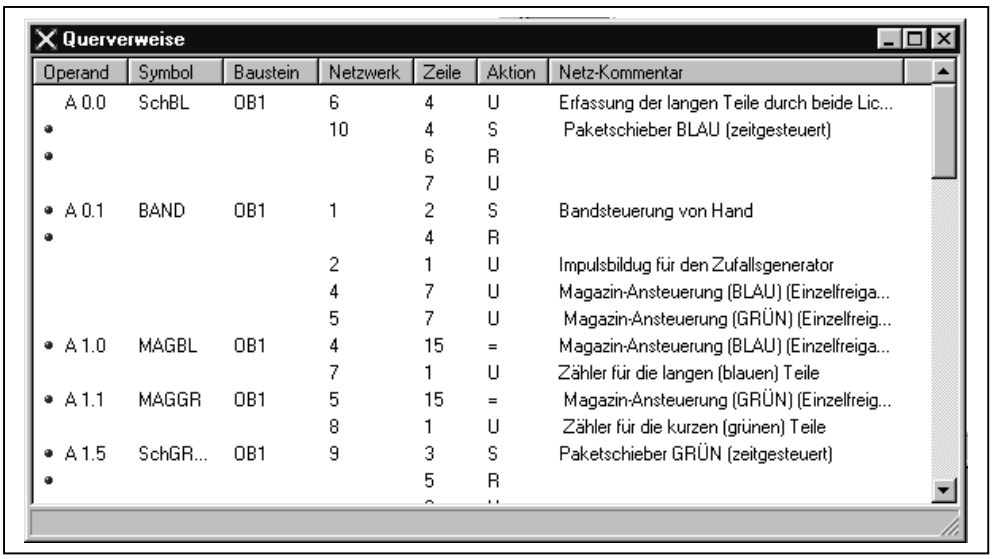


Wenn der Merker <M2.0> festgehalten hat, dass ein langes Paket vermessen wurde und dieses Paket die Lichtschranke <LSBL> erreicht hat, sorgt <SchBL> für den Auswurf.
<SchBL> setzt jetzt auch M 2.0 im Netzwerk 6 zurück.
Mit einer kurzen „1“ von T5 wird das Ventil-FlipFlop zurück gesetzt.

Vgl. TrySim-Projekt <SortierStrecke_1> im Verzeichnis <Lösungen>.

Hinweis:

In dieser Anleitung wurde die Symboltabelle nicht vorbildlich geführt. Es wurden zwar alle Ein- und Ausgänge aufgelistet, aber nicht alle sonst noch beteiligten Operanden, bzw. Elemente.
Für eine umfassende Dokumentation wäre es aber erforderlich, z.B. auch alle Timer aufzuführen, um bei späteren Änderungen nicht aus Unkenntnis einen bereits verwendeten Timer nochmals – für andere Programmteile – zu verwenden.
Es sollte Ihre Aufgabe sein, die Liste zu vervollständigen. Einen Überblick über alle verwendeten Operanden bieten die **<Querverweise>**, die Sie unter <SPS>|<Querverweise> finden.



Operand	Symbol	Baustein	Netzwerk	Zeile	Aktion	Netz-Kommentar
A 0.0	SchBL	OB1	6	4	U	Erfassung der langen Teile durch beide Lic...
•			10	4	S	Paketschieber BLAU (zeitgesteuert)
•				6	R	
				7	U	
• A 0.1	BAND	OB1	1	2	S	Bandsteuerung von Hand
•				4	R	
			2	1	U	Impulsbildug für den Zufallsgenerator
			4	7	U	Magazin-Ansteuerung (BLAU) (Einzelfreiga...
			5	7	U	Magazin-Ansteuerung (GRÜN) (Einzelfreig...
• A 1.0	MAGBL	OB1	4	15	=	Magazin-Ansteuerung (BLAU) (Einzelfreiga...
			7	1	U	Zähler für die langen (blauen) Teile
• A 1.1	MAGGR	OB1	5	15	=	Magazin-Ansteuerung (GRÜN) (Einzelfreig...
			8	1	U	Zähler für die kurzen (grünen) Teile
• A 1.5	SchGR...	OB1	9	3	S	Paketschieber GRÜN (zeitgesteuert)
•				5	R	

Hier sind auch alle Netzwerke und Aktionen aufgelistet, die bei einer Fehlersuche hilfreich sein können. Z.B. könnten Sie hier erkennen, wenn ein Ausgangsoperand fälschlicherweise in mehreren Netzwerken gesetzt wäre. Jedem Ausgang darf nur einmal ein „=“ zugewiesen sein. Auch eine „Kombination“ von „=“ und „S“ wäre falsch.

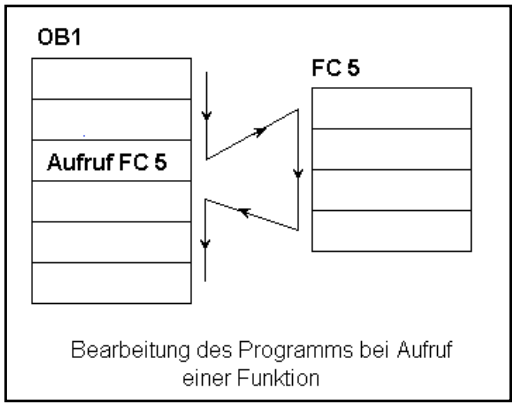
Projekt 27: Funktion FC

Schwerpunkte: Aufruf einer Funktion FC, Addition, Zählerbaustein, Multiplikation, Wortdarstellung

Vgl. TrySim-Projekt <FC> im Verzeichnis <Lösungen>.

Aufgabe:
An einem Zählerbaustein soll die Zahl 10 mit Tastendruck eingestellt und mittels Digitalanzeige dargestellt werden. Mit 2 weiteren Tasten ist dieser Zahlenwert um +/- 1 zu verändern. (Es sind nur positive Zahlen am Zähler einzustellen). Diese Zahl ist mit 30 zu addieren. Das Ergebnis kann mit einer Schalterbetätigung mit 2 multipliziert und ebenfalls angezeigt werden. Diese Aufgabe soll nicht nur im OB1, sondern in 2 Funktionen bearbeitet werden.

Theorie:
Diese kleine Zählaufgabe demonstriert die Möglichkeit, ein Unterprogramm aufzurufen.
In Funktionen formulieren Sie Teil-Aufgaben des Programms. Wenn Sie im OB 1 oder einem anderen Baustein eine Funktion aufrufen, werden die Anweisungen der Funktion bearbeitet. Am Ende der Funktion wird die Bearbeitung im aufrufenden Baustein fortgesetzt.

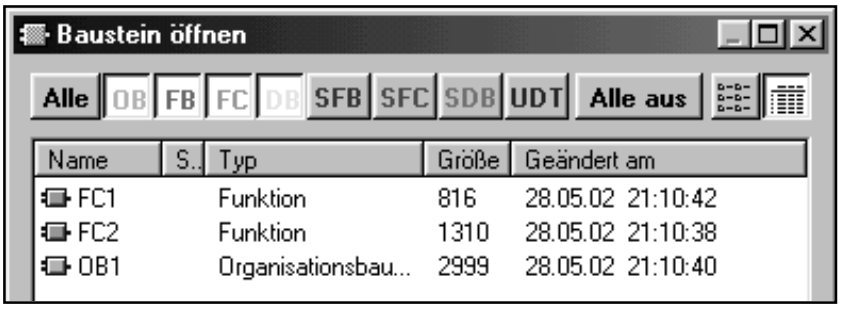


Bearbeitung des Programms bei Aufruf einer Funktion

Funktionen können auch geschachtelt aufgerufen werden, d.h. Sie können innerhalb einer Funktion eine weitere aufrufen. Funktionen, die häufig wiederkehrende Aufgabe zu erfüllen haben, können auch mehrfach aufgerufen werden.
(vgl. und vertiefe unter TrySim <Hilfe> <Index> "Aktualparameter" oder "Funktionen").

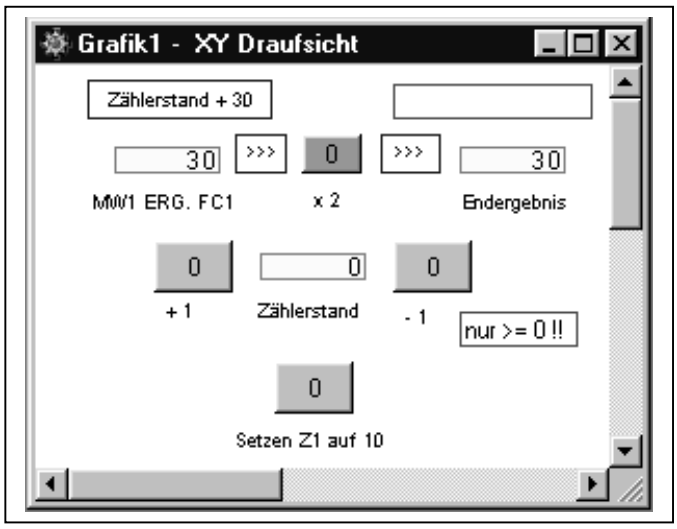
Beachten Sie bitte, dass Sie den Begriff "Funktion" nicht mit dem später folgenden Begriff "Funktionsbaustein" verwechseln.

So sieht später die fertige Bausteinübersicht aus.



Das Projekt ist vorbereitet unter <Vorlagen> <FC_1_V>. Sie können diese Vorlage öffnen und dann in Ihr Arbeitsverzeichnis kopieren unter <Projekt> <Speichern unter...>.

Die Steuereinheit und Anzeige sind bereits erstellt:

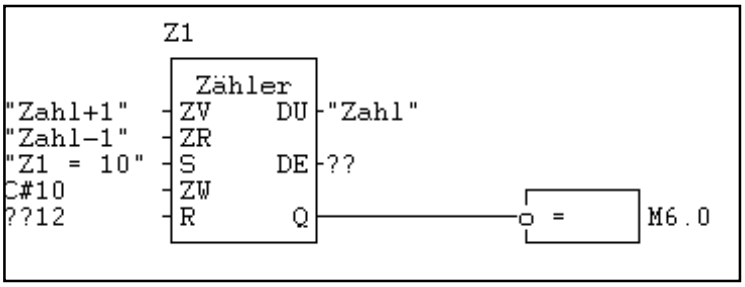


Bereits erstellte Symboltabelle:

Symbolische Adresse	Operand	Datentyp	Kommentar
Zahl	AW 512	INT	Zählerstand
Zahl+1	E 0.0	BOOL	Zähle dazu
x 2	E 0.1	BOOL	Multiplikator
Zahl-1	E 0.3	BOOL	Zähle ab
Z1 = 10	E 0.6	BOOL	Zählerstand = 10
Ergebnis_FC	MW 1	INT	AUSGANG FC1
ENDERG	MW 4	INT	Endergebnis aus OB 1

Bisher wurden alle Netzwerke ohne Einschränkungen oder Nachteile im OB 1 programmiert. Das war auch richtig bzw. ausreichend. Bei komplexeren Programmen oder bei wiederkehrenden Aufgaben ist es aber sinnvoll, Unterprogramme anzulegen. Dadurch wird die Lesbarkeit erhöht und bei Wiederholungen von Programmteilen wird das Programm einfacher.
Das folgende Programm soll nur das Prinzip des Funktionsaufrufs verdeutlichen und ist sicherlich kein klassischer Anwendungsfall für die Verwendung von Funktionen.
Das Programm startet immer automatisch im OB 1. Dort kann - wie bisher - mit beliebigen Anweisungen in einem Netzwerk begonnen werden.

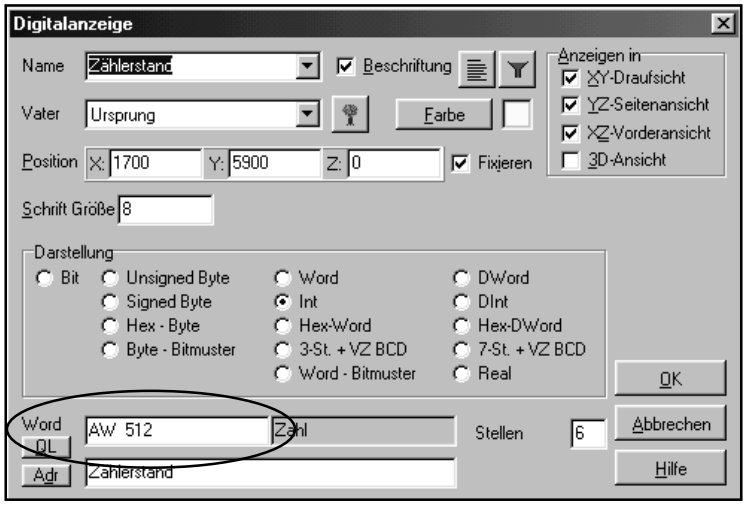
OB 1; Netzwerk 1: Der OB 1 wird automatisch aufgerufen. Hier startet das Programm.



Der Ihnen bekannte Zählerbaustein kann über Tasterbefehle unterschiedliche Zahlen am Ausgang <DU> darstellen. Die symbolische Adresse "Zahl" steht (siehe Symboltabelle) für das Ausgangswort

"AW 512". Dieses Wort kann mit der Digitalanzeige "Zählerstand" verknüpft werden, indem z.B. bei ausgeschalteter Anlage mit <rM> das Editierfenster der Anzeige geöffnet wird.

Editierfenster der Digitalanzeige "Zählerstand"



Die andere Methode, einen Operanden zuzuweisen, kennen Sie schon:
Bei ausgeschalteter Anlage mit <LM> das gewünschte Element anklicken und im Netzwerk mit <LM> an den entsprechenden Ein- bzw. Ausgang klicken.

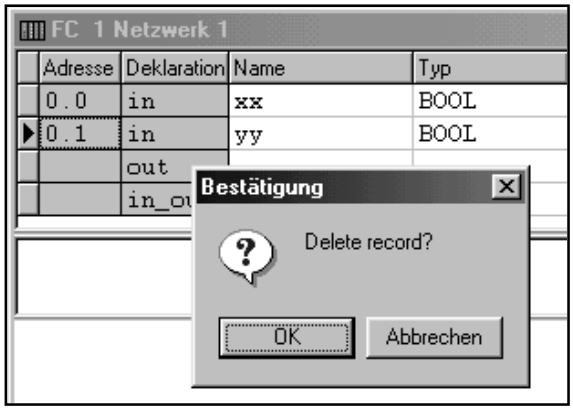
Zurück zum Netzwerk 1: Am Eingang <ZW> wird nach "C#" die Vorwahlzahl eingetragen, auf die der Zähler nach Betätigung von <E 0.6> (“Z1 = 10“) springen soll.
"C#..." bedeutet, die folgende Dezimalzahl wird intern vom Programm BCD-codiert. Diese Codierung ist intern für den Baustein erforderlich. Konstanten am Eingang ZW eines Zählers sollten immer in dieser Form angegeben werden.

Jetzt beginnt das Beispiel für den Aufruf des 1. Unterprogramms. Bevor man ein Unterprogramm aufruft, muss es bereits existieren. Zumindest muss der Programmkopf mit den deklarierten Parametern angelegt worden sein.
(vgl. und vertiefe unter TrySim <Hilfe>|<Index> "Funktionen" und dort "Parameter" etc.).

Der neue Baustein wird erstellt unter <SPS>|<NEU> und dort <FC> wählen.

Alle Werte, die an den Ein- oder Ausgängen dieser Funktion anliegen sollen, sind im Kopf der Funktion zu "deklarieren". Dabei ist zwischen Ein- und Ausgängen zu unterscheiden. Eingänge werden unter der Deklaration "in" mit frei zu wählendem Namen und dem Datentyp eingetragen. Die Zeilen für "in" vermehren sich, wenn der Cursor im Feld "Kommentar" steht und <ENTER> gedrückt wird. Sollten aus Versehen eine "in"_Zeile zu viel entstanden sein, brauchen Sie den Cursor nur in eine andere Zeile setzen und die überzählige Zeile verschwindet. Wenn Sie eine bereits deklarierte Zeile löschen wollen, markieren Sie diese Zeile im Bereich <Adresse> (jedenfalls nicht im Feld <Name>) und wählen <Entf> und in unten abgebildetem Fenster <OK>.

Wichtig: Die <Namen> dürfen keine Umlaute oder Lücken haben. Außer Buchstaben und Zahlen ist nur der Unterstrich erlaubt.



Hier würde die 2. Zeile mit der Adresse 0.1 gelöscht werden.

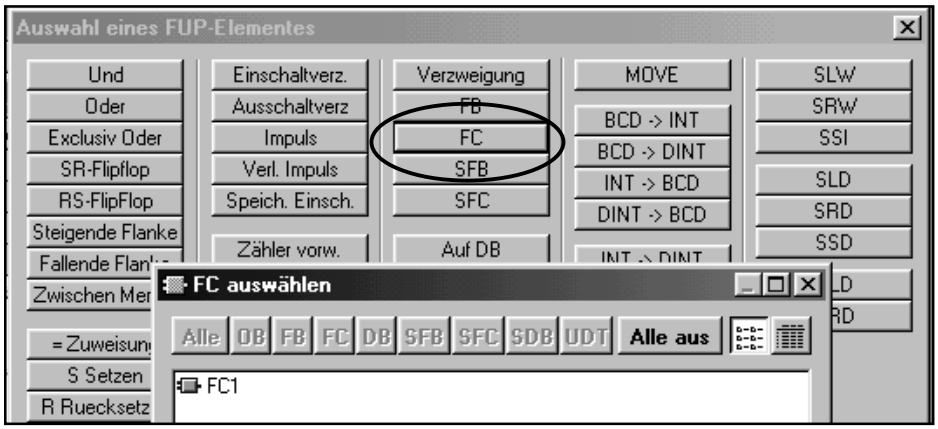
Die <Namen> können sie frei wählen. Sie sollten immer mit einem Kommentar versehen werden.
Die Eingangsgrößen sind hier Zahlen. Deshalb wird als Datentyp "INT" gewählt.
(vgl. TrySim <Hilfe>|<Index> "INT").
Das Ergebnis wird mit einem anderen Deklarationstypen ("out") an das "außenstehende" Programm ausgegeben. Auch hier ist als <Typ> "Int" zu wählen.
Mehr ist erst einmal für den Aufruf der FC nicht vorzubereiten.

Deklaration FC 1

FC 1 Netzwerk 1 Der Zählerwert AW 512 wird mit dem Wert 30 (aus dem OB1) addiert.

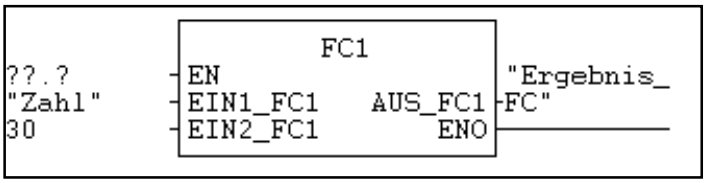
Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	EIN1_FC1	INT		Eingangsabfrage 1 (Formalparameter)
2.0	in	EIN2_FC1	INT		Eingangsabfrage 2 (Formalparamter)
4.0	out	AUS_FC1	INT		Ausgangswert
	in_out				

Beachten Sie einen wichtigen Schritt: Bevor Sie die FC aufrufen können, müssen Sie jetzt mit <Alles speichern> die bisherige Programmierung aktualisieren, sonst sehen Sie die neue FC nicht in der Auswahl.
Danach kann im OB 1 die Funktion aufgerufen werden. Dazu wird im OB 1 das neue Netzwerk 2 aktiviert und aus dem Icon "FUP-Elemente" <FC 1> angeklickt.



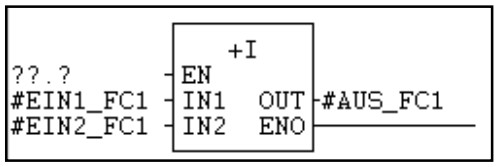
OB 1; Netzwerk 2: Das Hauptprogramm im OB1 ruft ohne Bedingungen die Funktion (FC) FC1 auf.

Das Ergebnis ist:



Die dargestellten Parameter <EN> und <ENO> entstehen automatisch und sind hier nicht von Bedeutung. Sie sehen, dass für alle im FC 1-Kopf deklarierten Parameter "Anschlüsse" vorhanden sind. Diese so genannten "Formalparameter" sind mit den äußeren "Aktualparametern" zu belegen. Laut Aufgabe sollen 2 Eingangswerte addiert werden.
"Zahl" steht - wie oben erläutert - für das AW 512, dem Wert (der Zahl) am Zählerbaustein. Der eine Formalparameter <EIN1_FC1> wird mit "Zahl" "verbunden". An den zweiten Formalparameter wird ohne Bezug zu einem Wort etc. direkt die Dezimalzahl "30" geschrieben. Damit hat man die äußeren Voraussetzungen geschaffen, dass diese beiden Zahlen innerhalb der Funktion verarbeitet werden können. Das eigentliche Programm dafür fehlt natürlich noch und soll jetzt folgen.
In der FC 1 können wie im OB 1 eigene Netzwerke programmiert werden, die nur innerhalb dieser Funktion gelten. Bei aktivem FC1 / Netzwerk1-Editierfenster wird aus dem Icon <FUP-Elemente> der Button <+ Int> ausgewählt. Dieser Baustein erlaubt die Addition der beiden Eingangswerte an <IN1> und <IN2>. Da wir uns innerhalb der FC 1 bewegen und die äußeren Werte (Aktualparameter) bereits an die Formalparameter der FC übergeben wurden, muss man jetzt nur noch die Formalparameter (die innen anliegenden Werte) an die Eingänge legen. Nach dem Schreiben von "EIN1_FC1" setzt das Programm automatisch ein "#" vor den Namen. Daran erkennt man, dass hier ein Formalparameter aufgerufen wurde. Es wäre auch möglich, an <IN2> direkt die Zahl "30" zu schreiben oder "Zahl" an Eingang <IN1> zu legen. Der Vorteil bei der Verwendung von Formalparametern ist - wie Sie später sehen werden - dass derselbe Baustein mit verschiedenen äußeren Daten arbeiten kann.

FC 1; Netzwerk 1: Der Zählerwert AW 512 wird mit dem Wert 30 (aus dem OB1) addiert. (vgl. OB1, Netzwerk 2).



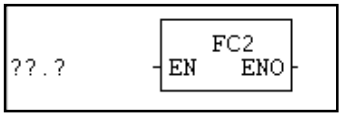
Auch beim Ausgang "sieht" dieser Baustein nicht über Grenzen der FC hinaus. Der Übergang zur Außenwelt findet erst durch das Netzwerk OB 1; Netzwerk 2 statt, in dem entschieden wird, an wen das Ergebnis an "#AUS_FC1" weiter geleitet wird. Dieses Ergebnis wird im Merkerwort MW 1 („Ergebnis_FC“) in der Digitalanzeige dargestellt.
Das Programm in der FC 1 ist jetzt beendet. Damit auch der Sprung vom OB 1 zur FC 1. Es ist kein "Ende"-Befehl erforderlich - das Betriebssystem der SPS setzt die Bearbeitung automatisch nach der Absprungstelle fort. Wir sind also wieder im OB 1.
Dieses MW 1 soll noch weiter verarbeitet werden. Es soll bei betätigtem Schalter < x2> mit 2 multipliziert werden. Diese Aufgabe könnte im OB 1 erledigt werden. Viele Anwender sind so geschult worden, dass sie der Meinung sind, alle Programme sollten in Unterprogrammen laufen und der OB 1 "verteilt" lediglich die Verzweigungen dorthin. Hierfür gibt es keine zwingende Notwendigkeit. Die Anwendung hängt von der Aufgabenstellung ab. Dennoch soll hier dieser erneute FC-Aufruf dargestellt werden.
Die Vorgehensweise ist genau so, wie bei der FC 1. Zuerst ist eine neue FC zu erzeugen mit <SPS>|<NEU>|<FC>. Sie sehen in der Maske, dass automatisch die nächste FC (2) vorgeschlagen wird. Hier sollen zur Abwechslung einmal gar keine Formalparameter erstellt werden. Dennoch ist es erforderlich, diesen Baustein anzulegen und das Programm erneut zu speichern.

Deklaration FC 2

FC 2 Netzwerk 1

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
▶		in				
		out				
		in_out				
		temp				

OB 1; Netzwerk 3: Erneuter Aufruf einer anderen FC aus dem OB1



Wie Sie sehen, sind keine Anschlüsse für Ein- und Ausgänge vorhanden (außer für die hier nicht verwendeten <EN> und <ENO>).
In AWL wird der Sinn dieses Netzwerkes etwas deutlicher: Hier wird nur die FC 2 aufgerufen, d.h., das Programm arbeitet in dieser Funktion weiter.
Die Befehlszeile lautet: "CALL FC 2 " //CALL ist der Aufruf-Befehl.

Nicht alle Befehlskombinationen lassen sich in FUP oder KOP in einem Netzwerk darstellen. Es kann aber ganz hilfreich sein, wenn Zusammenhänge in einem nicht zu langem Programm in einem Netzwerk dargestellt werden. Deshalb wird hier einmal in AWL programmiert. AWL erlaubt die Darstellung mehrerer Anweisungen in einem Netzwerk. Ein solches Netzwerk ist allerdings nicht mit Klick in FUP oder KOP zu übersetzen.

FC 2; Netzwerk 1: Multiplikation und Anzeige

```
      U  "x 2"              // Wenn der Schalter eingeschaltet ist... (s. Symboltabelle)
      SPBN  ende            // (falls nicht, springe zur Marke <ende:>)
      L   "Ergebnis_FC"     // Wenn die Bedingung erfüllt ist, lade das MW 1 (Ergebnis von FC 1)
      L   2                 // Lade den Wert 2,
      *I                    / multipliziere die Faktoren
      T   "ENDERG"          // und transferiere das Ergebnis zum MW 4 (Anzeige Endergebnis)
      BEA                   // absolutes Bausteinende, d.h., folgende Anweisung wird nicht mehr
                            // ausgeführt.
ende: L   "Ergebnis_FC"     // Falls der Schalter oben nicht eingeschaltet war, lade MW 1
      T   "ENDERG"          // und transferiere den Wert zum MW4
```

Der Lade-Befehl "L ..." "kümmert" sich nicht um die vorausgegangene Verknüpfungsprogrammierung. Wenn Sie einen Lade-Befehl also nur unter Bedingungen ausgeführt haben wollen, müssen Sie wie hier eine Sprungbedingung einfügen.
Ohne den bedingten Sprung zur Marke <ende:> würde <Ergebnis_FC> immer mit 2 multipliziert werden. Wenn Sie den Multiplikator ausschalten, muss sich auch der Wert <ENDERG> ändern. Deshalb muss unter <ende:> das aktuelle <Ergebnis_FC> geladen und nach <ENDERG> transferiert werden. Andererseits darf das vorstehende Programm (Multiplikation) nicht bis <ende:> durchlaufen, wenn sich <ENDERG> geändert hat. Mit "BEA" wird das Programm in diesem Netzwerk beendet.

Achten Sie auch darauf, dass nicht irgendwelche Operanden durch eine Sprungbedingung gesetzt und durch einen anderen Sprung nicht wieder aufgerufen werden! Das Programm merkt sich bis zu einer neuen Abfrage den einmal zugewiesenen Wert und reagiert dementsprechend - oder eben auch nicht mehr! Gelegentlich wundern sich unerfahrene Programmierer, dass ein Ausgang einfach nicht mehr abschaltet. Erst nach dem Ausschalten des Projektes und nach einem Neustart ist der "klebende" Ausgangsoperand doch auf "0" gegangen. In solchem Fall wurde wahrscheinlich ein Netzwerk mit dem mysteriösen – aktiven (!) - Ausgang gelöscht, so dass er nie wieder aufgerufen werden konnte. Dennoch bleibt der Operand mangels erneuter Zuweisung treu gesetzt!

Projekt 28: Wiederholter Aufruf einer Funktion (fehlerbehaftet)

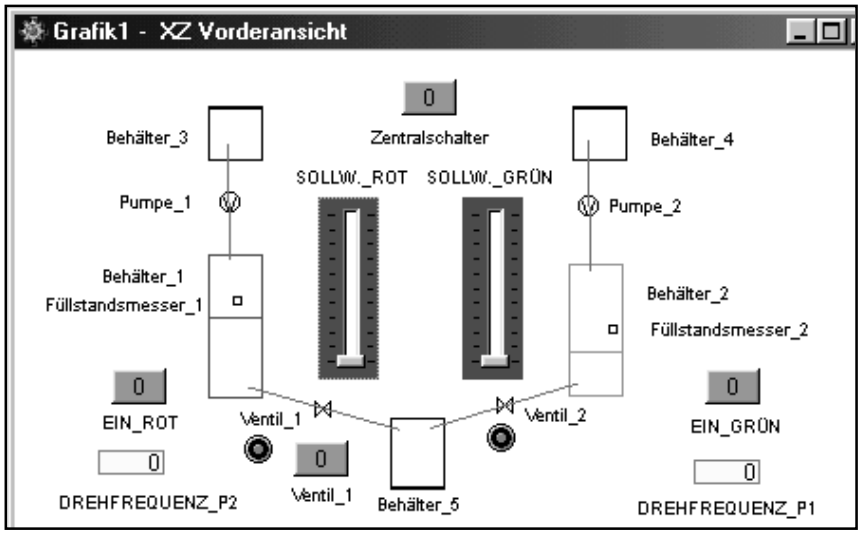
Schwerpunkte: Sprung SPB,SPBN; CALL FC, <I, -I

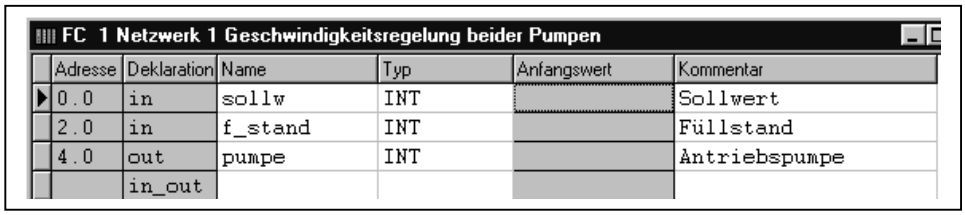
Aufgabe: Übung mit FC

2 Behälter können unabhängig von einander bis zu einer Sollwertvorgabe gefüllt werden. Die Pumpen-Drehfrequenz ist proportional zur Abweichung vom Sollwert zu regeln. Für beide Regelstrecken ist nur ein Programm zu erstellen, das nacheinander beide Füllstände unabhängig voneinander regelt. Die Füllstände werden mit dem Element <Füllstandsmesser> erfasst (vgl. TrySim-Hilfe <Index>| „Füllstandssensor“).
Der Clou der Aufgabe liegt darin, dass beide Behälter ständig entleert werden können. Es ist also nicht der Füllstand allein, sondern die Durchflussmenge über die Drehfrequenz zu regeln.
Mit dem Zentralschalter wird die Anlage freigegeben, gleichzeitig wird Ablauf-Ventil 2 geöffnet. Der Ablauf über Ventil 1 wird zusätzlich von Hand freigegeben. Jeder Behälter hat noch eine zusätzliche Freigabe.
Diese Freigabekombination wurde gewählt, um die Auswirkungen von falsch gesetzten Sprüngen, bzw. von "abgeschalteten" Ausgängen zu beobachten. Dieses Problem wurde im vorstehenden Projekt bereits angesprochen und soll hier durch eine Anwendung verdeutlicht werden

Dieses Projekt ist bereits vorbereitet im Ordner <Vorlagen> unter <FC_2Fehler_V>.

Symbolische Adresse	Operand	Datentyp	Kommentar
PUMPE_1	AW 512	INT	Antrieb Pumpe 1
PUMPE_2	AW 516	INT	Antrieb Pumpe 2
Ventil_1	A 0.2	BOOL	Ventil 1
Ventil_2	A 0.3	BOOL	Ventil 2
SW_ROT	EW 512	INT	Sollwert ROT
SW_GRUEN	EW 514	INT	Sollwert GRÜN
ZENTRAL	E 0.0	BOOL	Zentralschalter
FS_ROT	EW 516	INT	Füllstand ROT
FS_GRUEN	EW 518	INT	Füllstand Grün
EIN_GRN	E 0.1	BOOL	EIN-Schalter GRÜN
EIN_ROT	E 0.2	BOOL	EIN-Schalter ROT
EIN_VENT	E 0.3	BOOL	EIN-Schalter Ventil 1
LED_V1	A 0.4	BOOL	LED Ventil 1
LED_V2	A 0.5	BOOL	LED Ventil 2





FC 1 Netzwerk 1 Geschwindigkeitsregelung beider Pumpen

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
▶	0.0	in	sollw	INT		Sollwert
	2.0	in	f_stand	INT		Füllstand
	4.0	out	pumpe	INT		Antriebspumpe
		in_out				

OB1, Netzwerk 1: START u. Aufruf der Funktion ...und das Chaos nimmt seinen Lauf...

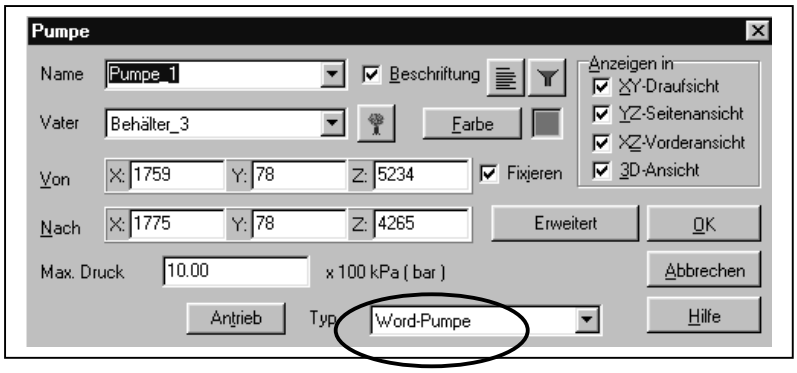
```
    U  "ZENTRAL"          // Wenn der <Zentralschalter> eingeschaltet wird...
    SPBN  ende            // (Falls nicht, springe zur Sprungmarke <ende:>
    =  "Ventil_2"         // ... schaltet das Ventil 2 ein (Ablauf Behälter 2) und…
    =  "LED_V2"           // …die zugehörige LED.
    U  "EIN_VENT"         // Wenn Schalter <Ventil_1> eingeschaltet wird...
    =  "Ventil_1"         // … schaltet das Ventil 1 ein (Ablauf Behälter 1) und…
    =  "LED_V1"           // … die zugehörige LED.

    UN "EIN_ROT"          // Wenn <EIN_ROT> nicht betätigt wird...
    SPB   __2             // ... springe zur Marke <__2:>.

     CALL FC 1            // Funktionsaufruf der FC 1
                          // Es folgt die Zuweisung der Aktualparameter
     sollw   := "SW_ROT"  // Am Eingang <sollw> liegt das EW 512 (“SW_ROT”)
     f_stand := "FS_ROT"  // Am Eingang <f_stand> liegt das EW 516 („FS_ROT“)
     pumpe   := "PUMPE_1" // Am Ausgang <pumpe> liegt das AW 512 („PUMPE_1“)

__2:  UN "EIN_GRN"        // Wenn <EIN_GRN> nicht betätigt ist...
      SPB   ende          // ... wird der Rest bis zur Marke <ende:> übersprungen.

     CALL FC 1            // wiederholter Funktionsaufruf der FC 1 mit anderen
                          // Aktualparametern
     sollw   := "SW_GRUEN"  //Am Eingang <sollw> liegt das EW 514 (“SW_GRUEN”)
     f_stand := "FS_GRUEN"  // Am Eingang <f_stand> liegt das EW 518 („FS_GRUEN“)
     pumpe   := "PUMPE_2"   // Am Ausgang <pumpe> liegt das AW 516 („PUMPE_2“)

                          // Die gleiche Funktion wird also nacheinander mit unterschiedlichen
                          // Aktualparametern verwendet.
ende: NOP 0               // Sprungziel ohne weitere Anweisungen.
```

FC1, Netzwerk 1: Geschwindigkeitsregelung beider Pumpen

```
      L   #sollw          // Lade den Formalparameter der Funktion mit den augenblicklichen
                          // Aktualwerten (entweder EW 512 oder EW 514).
      L   #f_stand        // Lade den Formalparameter aktueller Füllstand
      <I                  // Prüfe, ob 1. Wert kleiner als 2. Wert ist.
      SPB   null          // Wenn ja, springe zur Marke <Null:>. Dann soll die Pumpe nicht fördern.
       L   #sollw         // Sonst ...
       L   #f_stand       //       … bilde die …
      -I                  //                  ... Differenz ...
      SPA end             // ..und springe – ohne weitere Bedingungen – zur Marke <end:>.
Null:  L 0                // Lade den Wert 0
end:  T   #pumpe          // Transferiere den aktuellen Wert an den Formalparameter #pumpe.
```

Wieso wird mit diesem kleinen Programm die Geschwindigkeit zweier Motoren geregelt?

Im OB1 wird nacheinander zweimal die FC 1 aufgerufen. Dort werden die Eingangsparameter (Soll- und Istwert) für beide Anlagenteile verglichen und der jeweilige Pumpenantrieb wird mit der entsprechenden Stellgröße beaufschlagt.
Als Antrieb wurde eine <Word-Pumpe> im Editierfenster ausgewählt. Die Geschwindigkeit ändert sich also proportional zur Differenz zwischen Soll- und Istwert. Wenn der Sollwert kleiner als der Istwert ist, soll die Pumpe nicht rückwärts laufen, sondern gar nicht fördern. Diese Bedingungen regelt die FC 1.



vgl. TrySim-Projekt <Lösungen>|<FC_2Fehler>

Das ist das gesamte Programm. So weit - so gut. Sehen Sie es sich bitte in Ruhe an, bevor Sie es testen. Versuchen Sie, logische Brüche zu erkennen. Testen Sie es dann ein wenig und lehnen Sie sich stolz zurück, wenn die "Regelung" bei geschlossenen <EIN>-Schaltern funktioniert. Für die rote Anlage kann auch das Ablaufventil geöffnet bzw. geschlossen werden. Was passiert aber, wenn ein Sollwertsteller auf einen höheren Wert (nicht unbedingt max.) gestellt wird und dann der Zentralschalter schnell ausgeschaltet wird?
Hilflos werden Sie zusehen müssen, wie der Kessel überläuft und verzweifelt den NOT-AUS oder die Hauptsicherungen suchen.
Sie meinen vielleicht, das käme nicht vor? Es reicht aber nicht aus, nur die gewünschte Funktion zu erfüllen, sondern es ist Ihre Pflicht darüber nachzudenken, was alles falsch gemacht werden kann oder welche technischen Fehler passieren können. In der Simulation ist der Schaden, den Sie mit dem Programm anrichten, ja nur symbolischer Natur. In der Praxis dürfte Ihnen bei gefährlichen Flüssigkeiten ein solcher programmierter Überlauf sicherlich den Job oder weitere Aufträge kosten. Das vor dem Ernstfall zu finden, darin liegt der überaus große Wert dieses Simulationsprogrammes.

Wie ist dieses gefährliche Phänomen zu erklären?
Wenn der <Zentralschalter> nicht betätigt ist, springt im OB1 das Programm sofort ans Ende zur Sprungmarke <ende:>. Der Aufruf der FC 1 wird also übersprungen. Der letzte Wert, den die FC 1 ausgegeben hat, bleibt unerbittlich erhalten und treibt die Pumpe an.

Das Programm muss also unter allen Umständen die Ausgangswerte auf „0“ setzen können.

Besser – und richtig - geht´s mit folgender Version:

Im OB 1 finden keine Sprünge mehr statt; das Programm läuft immer alle „Stationen“ durch.

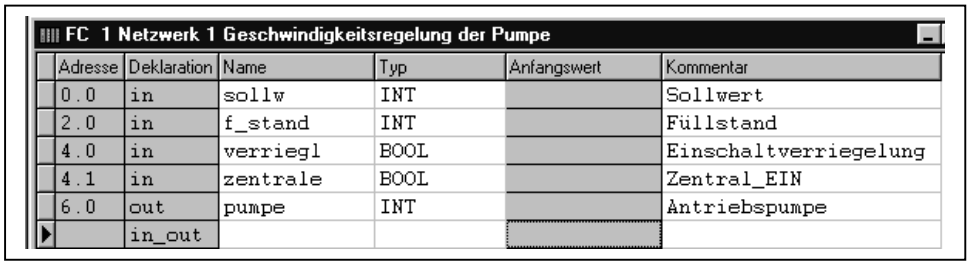
Außerdem werden in der FC 1 weitere Formalparameter deklariert.

vgl. TrySim-Projekt <Lösungen>|<FC_3>

Projekt 29: Wiederholter Aufruf einer Funktion (richtige Version)

(Fortsetzung von Projekt 28) vgl. TrySim-Projekt <Lösungen>|<FC_3>

Deklaration FC 1



FC 1 Netzwerk 1 Geschwindigkeitsregelung der Pumpe

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
	0.0	in	sollw	INT		Sollwert
	2.0	in	f_stand	INT		Füllstand
	4.0	in	verriegl	BOOL		Einschaltverriegelung
	4.1	in	zentrale	BOOL		Zentral_EIN
	6.0	out	pumpe	INT		Antriebspumpe
▶		in_out				

Die Programmsprünge werden in die FC verlagert. Dadurch wird verhindert, dass die variablen Ausgangswerte „eingefroren“ werden, falls die FC gar nicht mehr angesteuert würde.

OB1, Netzwerk 1: START u. Aufruf der Funktion (...richtige Version...)

```
U  "ZENTRAL"          // Wenn der <Zentralschalter> eingeschaltet wurde ...
=  "Ventil_2"         // ... wird <Ventil_2> geöffnet....
=  "LED_V2"           // ... und die LED eingeschaltet.
U  "ZENTRAL"          // Wenn der <Zentralschalter> eingeschaltet wurde ...
U  "EIN_VENT"         // ... und der Schalter <Ventil_1> geschlossen wurde...
=  "Ventil_1"         // … wird das Ventil <Ventil_1> geöffnet und…
=  "LED_V1"           // … die LED eingeschaltet.
CALL FC 1             // Funktionsaufruf FC 1
  sollw    := "SW_ROT"
  f_stand  := "FS_ROT"
  verriegl := "EIN_ROT"     // Abfrage, ob der Schalter <EIN_ROT> eingeschaltet wurde
  zentrale := "ZENTRAL"     // Abfrage, ob der Schalter <Zentralschalter> eingeschaltet wurde
  pumpe    := "PUMPE_1"
CALL FC 1
  sollw    := "SW_GRUEN"
  f_stand  := "FS_GRUEN"
  verriegl := "EIN_GRN"
  zentrale := "ZENTRAL"
  pumpe    := "PUMPE_2"
```

FC 1 wird hier also immer aufgerufen – egal, wie die Schaltbedingungen lauten.

FC1, Netzwerk 1: Geschwindigkeitsregelung beider Pumpen

```
    ON #verriegl      // Wenn der entsprechende Behälterschalter…
    ON #zentrale      // … oder der <Zentralschalter> nicht eingeschalten wurden…
    SPB   null        // …springt das Programm zur Marke <null:>.
    L   #sollw        // (hier ändert sich nichts)
    L   #f_stand
    <I
    SPB   null
    L   #sollw
    L   #f_stand
    -I
    SPA   pump
null: L   0           // Wenn obige Schalter nicht eingeschaltet wurden, bleiben die
                      // Pumpen stehen (Wert = 0 wird transferiert).
pump: T   #pumpe      // obige Werte werden an die Pumpen transferiert.
```

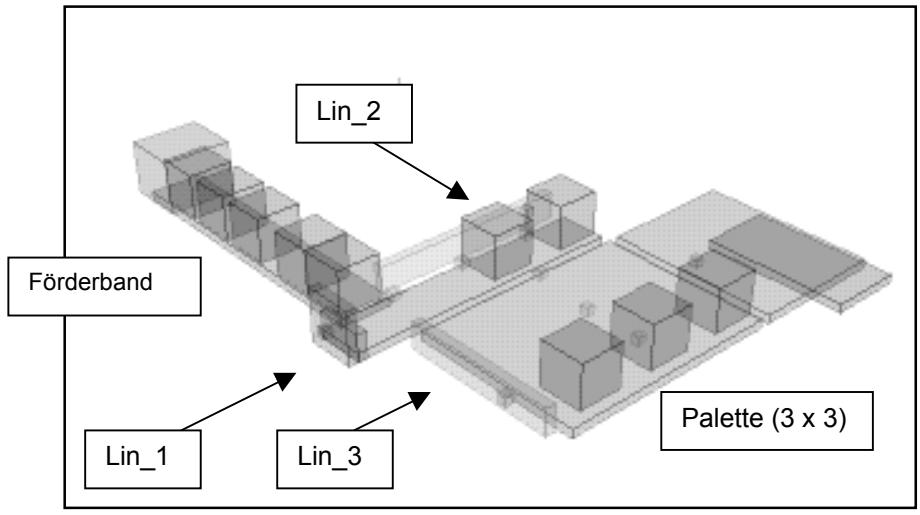
Projekt 30: Palettensteuerung

Schwerpunkte: Aufruf mehrerer FCs, Schrittkettennachbildung, Zählschaltung, Vergleicher

Aufgabe: Hinter einer Abfüllanlage sollen die Pakete in einer Anordnung 3 x 3 gepackt und anschließend gemeinsam auf eine zweite Palette geschoben werden. Die Pakete können angefordert werden, wenn das Förderband eingeschaltet ist. Im Einzelbetrieb (Automatik ist ausgeschaltet), stoppt die Anlage nach einem Zyklus, d.h. wenn eine Palette voll bepackt ist. Bei Automatikbetrieb werden die Zyklen fortgesetzt, sofern Pakete angefordert werden.
Beim Austasten mit <AUS> verliert die Steuerung sein "Gedächtnis". Die Bänder u. Paletten müssen dann von Hand geleert werden.
Bei ausgeschalteter Anlage kann die Palette waagerecht und senkrecht geleert werden.
Bei Betätigung des <STOP>-Schalters wird die Anlage nur angehalten und arbeitet nach dem zweiten Druck auf <STOP> ganz normal weiter.

Hinweis: Sollten sich in Ihrem eigenen Programm Pakete an ungewollter Stelle anhäufen, können diese im TrySim-Programm unter <Anlage> <Dynamiks löschen> <Nicht speicherbare> alles bereinigen.

Anlage:



"Lin_1" bis "Lin_3" sind die symbolischen Bezeichnungen für die **Lin**earbeweger (Antriebe), mit denen die Pakete verschoben werden.

Theoretische Erörterung:
Dieses Projekt soll als Beispiel dienen für das Zusammenwirken verschiedener kleiner Projekte, die miteinander verzahnt sind.
Das Programm wird in verschiedene Unterprogramme für entsprechende Teilprojekte aufgeteilt, die nacheinander - in logischer Reihenfolge - vom OB 1 aufgerufen werden.
Laut Aufgabenstellung ergibt sich eine zwingende Reihenfolge der Abläufe, die auch in der Reihenfolge der aktiven Netzwerke zu berücksichtigen ist. Diese zwingende Reihenfolge wird durch folgendes Prinzip erreicht:

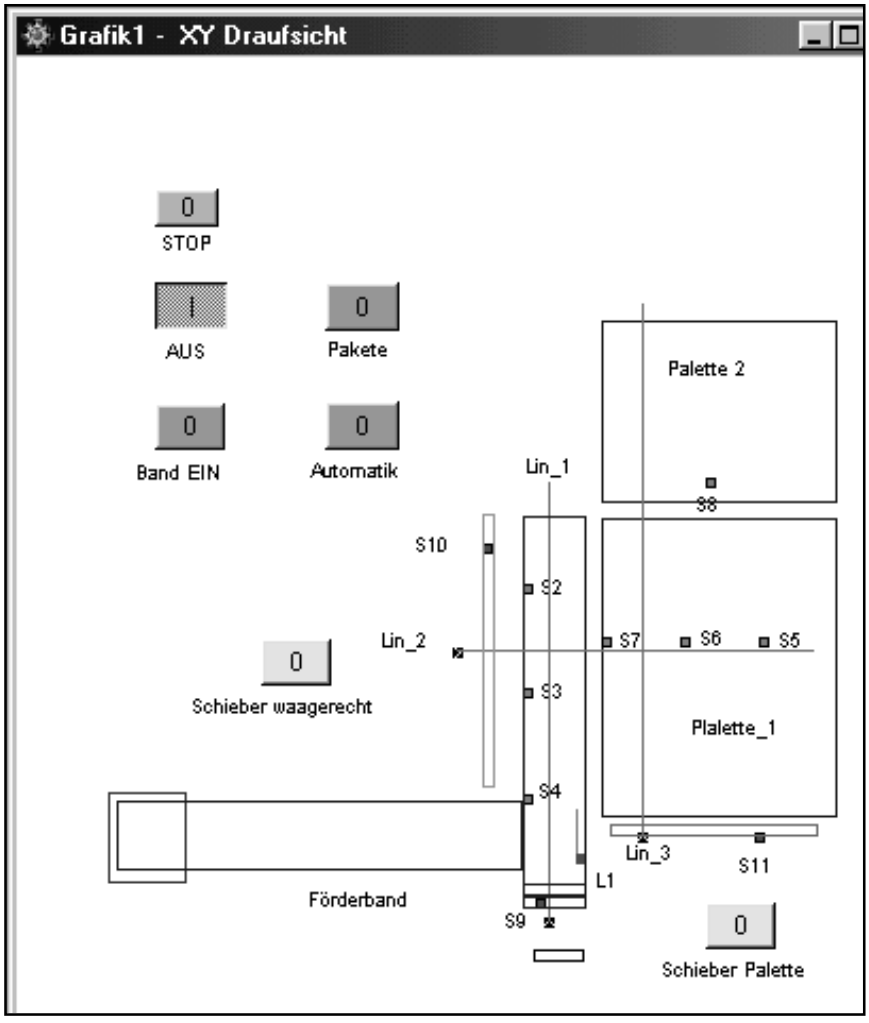
- Es werden nicht direkt die Ausgänge programmiert, sondern Merker.
- Die Ausgänge werden in separaten Netzwerken durch die Merker aktiviert.
- Es ist jeweils immer nur ein Merker-Netzwerk aktiv.
- Das jeweils aktive Netzwerk schaltet immer das folgende Netzwerk frei.
- Dieses folgende - dann aktive - Netzwerk setzt das vorausgehende Netzwerk auf "0" zurück.

Es soll folgende **Grobstruktur** angewendet werden:

OB 1; Netzwerk 1: Sprung zur FC 1: Start u. Förderband.
OB 1; Netzwerk 2: Sprung zur FC 2: Positionieren mit Lin_1 (3 Pakete in Reihe).
OB 1; Netzwerk 3: Sprung zur FC 3: Wahl der Palettenreihe mit Zählerauswertung.
OB 1; Netzwerk 4: Sprung zur FC 4: Palette beschicken mit Lin_2 (in 3 Reihen).
OB 1; Netzwerk 5: Sprung zur FC 5: Palette leeren mit Lin_3.
OB 1; Netzwerk 6: Sprung zur FC 6: Ausgänge.

Das Projekt ist vorbereitet unter <Vorlagen> <Palette_FC_V>. Sie können diese Vorlage öffnen und dann in Ihr Arbeitsverzeichnis kopieren unter <Projekt> <Speichern unter...>.

Vergleiche TrySim-Pult <Palette_FC_V>



Wenn ein Paket vom Förderband auf das senkrecht dargestellte Rollband geschoben wird, spricht die Lichtschranke L1 an. (Ob "0" oder "1", sollen Sie feststellen). Die Ausgangslagen der Schieber <Lin_1> bis >Lin_3> werden durch die Endtaster <S9> bis <S11> erfasst. Die Positionen der Pakete werden mit <S2> bis <S4> bzw. <S5> bis <S7> erfasst. <S8> ist der Endtaster für den Schieber <Lin_3>.
Bei ausgeschalteter Anlage kann die Palette waagerecht und senkrecht mit den <Schieber>-Tastern geleert werden.

Die **Symboltabelle** ist bereits erstellt.

Symbolische Adresse	Operand	Datentyp	Kommentar
A_FBand	A 0.0	BOOL	Förderbandantrieb
PaketGen	A 0.1	BOOL	Paketgenerator
A_LIN1	A 0.3	BOOL	Linearbeweger_1 senkrecht
A_LIN2	A 2.0	BOOL	Linearbeweger_2 waagerecht
A_LIN3	A 2.1	BOOL	Linearbeweger_3 senkrecht Palette
LiSchr	E 0.0	BOOL	Lichtschranke
TBandEIN	E 0.1	BOOL	Taster Förderband EIN (S)
S2	E 0.2	BOOL	Position oben
S3	E 0.3	BOOL	Position Mitte
S4	E 0.4	BOOL	Position unten
T_Lin_2	E 0.5	BOOL	Taster Lin_2 schieben v. Hand
S5	E 0.7	BOOL	Pos Palette rechts
S6	E 1.0	BOOL	Pos Palette Mitte
S7	E 1.1	BOOL	Pos Palette links
T_Lin_3	E 1.2	BOOL	Taster senkr. schieben v. Hand
S8	E 1.3	BOOL	Palettenschieber oben
S9	E 1.4	BOOL	Hochschieber unten
S10	E 1.5	BOOL	Querschieber links
S11	E 1.6	BOOL	Palettenschieber unten
T_AUS	E 1.7	BOOL	Taster AUS (Ö)
TPaket	E 2.0	BOOL	Taster für Pakete
T_AUTO	E 2.1	BOOL	Taster für Automatikbetrieb (S)
STOP	E 2.2	BOOL	Stopschalter zum Anhalten
M_FB1	M 1.0	BOOL	Förderband-Merker
MLin1_S2	M 1.1	BOOL	Merker Lin_1 bis S2
MLin1_Z1	M 1.3	BOOL	Merker Lin_1 zurück (1)
PAKET_2	M 1.4	BOOL	Paket vorlegen (2)
MLin1_S3	M 1.5	BOOL	Merker Lin_1 bis S3
MLin1_Z2	M 1.6	BOOL	Merker Lin_1 zurück (2)
PAKET_3	M 1.7	BOOL	Paket vorlegen (3)
MLin1_S4	M 2.0	BOOL	Merker Lin_1 bis S4
MLin1_Z3	M 2.1	BOOL	Merker Lin_1 zurück (3)
MLin_2_V	M 2.2	BOOL	Merker Lin_2 nach rechts
MLin_2_Z	M 2.3	BOOL	Merker Lin_2 n. links zurück
MLin_3_V	M 2.4	BOOL	Merker Lin_3 nach oben
MLin_3_0	M 2.5	BOOL	Merker Lin_3 in GrundPos.
M_S5	M 3.0	BOOL	Zählerstanderfassung Reihe rechts
M_S6	M 3.1	BOOL	Zählerstanderfassung Reihe Mitte
M_S7	M 3.2	BOOL	Zählerstanderfasssung Reihe links
MNeuStrt	M 4.0	BOOL	Neustart des FBandes
STARTMRK	M 5.0	BOOL	Startmerker
Zähler	Z 1	COUNTER	Zähler für Reihen

Alle symbolischen Adressen, die mit "M.." beginnen, deuten auf Merker hin.
Alle symbolischen Adressen, die mit "A.." beginnen, deuten auf Ausgänge hin.

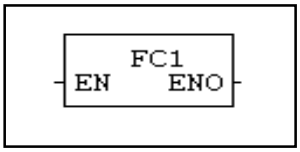
Bevor im OB 1 die FCs aufgerufen werden können, müssen diese unter <SPS>|<NEU>|<FC..> angelegt und abgespeichert worden sein.

Danach können alle FCs der Reihe nach in OB 1-Netzwerken angelegt werden. Man kann aber auch nach jedem FC-Aufruf die jeweilige FC programmieren. In diesem Beispiel wurde der zweite Weg gewählt, um die logische Verknüpfung zwischen dem OB 1-Aufruf und dem FC-Programm zu unterstreichen. Am Ende dieses Projektes finden Sie alle fertigen Netzwerke der Reihe nach aufgelistet.
Zuerst muss also FC 1 angelegt und gespeichert werden.

OB 1; Netzwerk 1: Sprung zur FC 1: Start u. Förderband

In FUP wählen Sie bei aktivem Editierfenster das Icon <FUP_Elemente> <FC> und doppelklicken mit <LM> <FC 1>.

Aufruf in FUP oder: Aufruf in AWL:

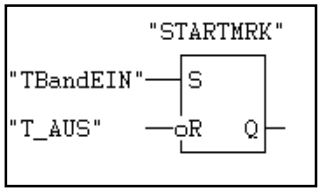



```
CALL FC 1
```

Hier wird nur zur FC 1 gesprungen. Das Unterprogramm wird dort in den Netzwerken erstellt.

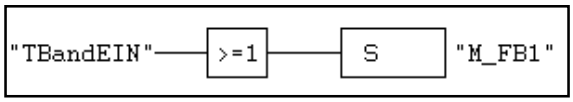
Danach wird unter <SPS> <Öffnen> der Baustein <FC 1> geöffnet. Dort wird im Kopf nichts deklariert. Es werden in den Netzwerken globale Variablen abgefragt, auf die ohne den "Umweg" über die Formalparameter zugegriffen werden kann.

FC 1; Netzwerk 1: Startmerker



Hier wird lediglich festgehalten, ob die Anlage eingeschaltet ist. Gesetzt wird das FF mit dem Taster <Band EIN>. Ausgeschaltet wird mit dem Taster <AUS>. (Wenn der Öffner betätigt wird, liegt eine "0" an E 1.7 (T_AUS). Da das FF aber mit "1" zurückgesetzt wird, muss das Signal invertiert werden).

FC 1; Netzwerk 2: M: Förderband_Merker, (Rumpfprogramm)



Gestartet wird über den Taster <BandEIN> (S) an "TBandEIN".

Rückgesetzt wird der Merker erst vom folgenden Merker M1.1 (MLin1_S2) (siehe FC 2; Netzwerk 1). Später werden alle Merker zusätzlich zentral mit <AUS> zurückgesetzt.

Es wird nicht direkt die Aktion - der Ausgang - programmiert, sondern immer erst nur ein Schrittmerker, der **an anderer Stelle abgefragt** wird.

Beachten Sie bitte das bereits oben erläuterte Prinzip:

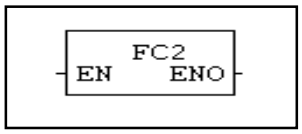
- Das jeweils aktive Netzwerk schaltet immer das folgende Netzwerk frei.
- Dieses folgende - dann aktive - Netzwerk setzt das vorausgehende Netzwerk auf "0" zurück.

Das passiert im neuen Netzwerk, das über OB 1 aufgerufen wird.

Das gerade erstellte Netzwerk reicht im Endstadium natürlich nicht aus, um das Förderband programmgerecht laufen zu lassen. Für den Anfang reicht es aber aus. Die einzelnen Schritte sollen logisch entwickelt werden, so dass erst später über weitere Setzbedingungen nachgedacht werden muss.

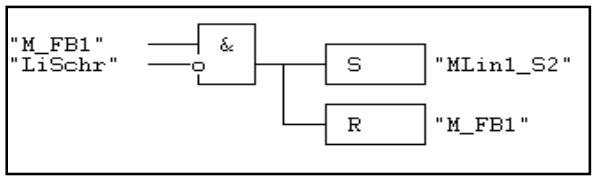
Jetzt muss eine neue Funktion (FC) erzeugt und abgespeichert werden, die dann vom OB 1 aufgerufen wird.

OB 1; Netzwerk 2: Positionieren mit Lin_1. (3 Pakete in Reihe)



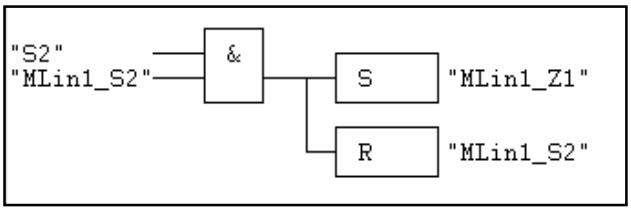
Hier wird nur zur FC 2 gesprungen. Das Unterprogramm wird dort in den Netzwerken erstellt.

FC 2; Netzwerk 1: M: Schieber Lin_1 bis S2 ("M:" steht für Merkerprogrammierung)



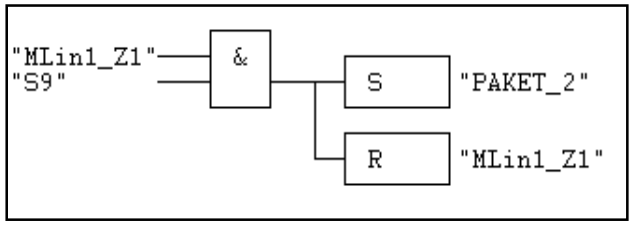
Durch das vorherige Netzwerk 2 der FC 1 wurde <M_FB1> auf "1" gesetzt. Damit wird dieses Netzwerk 1 in FC 2 vorbereitet.
Durch die Lichtschranke <L1> an <LiSchr> wird in diesem Folgeschritt M1.1 (<MLin1_2S>) gesetzt. Hier wird noch kein Ausgang gesetzt, sondern nur ein Merker!
Gleichzeitig wird der setzende Merker <M_FB1> zurückgesetzt.
(Sie können die Funktion der Ausgangssymbole zwischen "=", "S" oder "R" verändern, indem Sie nach der Markierung die Leertaste betätigen).
Erst in diesem Folgeschritt wird der vorherige Schritt zurückgesetzt.
Für die Ausgänge bedeutet das, dass das Förderband abgeschaltet und der Linearvorschub 1 eingeschaltet wird. Dazu werden in FC 6 folgende Ergänzungen vorgenommen:

FC 2; Netzwerk 2: M: Schieber Lin_1 zurück (1)



<MLin1_S2> bereitet dieses Netzwerk vor. Durch den Grenztaster <S2> wird dieser nächste Schritt M1.3 (<MLin1_Z1>) eingeschaltet und wiederum der vorherige Schrittmerker zurückgesetzt.

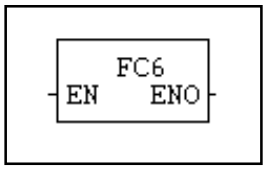
FC 2; Netzwerk 3: M: Paket vorlegen (2)



<S9> gibt 1-Signal, wenn die Grundstellung erreicht ist. Danach kann das nächste Paket gefördert werden. Dazu muss in der FC 6 der Bandantrieb aktiviert werden. Um die bisherige Anlage schon einmal zu testen, bietet sich an, diese FC 6 im OB 1 anzulegen und dort das Netzwerk für den Antrieb in der Grundform zu programmieren.

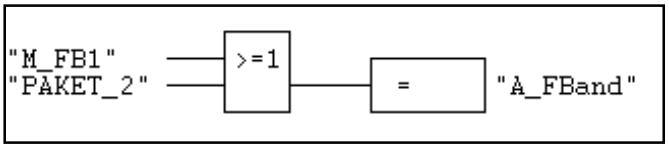
Jetzt muss eine neue Funktion (FC) erzeugt und abgespeichert werden, die dann vom OB 1 aufgerufen wird.

OB 1; Netzwerk 6: Ausgänge



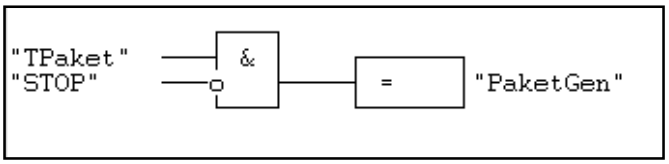
Hier wird nur zur FC 6 gesprungen. Das Unterprogramm wird dort in den Netzwerken erstellt.

FC 6; Netzwerk 1: A: Bandantrieb, Grundform



Wenn der Förderbandmerker "1" liefert oder die zweite Paketfreigabe erfolgt, wird der Förderband-Antrieb gestartet. Der Antrieb wird nicht gespeichert, sondern fällt auf "0" zurück, wenn das VKE "0" ist.

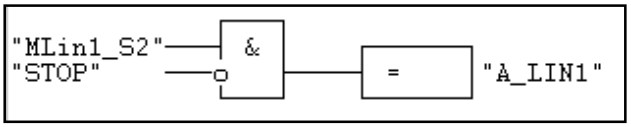
FC 6; Netzwerk 2: Paketgenerator



Irgendwo im Programm muss auch einmal die Freigabe für die Pakete erscheinen:
Wenn der Taster <Pakete> gedrückt wird, wird ein neues Paket erzeugt, das auf das Band fällt.
Das funktioniert aber nur, wenn der <STOP>-Schalter nicht betätigt wird.

Der erste Linearantrieb <Lin_1> soll für ein kleines lauffähigesTeilprojekt auch noch programmiert werden:

FC 6; Netzwerk 3: A: Schieber Lin_1 nach oben. Grundform



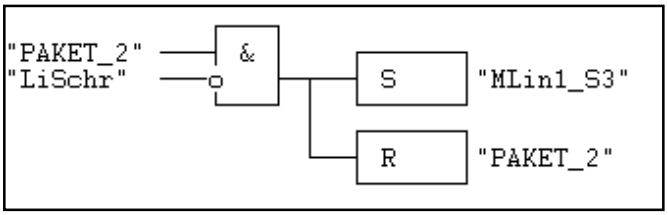
Merker <MLin1_S2> führt so lange "1", bis Endtaster <S2> erreicht ist.

Der Schieber fährt im unerregten Zustand des Ventils pneumatisch in die Grundstellung zurück. Deshalb muss für den Rücklauf kein Ausgang aktiviert werden.

Vielleicht möchten Sie das bisherige Programm testen? Sie brauchen nur unter <Projekt> <Alles speichern> drücken und das Auge-Icon einschalten.
Dann ist nur noch das Band einzutasten (Farbumschlag rot) und den <Pakete>-Schalter einschalten. Die Paketreihe wird von der Lichtschranke gestoppt, Lin_1 sollte bis zu <S2> vorfahren und anschließen bis in die Grundstellung zurückfahren.

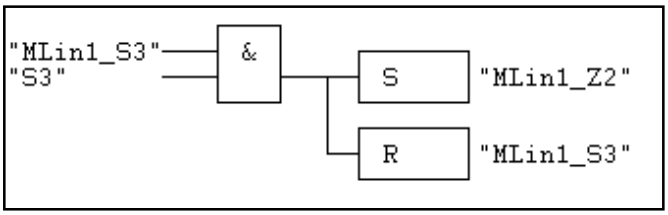
Danach können Sie mit <Anlage> <Dynamics> <Normale löschen> die Probepakete entfernen und mit der Programmierung fortfahren:

FC 2; Netzwerk 4: M: Schieber Lin_1 bis Mittelposition S3



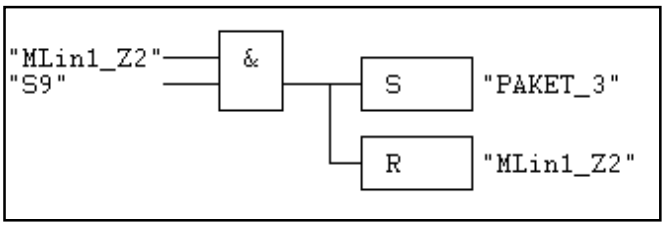
<Paket_2> schaltet nicht nur in der FC 6 den Bandantrieb ein, sondern bereitet den weiteren Schrittmerker <MLin1_S3> vor.

FC 2; Netzwerk 5: M: Lin_2 zurück ab Mittelposition.



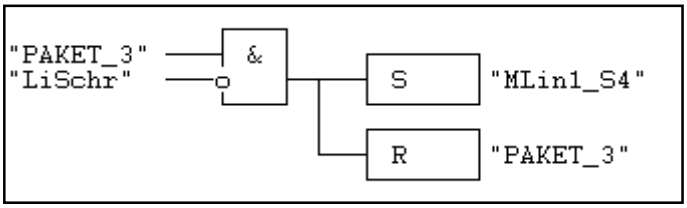
Durch den entsprechenden Grenztaster <S3> wird der Vorschub beendet. Der Schieber fährt pneumatisch gesteuert zurück in die Grundstellung.

FC 2; Netzwerk 6: M: Paket vorlegen (3)



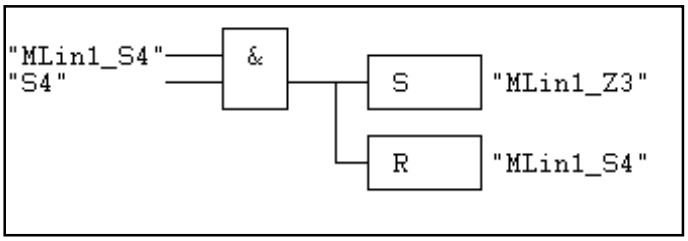
Ein drittes Paket wird vom Förderband auf <Lin_1> befördert, wenn <Lin_1> in der Grundstellung angekommen ist.

FC 2; Netzwerk 7: M: Schieber Lin_1 bis S4



Die Lichtschranke setzt den Merker <Paket_3> und damit den Antriebsmotor des Förderbandes zurück. <MLin1_S4> wird gesetzt und schaltet <Lin_1> ein, bis der Schieber <S4> erreicht.

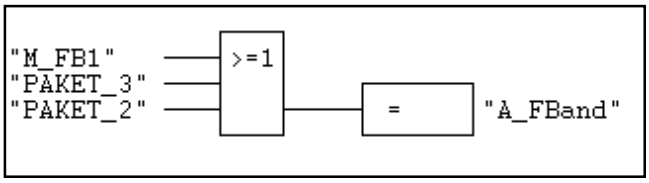
FC 2; Netzwerk 8: M: Lin_2 zurück (3)



Durch <S4> wird <Lin_1> zurückgesetzt.

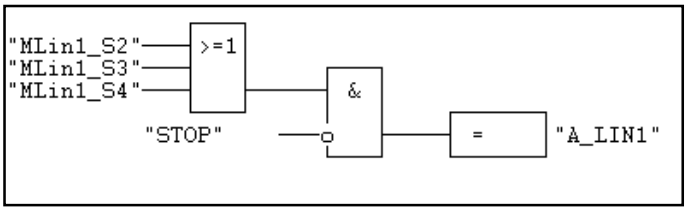
In diesem Stadium könnten Sie das bisherige Programm wieder testen. Dazu sind nur an den bestehenden Ausgängen (Operanden) für das Förderband und <Lin_1> einige Erweiterungen vorzunehmen.

FC 6; Netzwerk 1: A: Erweiterung der Bandsteuerung



Für den dritten Vorschub gibt <PAKET_3> "1"-Signal.

FC 6; Netzwerk 3: A: Schieber Lin_1 nach oben



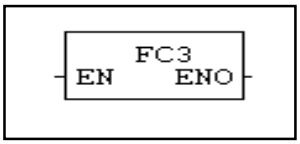
Die Merker <MLin_1> bis MLin_3> starten jeweils den Linearvorschub <Lin_1>. Die Merker - und damit der Antrieb - werden mit den entsprechenden Endtastern in der FC 2 abgeschaltet.

Diese Befehle müssen alle in einem Netzwerk für den (einen) Ausgang gegeben werden. Ein "beliebter" Fehler ist, denselben Ausgang in verschiedenen Netzwerken wiederholt, aber von unterschiedlichen Operanden zu schalten. Dann würde nur die Verknüpfung des letzten Netzwerkes wirksam!

Wenn dieser Abschnitt zur Zufriedenheit läuft, reicht ein einziges Merkerbit, um ein weiteres - eigentlich eigenständig zu programmierendes - Teilprojekt zu beginnen. Der letzte, noch aktive Merker ist <MLin1_Z3>, was bedeutet, der Antrieb <Lin_1> ist nach dem Transport von 3 Paketen in die Grundstellung zurückgefahren. Hier wird später angeknüpft.
Nach dem gleichen Schema - Ansteuerung von verschiedenen Endtastern - könnte die Palette mit 3 Reihen von Paketen bestückt werden. Die folgende FC zeigt eine Variante der Positionierung auf, ohne entscheiden zu wollen, welche die bessere Lösung ist. Es ist schlicht eine Übung mit einem Zähler.

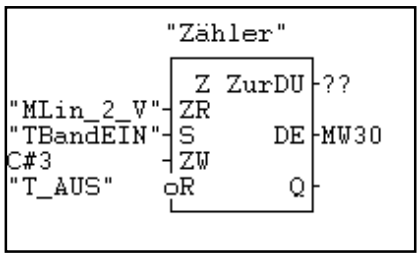
Jetzt muss eine neue Funktion (FC) erzeugt und abgespeichert werden, die dann vom OB 1 aufgerufen wird.

OB 1; Netzwerk 3: Wahl der Palettenreihe durch Zählerauswertung



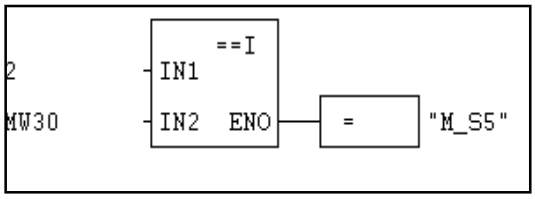
Hier wird nur zur FC 3 gesprungen. Das Unterprogramm wird dort in den Netzwerken erstellt.

FC 3; Netzwerk 1: Wahl der Palettenreihe



Durch diesen Rückwärtszähler wird die Zahl der freien Reihen auf der Palette erfasst. Beim ersten Bandstart wird mit dem Taster <Band EIN> über "TBandEIN" der Zählerbaustein auf die Zahl "3" eingestellt. Immer, wenn der Schieber <Lin_2> eine Reihe von 3 Paketen auf die Palette schiebt, zählt der Zähler um 1 zurück.
Der Zählerstand wird über das Merkerwort "MW30" in den folgenden Netzwerken ausgewertet.

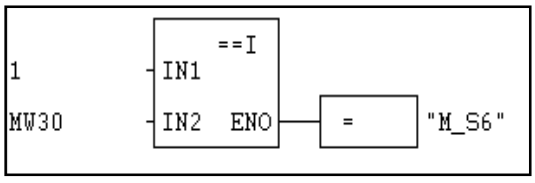
FC3; Netzwerk 2: Zählerstanderfassung Reihe rechts



Den Vergleicher finden Sie mit Icon <FUP-Elemente>, dort unter <==Int>.
Wenn das MW30 den Wert 2 enthält, ist das VKE an <M_S5> "1". Der Vergleichswert "2" wird an <IN1> geschrieben.

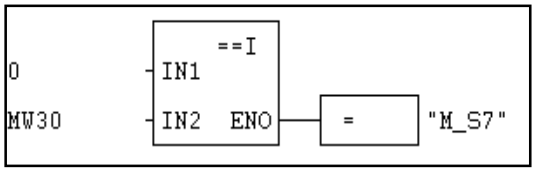
Durch dieses Netzwerk wird entschieden, wie weit der Schieber <Lin_2> fährt.

FC3; Netzwerk 3: Zählerstanderfassung Mitte



Wenn das MW30 den Wert 1 enthält, ist das VKE an <M_S6> "1". Der Vergleichswert "1" wird an <IN1> geschrieben.

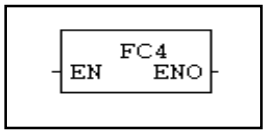
FC3; Netzwerk 4: Zählerstanderfassung Reihe links



Wenn das MW30 den Wert 0 enthält, ist das VKE an <M_S7> "1". Der Vergleichswert "0" wird an <IN1> geschrieben.

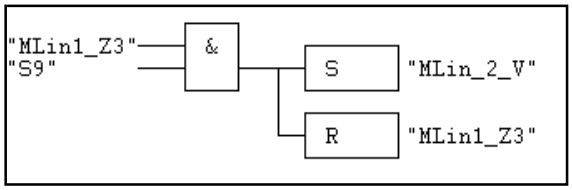
Für die Verschiebung der Paketreihe wird eine neue Funktion (FC) erzeugt und abgespeichert, die dann vom OB 1 aufgerufen wird.

OB 1; Netzwerk 4: Palette beschicken mit Lin_2



Hier wird nur zur FC 4 gesprungen. Das Unterprogramm wird dort in den Netzwerken erstellt.

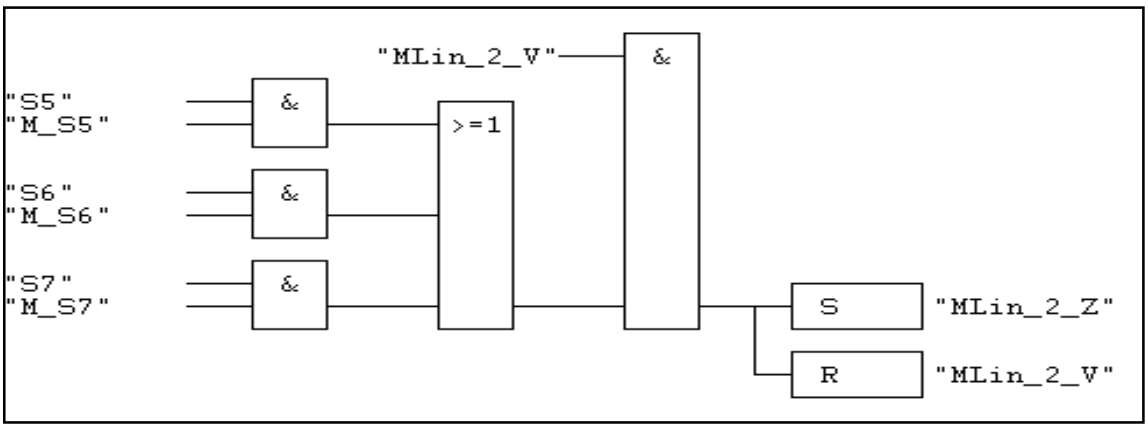
FC 4; Netzwerk 1: M: 3 Pakete waagerecht verschieben



MLin1_Z3 ist der Schrittmerker aus der FC 2, der als einzige Abfrage in diesen Teil des Projektes übernommen wird.

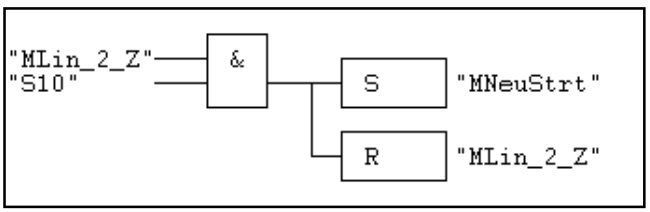
Nachdem die Schrittmerker bis hierher eine zwingende Reihenfolge vorgegeben haben, gibt der Merker MLin1_Z3 das Schieberventil "LIN_2" frei, wenn <Lin_1> seine Ausgangsposition <S9> erreicht hat.

FC 4; Netzwerk 2: M: SCHIEBER Lin_2 nach links ZURÜCK



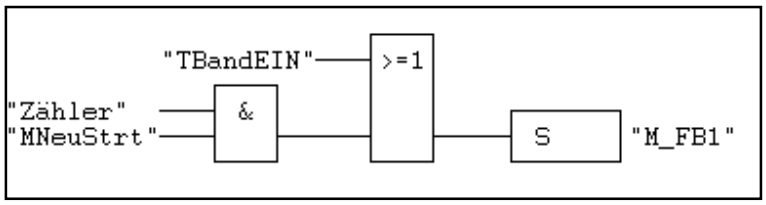
Wenn die berechnete Plattenposition erreicht ist, wird der Vorschub abgeschaltet und der Schieber fährt zurück in die Grundposition.

FC 4; Netzwerk 3: M: Grundposition Schieber Lin_2 links erreicht



Da bisher nur eine Reihe auf die Palette geschoben wurde, muss der Zyklus von Anfang an neu beginnen. Der erste Schrittmerker M_FB1 muss wieder gestartet werden. Dafür wird das entsprechende Netzwerk erweitert:

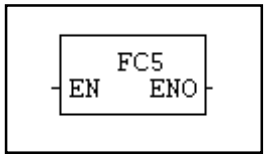
FC 1; Netzwerk 2: M: Förderband_Merker; Erweiterung



Wenn der Zählerausgang Q (der einfach mit "Z1" oder "Zähler" als Abfragebit aufgerufen werden kann) eine "1" führt (wenn also der Zählerstand >0 ist), und mit MNeuStrt ein neuer Zyklus freigegeben wurde, wird der Bandmerker ebenfalls gesetzt.

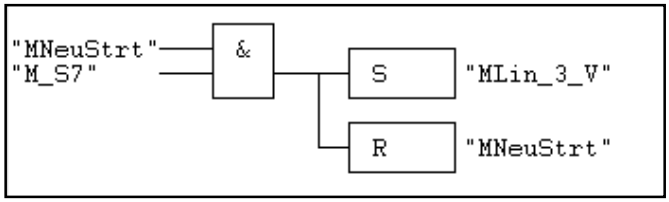
Die FC 2 wird ohne weitere Programmierung abgearbeitet, ebenso FC 3.und FC 4 (mit veränderten Werten).
Nachdem sich die ganze Palette gefüllt hat, muss eine neue Funktion (FC) erzeugt und abgespeichert werden, die dann vom OB 1 aufgerufen wird.

OB 1; Netzwerk 5: Palette leeren mit Lin_3



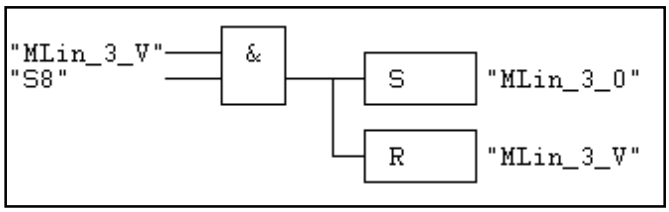
Hier wird nur zur FC 5 gesprungen. Das Unterprogramm wird dort in den Netzwerken erstellt.

FC 5; Netzwerk 1: M: PalettenSchieber Lin_3 senkrecht hochfahren



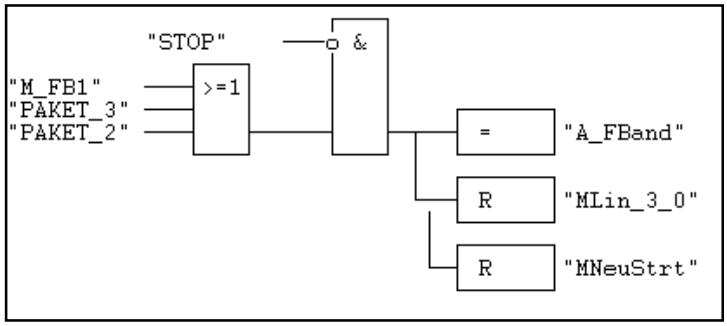
Wenn der Schieber Lin_2 in Grundposition steht (MNeuStrt = "1") und die Zählerauswertung der FC 3 anzeigt, dass die Palette voll ist (M_S7 = "1"), soll die Palette über MLin_3 geleert werden.

FC 5; Netzwerk 2: M: Palettenschieber senkrecht zurück.



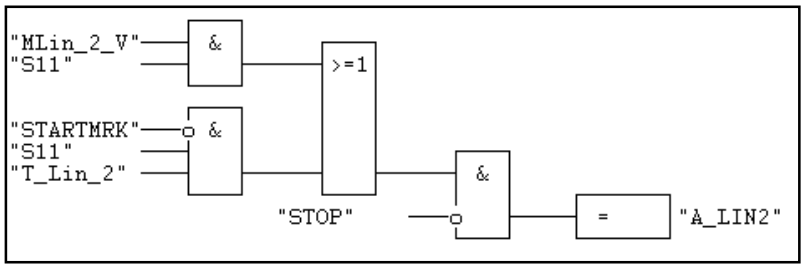
Wenn Lin_3 den Endtaster <S8> erreicht hat, wird MLin_3_V zurückgesetzt und damit auch der Antrieb.

FC 6; Netzwerk 1: A: Bandantrieb, vervollständigt



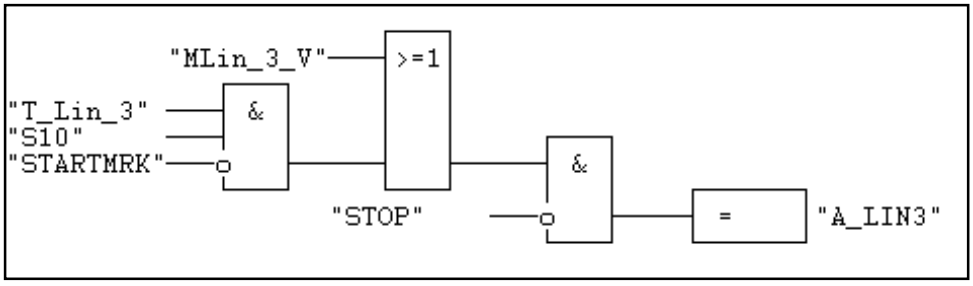
Jetzt sind noch die Ausgänge zu vervollständigen. Mit der <STOP>-Taste soll der Ausgang abgeschaltet werden können.
Wenn das Förderband wieder eingeschaltet wird, wird der letzte Schrittmerker (MLin_3_0) zurückgesetzt. Ebenfalls MNeuStrt.

FC 6; Netzwerk 4: A: 3 Pakete waagerecht verschieben. (Lin_2 nach rechts)



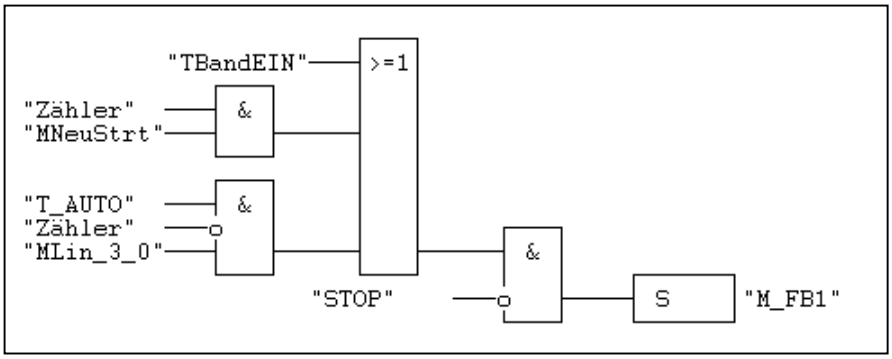
Wenn <Lin_3> in Grundposition steht, liefert <S11> "1"-Signal. Erst dann darf <Lin_2> aktiv werden.
Wenn die Anlage ausgeschaltet wurde, liefert STARTMERK "0"-Signal. Wenn dann der Taster <Schieber waagerecht> über T_Lin_2 "1"-Signal liefert und <STOP> nicht betätigt wird, kann der Schieber <Lin_2> "von Hand" geführt werden.

FC 6; Netzwerk 5: A: Palette leeren. (Lin_3 hochfahren)



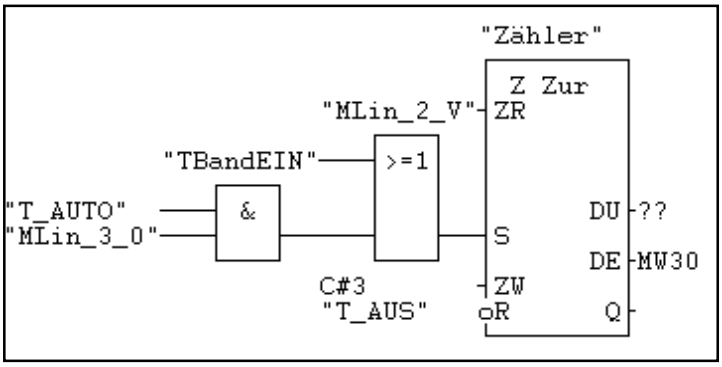
MLin_3_V steuert <Lin_3> automatisch.
Wenn <Lin_2> in Grundposition steht, liefert <S10> "1"-Signal. Erst dann darf <Lin_3> aktiv werden.
Wenn die Anlage ausgeschaltet wurde, liefert STARTMERK "0"-Signal. Wenn dann der Taster <Schieber senkrecht> über T_Lin_3 "1"-Signal liefert und <STOP> nicht betätigt wird, kann der Schieber <Lin_3> "von Hand" geführt werden.

FC 1; Netzwerk 2: vollständige Version



Außer in der bisherigen Version darf der Band-Merker auch gesetzt werden, wenn <Automatik> eingeschaltet wurde, der Zähler auf "0" gesetzt wurde (Palette ist voll) und der Schieber <Lin_3> wieder in der Grundposition angekommen ist.

FC 3; Netzwerk 1: Wahl der Palettenreihe, vollständig



Der Zähler wird auch bei Automatikbetrieb (T_AUTO = "1") auf "3" gesetzt, wenn zusätzlich der Schrittmerker MLin_3_0 gesetzt ist, d.h., wenn die Palette geleert wurde.
Zum Schluss sollen alle Schrittmerker - und damit auch alle Ausgänge - mit dem Taster <AUS> über „T_AUS“ zurückgesetzt werden. Hierfür wird ein abschließendes Netzwerk im OB 1 gewählt.

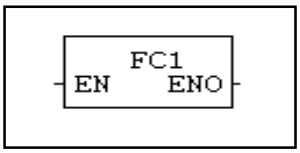
OB 1; Netzwerk 7: Rücksetzen der Merker durch "AUS"

```
UN "T_AUS"        // Wenn der Taster betätigt wird, liefert er eine "0"
R  "M_FB1"        // Dann werden alle aufgeführten Merker rückgesetzt.
R  "MLin1_S2"
R  "MLin1_Z1"
R  "PAKET_2"
R  "MLin1_S3"
R  "MLin1_Z2"
R  "PAKET_3"
R  "MLin1_S4"
R  "MLin1_Z3"
R  "MLin_2_V"
R  "MLin_2_Z"
R  "MLin_3_V"
R  "MLin_3_0"
R  "M_S5"
R  "M_S6"
R  "M_S7"
R  "MNeuStrt"
```

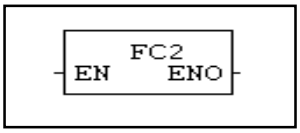
Es folgt noch einmal die geordnete Zusammenstellung der Netzwerke

OB 1

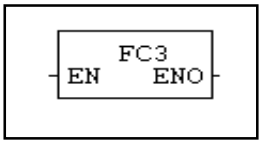
OB 1; Netzwerk 1: Sprung zur FC 1: Start u. Förderband



OB 1; Netzwerk 2: Positionieren mit Lin_1. (3 Pakete in Reihe)

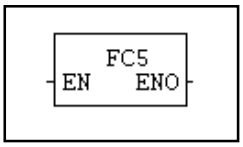


OB 1; Netzwerk 3: Wahl der Palettenreihe durch Zählerauswertung

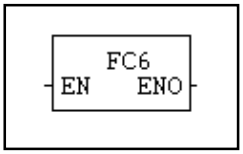


OB 1; Netzwerk 4: Palette beschicken mit Lin_2

OB 1; Netzwerk 5: Palette leeren mit Lin_3



OB 1; Netzwerk 6: Ausgänge

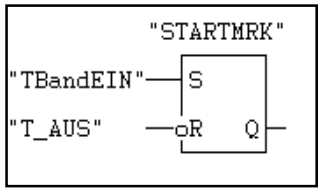


OB 1; Netzwerk 7: Rücksetzen der Merker durch "AUS"

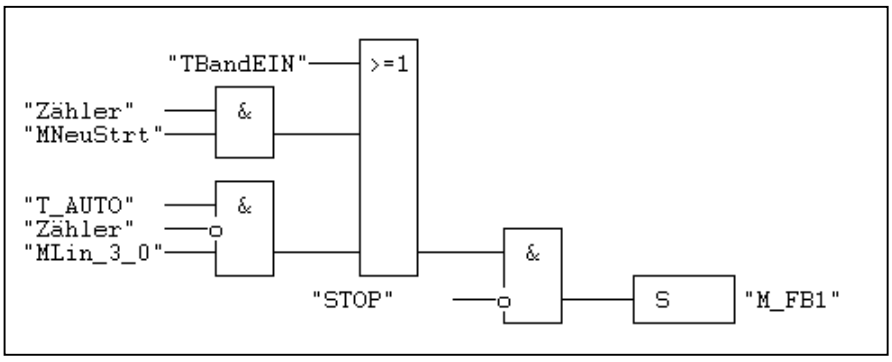
```
UN "T_AUS"        // Wenn der Taster betätigt wird, liefert er eine "0"
R  "M_FB1"        // Dann werden alle aufgeführten Merker rückgesetzt.
R  "MLin1_S2"
R  "MLin1_Z1"
R  "PAKET_2"
R  "MLin1_S3"
R  "MLin1_Z2"
R  "PAKET_3"
R  "MLin1_S4"
R  "MLin1_Z3"
R  "MLin_2_V"
R  "MLin_2_Z"
R  "MLin_3_V"
R  "MLin_3_0"
R  "M_S5"
R  "M_S6"
R  "M_S7"
R  "MNeuStrt"
```

FC 1 - Start u. Förderband

FC 1; Netzwerk 1: Startmerker

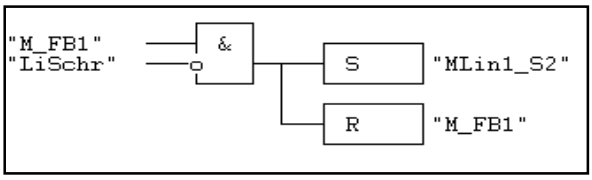


FC 1; Netzwerk 2: Merker Förderband

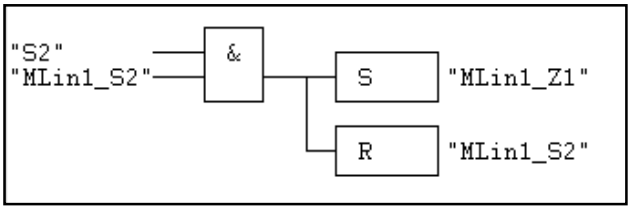


FC 2 - Positionieren mit Lin_1 (3 Pakete in Reihe)

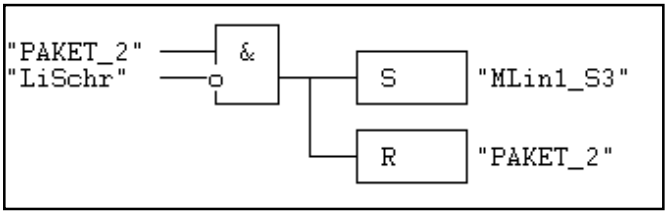
FC 2; Netzwerk 1: M: Schieber Lin_1 bis S2 ("M:" steht für Merkerprogrammierung)



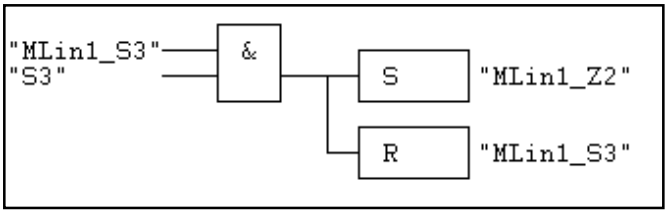
FC 2; Netzwerk 2: M: Schieber Lin_1 zurück (1)



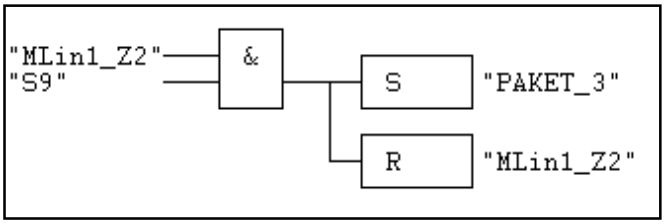
FC 2; Netzwerk 4: M: Schieber Lin_1 bis Mittelposition S3



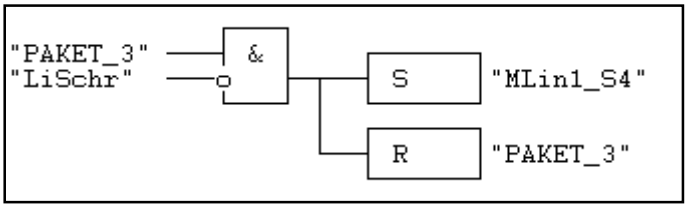
FC 2; Netzwerk 5: M: Lin_2 zurück ab Mittelposition (2).



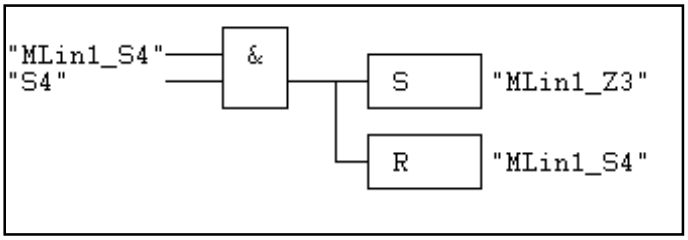
FC 2; Netzwerk 6: M: Paket vorlegen (3)



FC 2; Netzwerk 7: M: Schieber Lin_1 bis S4

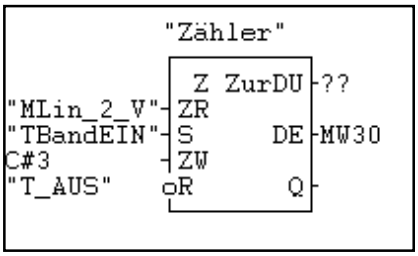


FC 2; Netzwerk 8: M: Lin_2 zurück (3)

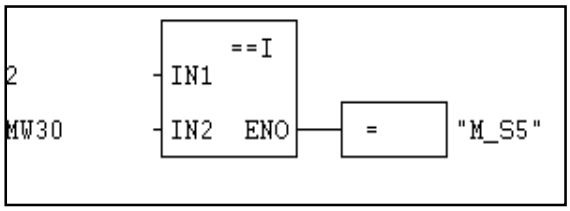


FC 3 - Wahl der Palettenreihe mit Zählerauswertung

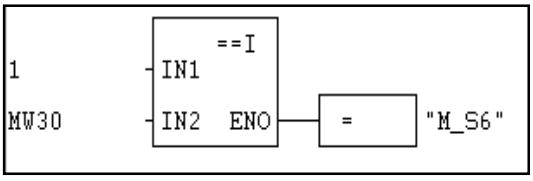
FC 3; Netzwerk 1: Wahl der Palettenreihe



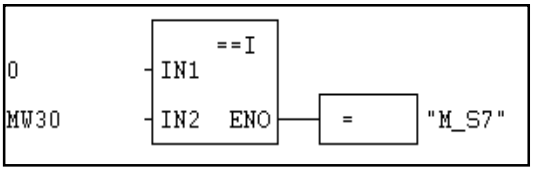
FC3; Netzwerk 2: Zählerstanderfassung Reihe rechts



FC3; Netzwerk 3: Zählerstanderfassung Mitte

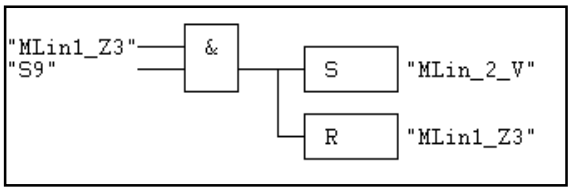


FC3; Netzwerk 4: Zählerstanderfassung Reihe links

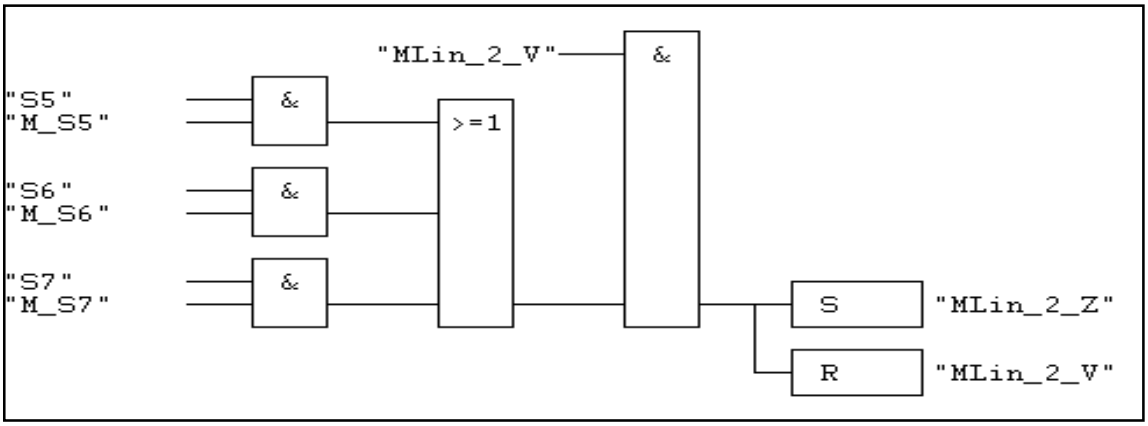


FC 4 - Palette beschicken mit Lin_2 (in 3 Reihen).

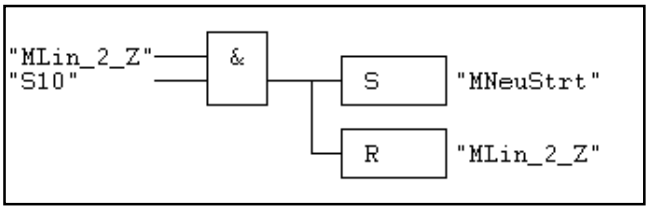
FC 4; Netzwerk 1: M: 3 Pakete waagerecht verschieben



FC 4; Netzwerk 2: M: SCHIEBER Lin_2 nach links ZURÜCK

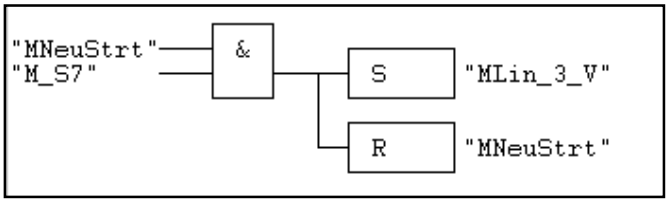


FC 4; Netzwerk 3: M: Grundposition Schieber Lin_2 links erreicht

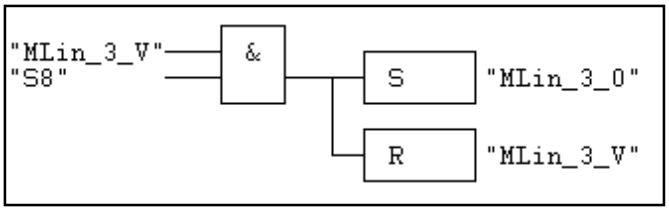


FC 5 - Palette leeren mit Lin_3

FC 5; Netzwerk 1: M: PalettenSchieber Lin_3 senkrecht hochfahren

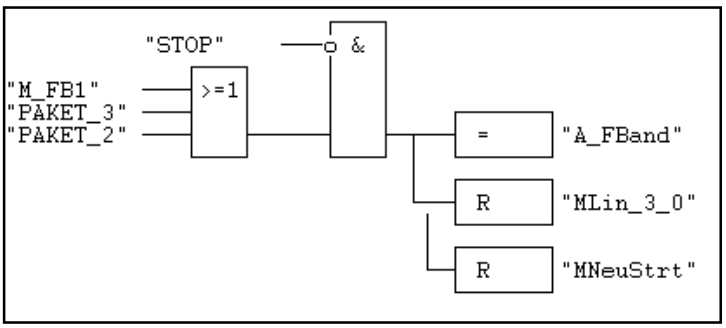


FC 5; Netzwerk 2: M: Palettenschieber senkrecht zurück.

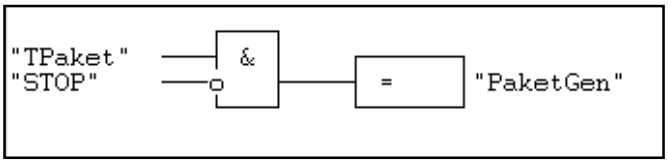


FC 6 - Ausgänge.

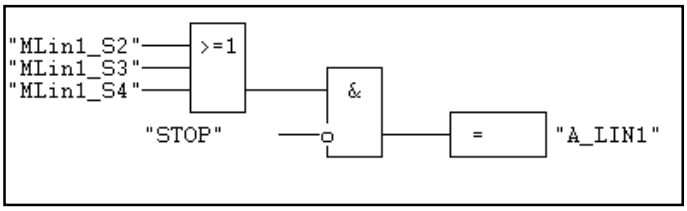
FC 6; Netzwerk 1: A: Bandantrieb, vervollständigt



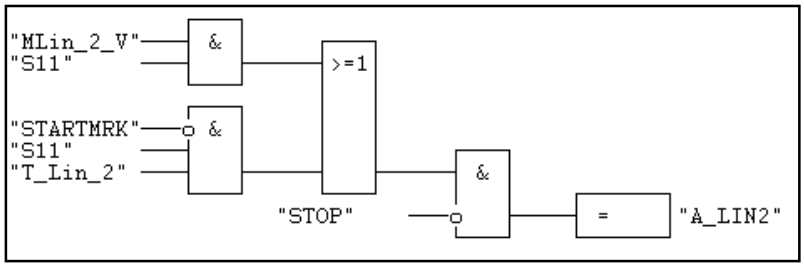
FC 6; Netzwerk 2: Paketgenerator



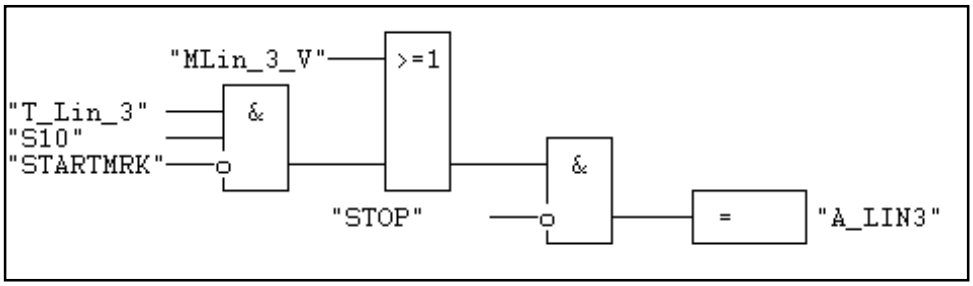
FC 6; Netzwerk 3: A: Schieber Lin_1 nach oben



FC 6; Netzwerk 4: A: 3 Pakete waagerecht verschieben. (Lin_2 nach rechts)



FC 6; Netzwerk 5: A: Palette leeren. (Lin_3 hochfahren)



vgl. TrySim-Projekt <Lösungen>|<Palette_FC>.

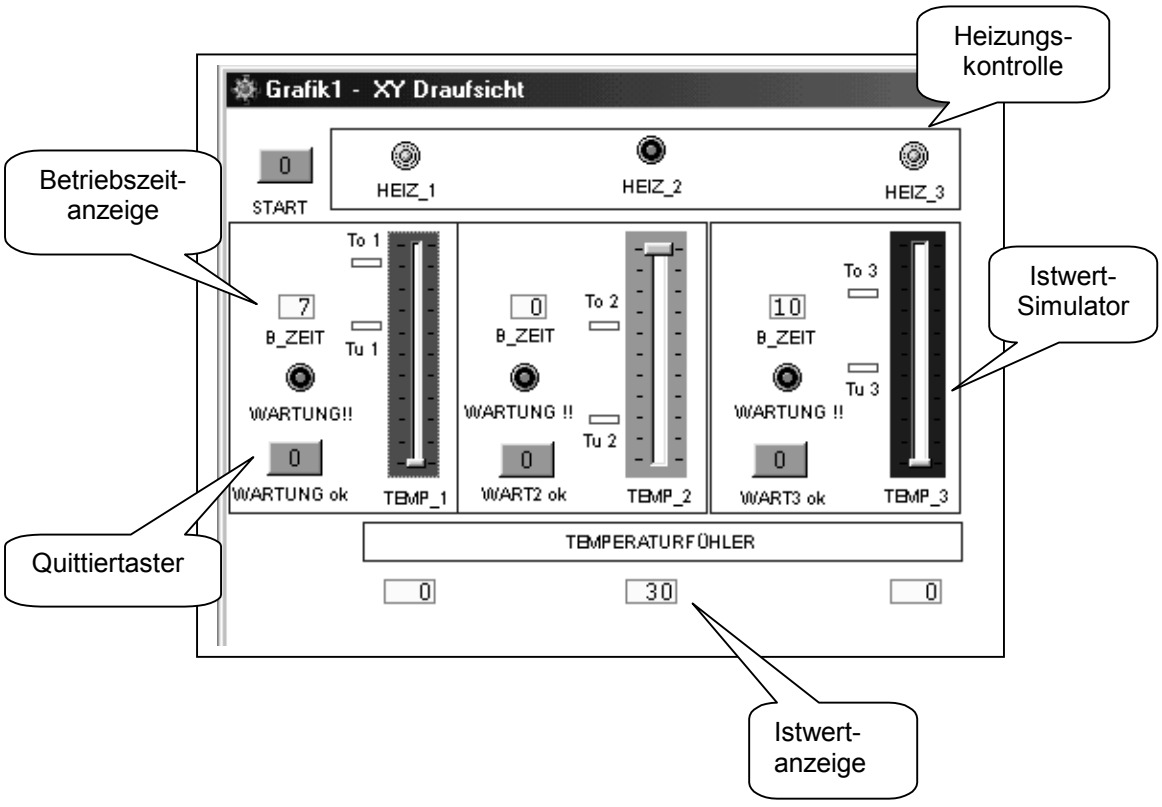
Projekt 31: Mehrfachaufruf einer Funktion mit DB-Abfrage

Schwerpunkte: FC, DB, Impuls, SPBN, <I, >I, +I, L, T.

Aufgabe: Eine SPS-Steuerung soll die Temperatur in drei Gewächshäusern regeln. In jedem der Gewächshäuser leben ganz unterschiedliche Pflanzen. Sie benötigen für optimale Lebensbedingungen unterschiedliche Temperaturbereiche. Wenn jeweils die untere "Wohlfühltemperatur" Tu unterschritten wird, schaltet die entsprechende Heizung ein, bis die obere Temperatur To erreicht ist.
Bei To wird die Heizung automatisch ausgeschaltet. Danach kühlt die Luft wieder ab, bis bei Tu der Zyklus neu beginnt.
Zusätzlich werden die Einschaltzeiten der Heizungen überwacht. Nach 30 "Einheiten" gibt es Blinkalarm. Nach der Wartung wird die Einschaltzeit auf Null zurückgesetzt.
Die Raumtemperaturen werden in diesem Beispiel per Hand mit den Schiebereglern simuliert.

Da die Rahmenbedingungen für alle drei Gewächshäuser gleich sind und sich nur die Parameter unterscheiden, bietet sich ein wiederholter Aufruf desselben Programms an. Das „eigentliche“ Programm wird in eine FC geschrieben. Dieser FC werden die unterschiedlichen Temperaturparameter zugewiesen und die entsprechenden Heizungen werden über die Aktualparameter geschaltet. Zur Erweiterung Ihrer Kenntnisse werden die Temperaturparameter in Datenbausteine geschrieben und von dort abgefragt.

Dieses Projekt ist bereits vorbereitet im Ordner <Vorlagen> unter <FC4_V>.



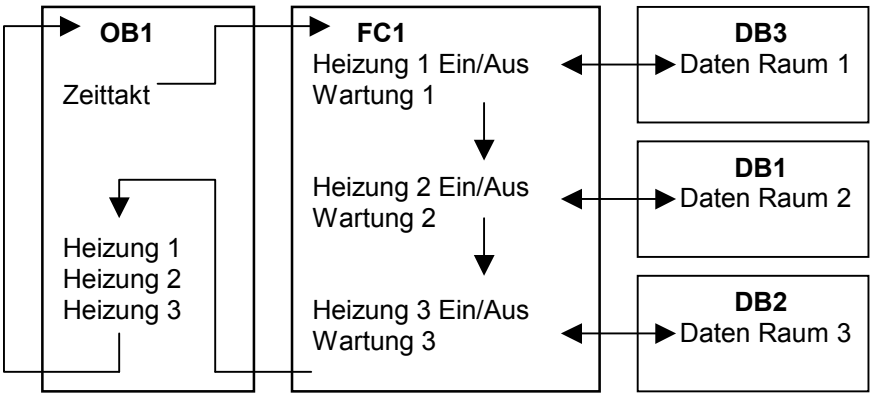
Symbolische Adresse	Operand	Datentyp	Kommentar
Heizung1	A 0.0	BOOL	Heizung Raum 1
Heizung2	A 0.1	BOOL	Heizung Raum 2
Heizung3	A 0.2	BOOL	Heizung Raum 3
Wart1LED	A 0.3	BOOL	LED "Wartung 1 ist fällig"
Wart2LED	A 0.4	BOOL	LED "Wartung 2 ist fällig"
Wart3LED	A 0.5	BOOL	LED "Wartung 3 ist fällig"
START	E 0.1	BOOL	START- / STOP-Schalter
WART1_ok	E 0.0	BOOL	Quittierung der Wartung Raum 1
WART2_ok	E 0.2	BOOL	Quittierung der Wartung Raum 2
WART3_ok	E 0.3	BOOL	Quittierung der Wartung Raum 3
Heiz1EIN	M 4.0	BOOL	Setzbefehl Heizung 1
Heiz1AUS	M 4.1	BOOL	Rücksetzen Heizung 1
Heiz2EIN	M 4.2	BOOL	Setzbefehl Heizung 2
Heiz2AUS	M 4.3	BOOL	Rücksetzen Heizung 2
Heiz3EIN	M 4.4	BOOL	Setzbefehl Heizung 3
Heiz3AUS	M 4.5	BOOL	Rücksetzen Heizung 3
Imp10S	M 5.0	BOOL	Alle 10 sec für einen Zyklus "1"
TEMP_1	EW 512	INT	Istwert Temperatur Raum 1
TEMP_2	EW 514	INT	Istwert Temperatur Raum 2
TEMP_3	EW 516	INT	Istwert Temperatur Raum 3
DB_Raum2	DB 1	DB 1	Datenbaustein für Raum 2
DB_Raum3	DB 2	DB 2	Datenbaustein für Raum 3
DB_Raum1	DB 3	DB 3	Datenbaustein Für Raum 1
T0OB1	T 0	TIMER	Zeitglied für Takterzeugung

Symboltabelle

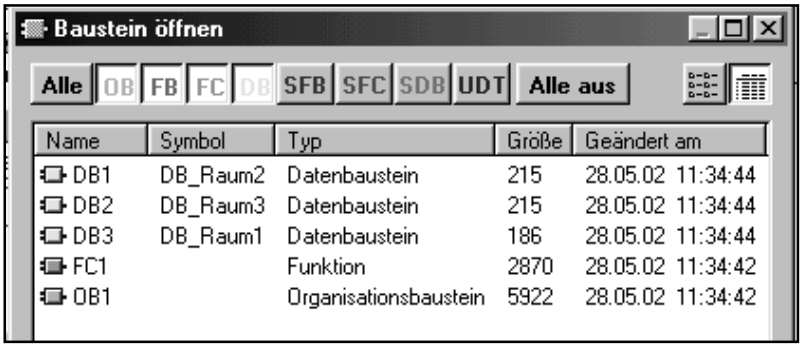
Reihenfolge der Aufrufe

Wie üblich startet das Programm automatisch den OB1. Von dort wird die FC aufgerufen, die wiederum die Daten aus Datenbausteinen abruft. Wie eine FC angelegt wird, wurde bereits im Projekt 23 erklärt. Bevor Sie das Programm der FC schreiben, legen Sie bitte in ähnlicher Weise die Datenbausteine an mit <SPS>|<NEU> und wählen im Fenster <Neuer Baustein> <DB>. Dieser Vorgang wird für jeden gewünschten DB wiederholt. Die Reihenfolge wird automatisch festgelegt. Welchen DB Sie welchem Gewächshaus zuordnen, ist allerdings beliebig zu wählen. Speichern Sie bitte diese Schritte, bevor Sie zur Programmierung übergehen.

Die Reihenfolge der Aufrufe soll dem folgenden, etwas vereinfachten Schema entsprechen.

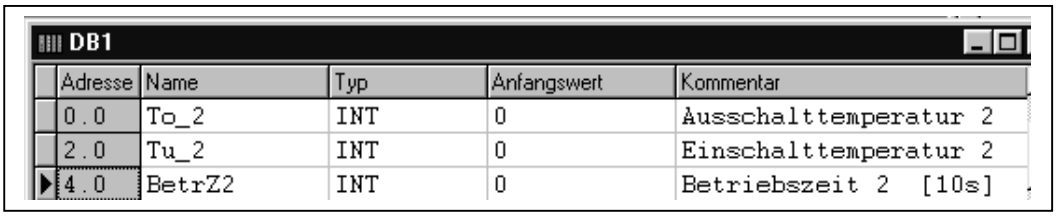


Mit <SPS>|<Öffnen> wird folgendes Fenster angezeigt. Sie können jeden Baustein einzeln markieren und öffnen - so, wie Sie es schon vom OB1 her kennen.

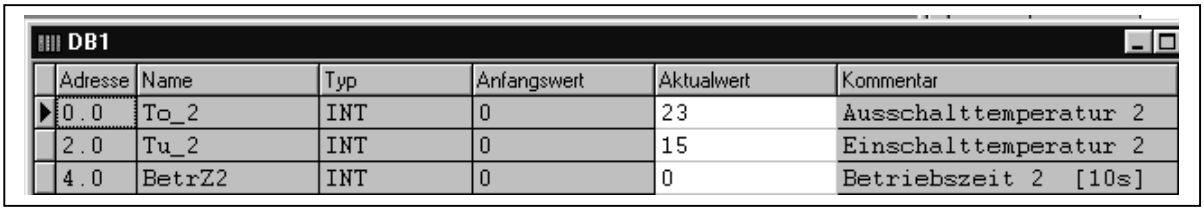


Damit das Programm der FC auf die Daten in den DBs zugreifen kann, müssen diese Daten zuerst in den DBs eingetragen werden. Sonst weist Sie TrySim schon beim Programmieren mit Symbolik darauf hin, dass die Daten fehlen.
Wenn der Cursor im rechten Feld (Kommentarfeld) steht, kann mit <Enter> eine neue Zeile geöffnet werden.

DB1:



Der DB1 wurde willkürlich dem zweiten Gewächshaus zugewiesen. „To“ soll für „obere Schalttemperatur“ stehen und „Tu“ für die niedrigste Temperatur. Der Typ „INT“ ist für die Zahlenwerte erforderlich. Der Anfangswert ist bedeutungslos und bleibt bei „0“.
Die beiden ersten Zeilen werden gleich Werte bekommen, die Sie als Programmierer in diesem Baustein vorgeben. Die dritte Zeile dient zur Aufnahme und Speicherung eines Wertes, den das Programm hierhin schreiben soll.
Zum Eintragen der Werte müssen Sie das entsprechende Fenster aktivieren und dann <Ansicht> und <Datenansicht> wählen. In die Spalte <Aktualwert> werden die vorgegebenen Werte eingetragen. Dort werden auch die hineingeschriebenen Daten angezeigt und gespeichert. Sie haben jetzt eine Möglichkeit, Daten aus der SPS zu laden, zu speichern und diese Daten auch nach einem erneuten Programmstart als neue Startwerte aufzurufen.



DB2:

DB2

	Adresse	Name	Typ	Anfangswert	Kommentar
▶	0.0	To_3	INT	0	Ausschalttemperatur 3
	2.0	Tu_3	INT	0	Einschalttemperatur 3
	4.0	BetrZ3	INT	0	Betriebszeit 3 [10s]

Dieser Baustein wird in gleicher Weise vorbereitet.

DB3:

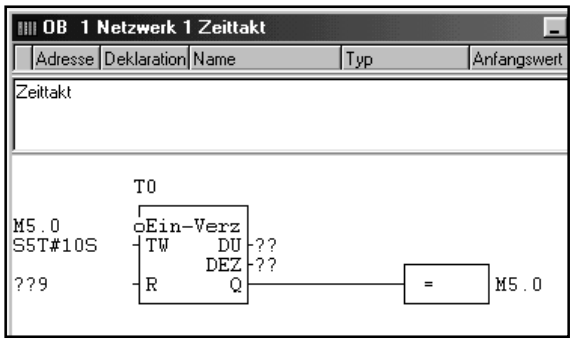
DB3

	Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
▶	0.0	res0	INT	0	0	Reserve
	2.0	res2	INT	0	0	Reserve
	4.0	BetrZ1	INT	0	20	Betriebszeit1 [10s]

Dieser Baustein wird in gleicher Weise vorbereitet. Obwohl die Betriebsstunden in Zeile 3 vom Programm geschrieben werden soll, kann man das auch „von Hand“ machen. „res0“ und „res2“ sind nicht belegt (siehe weiter unten) und nur aus Symmetriegründen genauso wie die anderen DBs angelegt.

Das folgende Netzwerk gilt quasi übergeordnet für alle Gewächshäuser, bzw. ist unabhängig von ihnen. Es gibt lediglich einen Impulstakt vor, aus dem später der Betriebsstundenzähler gespeist wird.

OB1, Netzwerk 1: Zeittakt



In einer Art „Selbstmordschaltung“ wird alle 10 Sekunden ein kurzer Impuls gebildet, der den Betriebszeitzählern den Takt gibt.

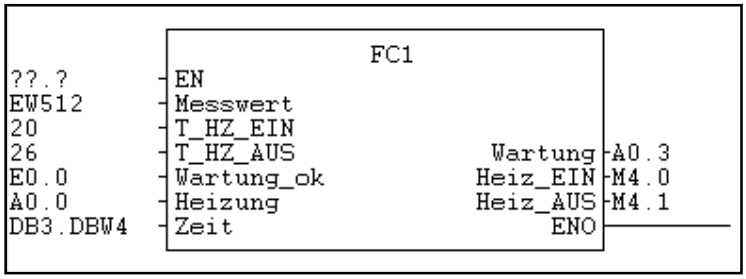
Deklaration FC1

FC 1 Netzwerk 2 Einschaltzeit

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
	0.0	in	Messwert	INT		Istwert [o C]
	2.0	in	T_HZ_EIN	INT		Temperatur zum Einschalten
	4.0	in	T_HZ_AUS	INT		Temperatur zum Ausschalten
	6.0	in	Wartung_ok	BOOL		Quittierung der Wartung
	6.1	in	Heizung	BOOL		Heizung ist eingeschaltet
	8.0	out	Wartung	BOOL		Wartung ist fällig
	8.1	out	Heiz_EIN	BOOL		Heizung einschalten
	8.2	out	Heiz_AUS	BOOL		Heizung ausschalten
	10.0	in_out	Zeit	INT		Betriebszeit [10 s / digit]
▶		temp				

Zu den bereits bekannten Deklarationen <in> und <out> kommt hier noch <in_out> hinzu. Das bedeutet, diese Daten können sowohl vom Aktualparameter geliefert werden, als auch von der Funktion an den Aktualparameter gesendet werden.
Konkret bedeutet das: Der alte Zählerstand wird gelesen, verarbeitet und modifiziert wieder an die Quelle geschickt.
Nach der Deklaration können Sie die FC in bekannter Weise aufrufen und die Aktualparameter zuweisen.

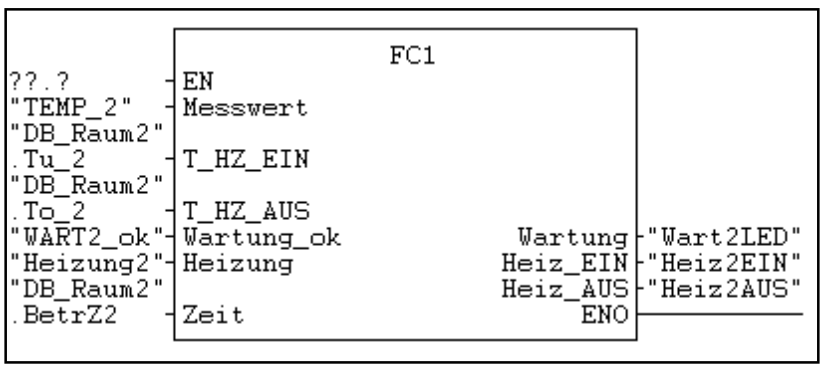
OB1, Netzwerk 2: Aufruf der Funktion FC1 für Heizung 1



Sie sehen hier vor „Zeit“ die Schreibweise des Zieles im DB3 ohne symbolische Darstellung.
Die Adresse ist in der dritten Zeile das Word 4, beginnend mit Bit 4.0.
Jede Zeile entspricht 16 Bit.
Folglich wird die Zeile benannt mit „DBW4“ für
„**D**aten**B**austein**W**ort **4**“.

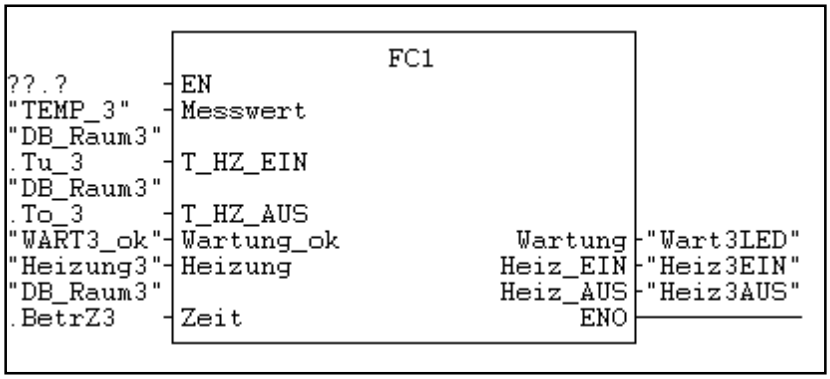
„EW 512“ liefert als Schieberegler den Istwert der Temperatur. „20“ und „26“ sind die Temperaturgrade, bei der die Heizung schalten soll. Diese Werte werden, um die Möglichkeiten bzw. Varianten zu zeigen, hier direkt als Konstanten eingetragen. „Heiz_EIN“ entscheidet für jeden Heizkreis, ob die Heizung eingeschaltet werden soll. Dieses Bit kann aber nicht speichernd von der FC ausgegeben werden, da die Bedingung für jede Heizung unterschiedlich sein kann. Deshalb werden in jedem Aufruf spezielle Aktualparameter verknüpft, die im OB1 dann die Heizung steuern.

OB1, Netzwerk 3: Aufruf der Funktion FC1 für Heizung 2



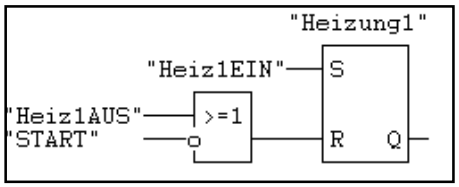
Symbolische Darstellung

OB1, Netzwerk 4: Aufruf der Funktion FC1 für Heizung 3



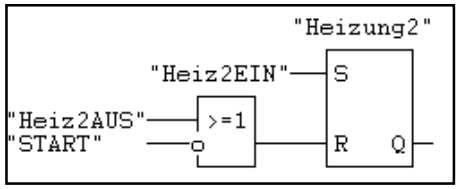
Rückkehr zum OB1

OB1, Netzwerk 1: Heizung 1

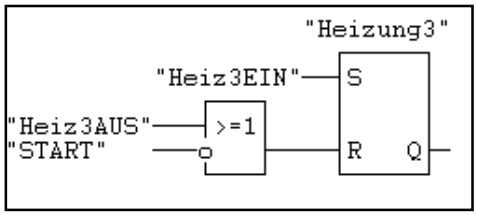


Das Programm wirkt als 2-Punkt-Regler. Mit dem Befehl „Heiz_EIN“ der FC1, der an den Aktualparameter „Heiz1EIN“ übergeben wurde, wird die Heizung 1 eingeschaltet, bis mit „Heiz1AUS“ wieder ausgeschaltet wird. Ebenso wird ausgeschaltet, wenn der <START>-Schalter eine „0“ liefert.

OB1, Netzwerk 1: Heizung 2



OB1, Netzwerk 1: Heizung 3



Wechsel zur FC1

FC1, Netzwerk 1: Ein- u. Ausschaltpunkt

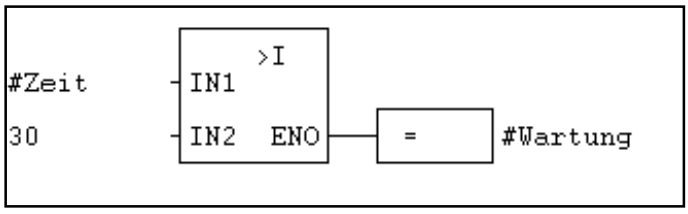
Jetzt muss noch geklärt werden, unter welchen Bedingungen die Heizungen ein- bzw. ausgeschaltet werden sollen. Das eigentliche Programm steht in der Funktion FC1. Dafür wurden oben bereits die Formalparameter deklariert, die hier – intern in der FC – im Programm verknüpft werden.

```
    L  #Messwert        //Lade den augenblicklich geltenden Istwert (Aktualparameter)
    L  #T_HZ_EIN        //Lade den augenblicklich geltenden unteren Schaltwert (Aktualpar..)
    <I                  //Wenn der Istwert kleiner als der Tu ist...
    =  #Heiz_EIN        //...wird die zugehörige Heizung eingeschaltet.
    L  #Messwert        //Lade den augenblicklich geltenden Istwert (Aktualparameter)
    L  #T_HZ_AUS        //Lade den augenblicklich geltenden oberen Schaltwert (Aktualpar.)
    >I                  //Wenn der Istwert größer als der To ist...
    =  #Heiz_AUS        //...wird die zugehörige Heizung ausgeschaltet.
```

FC1, Netzwerk 2: Einschaltzeit

```
    U  "Imp10S"     //Immer wenn im OB1, Netzwerk1 ein Impuls gebildet wird…
    U  "START"      //… der <START>-Schalter "1" liefert...
    U  #Heizung     // und die jeweilige Heizung eingeschaltet ist...
    SPBN  end4      // (falls nicht, springe zur Marke <end4:>)
    L  #Zeit        // (sonst) lade den Wert des entsprechenden Betriebsstundenzählers
    L  1            // Lade den Wert „1"
    +I              // addiere die beiden Werte, d.h. erhöhe den Wert um 1
    T  #Zeit        // Transferiere den neuen Wert in den entsprechenden DB.
end4: NOP 0         // Durch den Sprung wird kein neuer Wert eingetragen.
```

FC1, Netzwerk 3: Wartung fällig?



Wenn die ermittelte und gespeicherte Zeit des DBs größer als 30 Einheiten ist, wird die Wartungs-LED eingeschaltet.

FC1, Netzwerk 4: Rücksetzen des Zählers nach Wartung

```
    U  #Wartung_ok  // Wenn der Taster betätigt wird...
    SPBN  end6      // (falls nicht, springe zur Marke <end6:>)
    L  0            // (sonst) lade den Wert „0" und ...
    T  #Zeit        // ...Transferiere ihn zum entsprechenden DB.
end6: NOP 0         // Durch den Sprung wird kein neuer Wert eingetragen.
```

Zur Erinnerung: Durch den Befehl „NOP 0" wird gar nichts gemacht, es wird keine Operation ausgeführt. Die Programmsystematik erfordert nur in jeder Zeile irgendeine Anweisung.

Vgl. TrySim-Projekt <FC4> im Verzeichnis <Lösungen>.

Sie haben gelernt:

- Das Programm einer FC kann über verschiedene Aktualparameter unterschiedliche, aber gleich strukturierte Aufgaben bearbeiten.
- Daten können aus einem Datenbaustein abgefragt werden.
- Daten können in einen Datenbaustein transferiert werden.
- Daten können in einem Datenbaustein gespeichert werden.

Projekt 32: Funktionsbausteine FB

Schwerpunkte: FB, Instanz-DB

Aufgabe: Diese Problemstellung entspricht exakt der aus dem vorherigen Projekt 26.
Das Programm soll jedoch mit Funktionsbausteinen erstellt werden.

Theoretische Erörterung:

Funktionsbausteine gleichen in vielem den Funktionen. Der Unterschied zwischen diesen beiden besteht darin, dass beim Aufruf eines Funktionsbausteines (fast) immer ein Datenbaustein angegeben wird, in dem die Daten stehen, mit denen der Baustein arbeiten soll. Dieser Datenbaustein wird Instanz-Datenbaustein genannt. Seine Struktur wird bei der Programmierung des FB's festgelegt, und auf seine Daten kann innerhalb des FB's besonders komfortabel zugegriffen werden.

Dieses Buch will nicht alle Erläuterungen des TrySim-Programms wiederholen.
Deshalb: Vergleichen Sie bitte hierzu die TrySim-Hilfe unter <Hilfe>|<Index>| „FB".

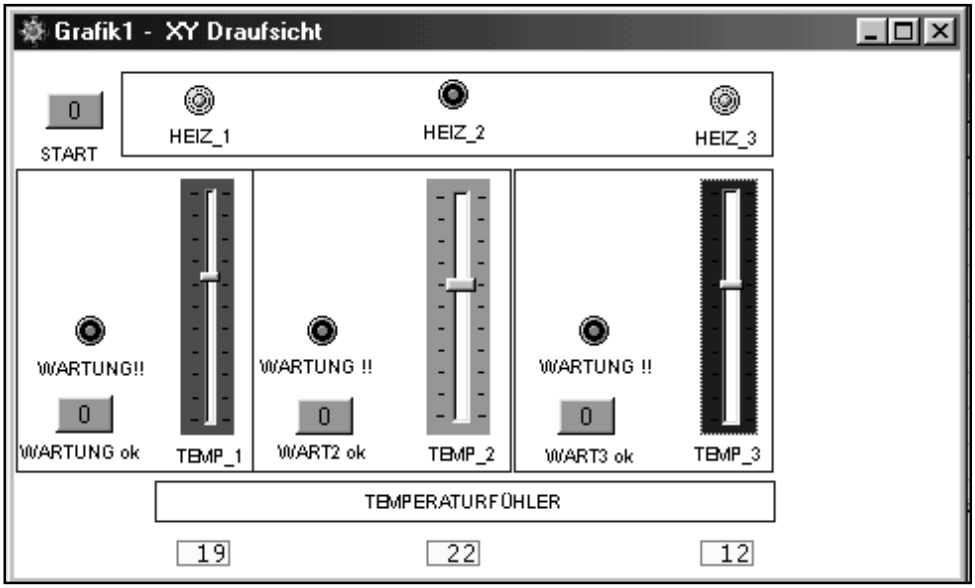
Durchführung des Projektes mit TrySim:

Im Verzeichnis <Vorlagen> ist das Projekt unter **<FB_V>** vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

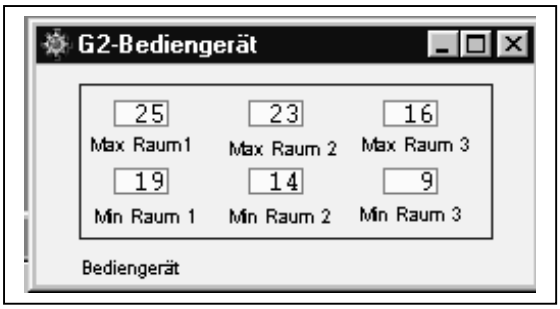
Die folgende Symboltabelle ist bereits vorbereitet und schon aus dem vorherigen Projekt bekannt.

Symbolische Adresse	Operand	Datentyp	Kommentar
Heizung1	A 0.0	BOOL	Heizung Raum 1
Heizung2	A 0.1	BOOL	Heizung Raum 2
Heizung3	A 0.2	BOOL	Heizung Raum 3
Wart1LED	A 0.3	BOOL	LED "Wartung 1 ist fällig"
Wart2LED	A 0.4	BOOL	LED "Wartung 2 ist fällig"
Wart3LED	A 0.5	BOOL	LED "Wartung 3 ist fällig"
DB_Raum2	DB 1	FB 2	Datenbaustein für Raum 2
DB_Raum3	DB 2	FB 2	Datenbaustein für Raum 3
DB_Raum1	DB 3	FB 2	Datenbaustein für Raum 1
WART_ok1	E 0.0	BOOL	Taster Wartung Raum 1 ok
WART_ok2	E 0.2	BOOL	Taster Wartung Raum 2 ok
WART_ok3	E 0.3	BOOL	Taster Wartung Raum 3 ok
START	E 0.1	BOOL	START- / STOP-Schalter
TEMP_1	EW 512	INT	Istwert Temperatur Raum 1
TEMP_2	EW 514	INT	Istwert Temperatur Raum 2
TEMP_3	EW 516	INT	Istwert Temperatur Raum 3
Heiz1EIN	M 4.0	BOOL	Setzbefehl Heizung 1
Heiz1AUS	M 4.1	BOOL	Rücksetzen Heizung 1
Heiz2EIN	M 4.2	BOOL	Setzbefehl Heizung 2
Heiz2AUS	M 4.3	BOOL	Rücksetzen Heizung 2
Heiz3EIN	M 4.4	BOOL	Setzbefehl Heizung 3
Heiz3AUS	M 4.5	BOOL	Rücksetzen Heizung 3
Imp10S	M 5.0	BOOL	Alle 10 sec für einen Zyklus "1"
T0OB1	T 0	TIMER	Zeitglied für Takterzeugung

Lassen Sie sich bitte nicht von der vermeintlichen Fehleranzeige beim Laden von <FB_V>. Klicken Sie so lange auf <Nein>, bis das Projekt angezeigt wird.
Erklärung: Im „Bediengerät" wurden schon Adressen in den Datenbausteinen zugeordnet. Diese Datenbausteine sind aber noch gar nicht vorhanden, sondern werden erst von Ihnen erstellt.



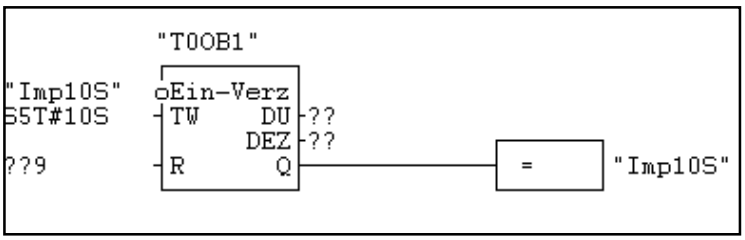
Zur bereits bekannten Anlage kommt ein zweites Grafikfenster hinzu:



Bediengerät:
Einstellung der Schalttemperaturen

Hier können Sie die Grenzwerte einstellen!

OB 1; Netzwerk 1: Zeittakt



In diesem Netzwerk wird alle 10 s ein Impuls in einer Art „Selbstmordschaltung“ erzeugt. Solange der Ausgang „0“ führt, liegt am Eingang eine „1“ an. Nach Ablauf der Zeit kippt am Ausgang und somit auch am Eingang der Pegel um, so dass im nächsten Zyklus <T0OB1> wieder ausgeschaltet wird. Diese Impulse werden zum Zählen für die „Betriebsstunden“ dienen.

Bevor im nächsten Netzwerk des OB 1 der Funktionsbaustein FB 2 aufgerufen werden kann, muss der FB angelegt und deklariert sein. (<FB 2> wurde willkürlich gewählt, um zu zeigen, dass es keine feste Reihenfolge geben muss).
In diesem Baustein wird das eigentlichen Programm geschrieben. Es gilt für jedes Gewächshaus, indem die individuellen Parameter jeweils zugeordnet werden. Es gibt also nur **ein** Programm.
Die Aufrufe des Programms erfolgen aus dem OB 1. Die zugehörigen Daten werden in entsprechende Instanz-DBs geschrieben, bzw. daraus gelesen.

FB 2

Erzeugen Sie in der gewohnten Weise einen neuen Baustein, nämlich den FB 2. Deklarieren Sie die Formalparameter mit Name, Typ und Kommentar, wie Sie das bei der Erzeugung von FCs gelernt haben. Die bei FBs neuen <stat> Variablen, die gleich erklärt werden, können Sie genau wie <in> und <out> Parameter erzeugen. Ihr FB-Kopf sollte am Ende so aussehen:

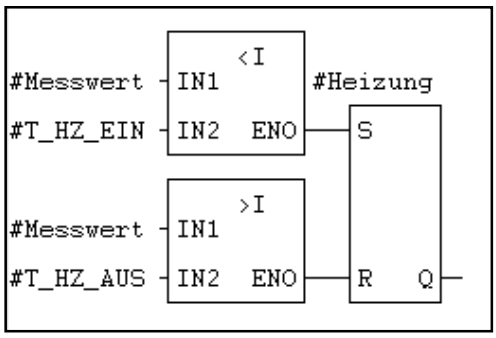
FB 2; Deklaration

FB 2 Netzwerk 1 Ein- u. Ausschaltpunkt

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Messwert	INT	0	Ist-Temperatur in Grad C
2.0	in	Wartung_ok	BOOL	FALSE	
4.0	out	Heizung	BOOL	FALSE	Heizung anschalten!
4.1	out	Wartung	BOOL	FALSE	Heizung muss gewartet werden!
	in_out				
6.0	stat	T_HZ_EIN	INT	0	Min-Temperatur, wird über Bediengerät gesetzt
8.0	stat	T_HZ_AUS	INT	0	Max-Temperatur, wird über Bediengerät gesetzt
10.0	stat	Zeit	INT	0	Betriebs"stunden" zähler der Heizung
	temp				

Die Anfangswerte müssen Sie nicht eintragen. Die zugehörigen Daten werden in die Instanz-DBs geschrieben, die automatisch erstellt werden. Das wird später erfolgen.
Zur Speicherung von Ergebnissen, die innerhalb eines FBs (nicht OB oder FC) von einem Zyklus zum nächsten ihren Wert beibehalten, können Sie statische Variablen deklarieren. Das gilt für die oben dargestellten Größen unter <stat>..

FB 2; Netzwerk 1: Ein- u. Ausschaltpunkt



Wenn der <#Messwert> kleiner ist als die minimal zulässige Temperatur, wird <# Heizung> eingeschaltet. Der zuständige Messwert wird über den Aktual-parameter im OB 1 zugeordnet.
Die zuständige Heizung wird auch über den Aktual-parameter im OB 1 zugeordnet.
Wenn der <#Messwert> größer ist als die maximal zulässige Temperatur, wird <# Heizung> ausgeschaltet.

FB 2; Netzwerk 2: Einschaltzeit

Hier werden die Einschaltzeiten der Gewächshaus-Heizungen programmiert.

```
      U  "Imp10S"      // Immer, wenn ein Zählimpuls geliefert wird
      U  "START"       // und <START> eine „1“ liefert
      U  #Heizung      // und die zugehörige Heizung eingeschaltet ist
      SPBN  end4       // (falls nicht, springe zur Sprungmarke <end4:>)
      L   #Zeit        // falls vorstehenden Bedingungen erfüllt sind, lade die bisherige Zeit
      L   1            // Lade die Zahl 1
      +I               // addiere beide Werte
      T   #Zeit        // und transferiere den neuen Wert in den zugehörigen IDB.
end4: NOP 0            // Diese Sprungmarke dient nur zum Überspringen der
                       // vorausgehenden Zeilen.
```

FB 2; Netzwerk 3: Wartung fällig?

Eine Wartung ist erforderlich, wenn der „Kontostand“ im IDB den Wert „31“ anzeigt.

```
L   #Zeit          // Lade die entsprechende, gespeicherte Zeit aus dem IDB.
L   30             // Lade die Zahl 30
>I                 // Wenn #Zeit > 30 ist...
=  #Wartung        // schalte die entsprechende LED „Wartung !!“ ein
```

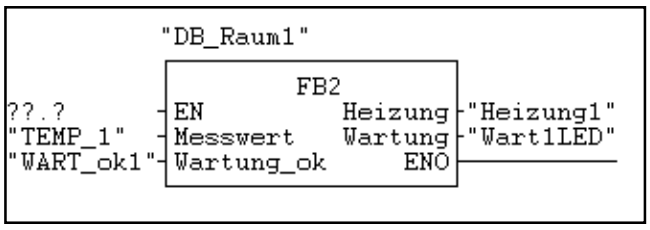
FB 2; Netzwerk 4: Rücksetzen nach Wartung

```
    U  #Wartung_ok     // Nachdem die Wartung durchgeführt wurde...
    SPBN  end6         // (falls nicht, springe zu <end6:>
    L   0              // Lade die Zahl 0
    T   #Zeit          // und transferiere sie an <#Zeit>, so dass das entsprechende
                       // Zeitkonto im IDB auf 0 gesetzt wird.
end6: NOP 0            // Diese Sprungmarke dient nur zum Überspringen der
                       // vorausgehenden Zeilen.
```

Weiter im OB 1

OB 1; Netzwerk 2: Aufruf des Funktionsbausteins FB 2 für Heizung 1

Rufen Sie jetzt unter dem Icon <FUP-Elemente>|<FB> auf. Es wird nur <FB 2> eingeblendet, da kein anderer vorhanden ist. Markieren Sie den Baustein, drücken <OK> und ergänzen Sie die Aktualparameter für den ersten Raum. In den Kopf schreiben Sie „**DB3**“. Angezeigt wird - wenn Sie <Ansicht>|<symbolische Darstellung> gewählt haben - <DB_Raum1>, weil diese Zeile in der Symboltabelle schon vorbereitet wurde. Die Zuordnung zum <FB 2> wird in der Symboltabelle unter <Datentyp> vermerkt. Unter <SPS> <Öffnen> sehen Sie, dass allein durch den Aufruf der DB3 automatisch angelegt wurde. Wenn Sie ihn öffnen, werden Sie feststellen, dass die Deklarationsdaten des zugehörigen FB 2 übernommen wurden. Sie können jetzt **dort** die Aktualwerte eintragen, bzw. für diese Aufgabe im Bediengerät (Grafikfenster 2) die entsprechenden Operanden.
Hinweis: Die Digitalanzeigen bei TrySim können durch Doppelklick während der Simulationslaufzeit auch als Quelle benutzt werden. Die Werte können „von Hand“ umgeschrieben werden. Wenn die Zieladresse in einem DB ist, wird dort der Wert auch geändert.



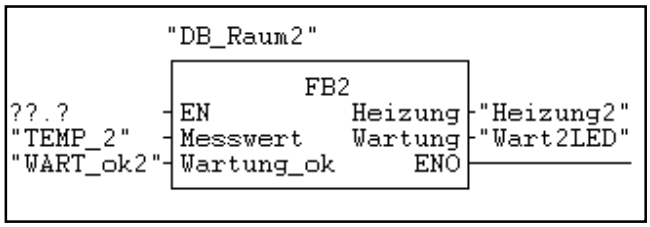
Instanzdatenbaustein (I)DB 3

DB3

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	Messwert	INT	0	19	Ist-Temperatur in Grad C
2.0	Wartung_ok	BOOL	FALSE	FALSE	
4.0	Heizung	BOOL	FALSE	TRUE	Heizung anschalten!
4.1	Wartung	BOOL	FALSE	FALSE	Heizung muss gewartet werden!
6.0	T_HZ_EIN	INT	0	19	Min-Temperatur, wird über Bediengerät gesetzt
8.0	T_HZ_AUS	INT	0	25	Max-Temperatur, wird über Bediengerät gesetzt
10.0	Zeit	INT	0	13	Betriebs"stunden" zähler der Heizung

OB 1; Netzwerk 3: Aufruf des Funktionsbausteins FB2 für Heizung 2

Der zugehörige Instanz-DB für den zweiten Raum wird in gleicher Weise erstellt. Tragen Sie dazu den nächsten DB im Kopf ein: „DB**1**“. Die Zuordnungen wurden in der Symboltabelle willkürlich gewählt, damit klar wird, dass hier keine bestimmte Reihenfolge eingehalten werden muss.



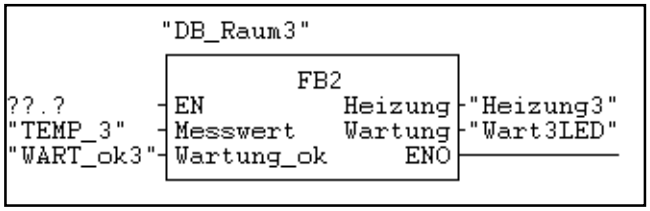
Instanzdatenbaustein (I)DB 1

DB1

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	Messwert	INT	0	12	Ist-Temperatur in Grad C
2.0	Wartung_ok	BOOL	FALSE	FALSE	
4.0	Heizung	BOOL	FALSE	TRUE	Heizung anschalten!
4.1	Wartung	BOOL	FALSE	FALSE	Heizung muss gewartet werden!
6.0	T_HZ_EIN	INT	0	14	Min-Temperatur, wird über Bediengerät gesetzt
8.0	T_HZ_AUS	INT	0	23	Max-Temperatur, wird über Bediengerät gesetzt
10.0	Zeit	INT	0	13	Betriebs"stunden" zähler der Heizung

OB 1; Netzwerk 4: Aufruf des Funktionsbausteins FB2 für Heizung 3

Der zugehörige Instanz-DB für den dritten Raum wird in gleicher Weise erstellt. Tragen Sie dazu den nächsten DB im Kopf ein: „DB**2**“.

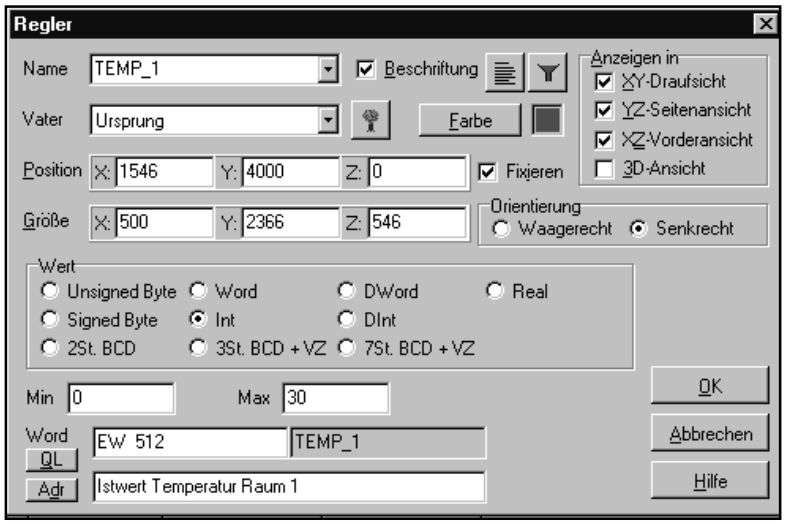


Instanzdatenbaustein (I)DB 2

DB2

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	Messwert	INT	0	0	Ist-Temperatur in Grad C
2.0	Wartung_ok	BOOL	FALSE	FALSE	
4.0	Heizung	BOOL	FALSE	TRUE	Heizung anschalten!
4.1	Wartung	BOOL	FALSE	FALSE	Heizung muss gewartet werden!
6.0	T_HZ_EIN	INT	0	9	Min-Temperatur, wird über Bediengerät gesetzt
8.0	T_HZ_AUS	INT	0	16	Max-Temperatur, wird über Bediengerät gesetzt
10.0	Zeit	INT	0	13	Betriebs"stunden" zähler der Heizung

Die Temperatur-Istwerte werden in dem Grafikfenster 1 durch die Schieberegler nachgebildet und darunter angezeigt. Diese Wörter (hier: EW 512 für Raum 1) werden als symbolische Adressen (Aktualparameter) in den OB 1-Netzwerken an die FB 2-Bausteine gelegt.



Die Min.- / Max.Bereiche können in diesen Editierfenstern geändert werden.

vgl. TrySim-Projekt <FB> im Verzeichnis <Lösungen>.

Sie können jetzt:

- Einen FB erstellen und deklarieren
- Ein Programm in einem FB schreiben
- Einen FB vom OB 1 aus aufrufen
- Im OB1-Aufruf dem FB einen Instanz-DB zuordnen
- Diesen Instanz-DB öffnen und Aktualdaten anlegen
- Bei TrySim: Änderungen der Aktualdaten im Instanz-DB beobachten

Projekt 33: Teilsummen- / Mittelwertbildung

Schwerpunkte: Addition +I, Pointer P#..., Lade L, Transfer T in DB, Flankensteuerung FP, Sprünge, speicherindirekte Adressierung, Schleife LOOP, Zähler Z, Vergleich >I, Dividieren /I, FC, Schiebeoperation SLW, Verwendung von breakpoint.

Aufgabe: Teilsummen- /Mittelwertbildung

In einem Datenbaustein sind 10 Werte gespeichert (z.B. Messwerte, die in den DB geschrieben wurden).

Es soll möglich sein, verschieden viele (aufeinander folgende !) Werte zu addieren und daraus den Mittelwert zu bilden. Der Startwert der Messreihe, die zu berechnen ist, kann ebenfalls frei bestimmt werden.
Auch Einzelabfragen sind möglich.

Es ist dafür zu sorgen, dass nur im richtigen Datenbereich (also nicht über die Zeile 10 hinaus) die Daten erfasst werden. Das Speicherwort für die Summenbildung wird "von Hand" oder bei jeder Verstellung der Vorwahl (Änderung des Messbereiches) auf "0" gesetzt.
Bei einer Mehrfachbetätigung oder Dauerdruck auf <BERECHNEN> darf sich die Anzeige nicht ändern.

Theorie:
In modernen Steuerung- und Regelungsanlagen ist es unerlässlich, Messwerte anzuzeigen, zu speichern und zu protokollieren. Als "Speichermedium" der SPS dient der Datenbaustein DB. Hier können wortweise Daten abgelegt und gelesen werden. Normalerweise besteht ein Programm aus der Verknüpfung von fest zugewiesenen Operanden. Was aber, wenn man aus einer Tabelle verschiedene Werte aufrufen möchte?
Dann müsste man für alle Alternativen die entsprechenden Programmzeilen schreiben und über Sprungfunktionen zu genau der gewünschten - und programmierten - Varianten springen.
Es geht aber erfreulicherweise auch anders:
Es gibt Befehle / Operationen, die es erlauben, in einem festen Grundkonzept mit variablen Operanden zu arbeiten. **Ausgehend von einer gewählten Bezugsadresse** - z.B. der ersten Position in einem Datenbaustein, beschrieben als "DBW0" - **können die anderen Werte als Abstand dazu beschrieben werden.**
In diesem kleinen Beispiel benötigt man fast keine "normalen" Verknüpfungen, die von der reinen Steuerungstechnik her bekannt sind, sondern Sie lernen eine Vielzahl weiterer SPS-Operationen kennen, die erst die Grundlage für ein Automatisierungsgerät bilden.

Zur Praxis mit TrySim:

Im Verzeichnis <Vorlagen> ist das Projekt unter <LOOP_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

DB1

	Adresse	Name	Typ	Anfangswert	Kommentar
	0.0	WertNr_0	INT	10	Wert 0 Byte 0
	2.0	WertNr_1	INT	20	Wert 1 Byte 2
	4.0	WertNr_2	INT	30	Wert 2 Byte 4
	6.0	WertNr_3	INT	5	Wert 3 Byte 6
	8.0	WertNr_4	INT	50	Wert 4 Byte 8
	10.0	WertNr_5	INT	5	Wert 5 Byte 10
	12.0	WertNr_6	INT	3	Wert 6 Byte 12
	14.0	WertNr_7	INT	2	Wert 7 Byte 14
	16.0	WertNr_8	INT	6	Wert 8 Byte 16
	18.0	WertNr_9	INT	4	Wert 9 Byte 18
▶	20.0	SUMME	INT	0	(Teil)-Summe

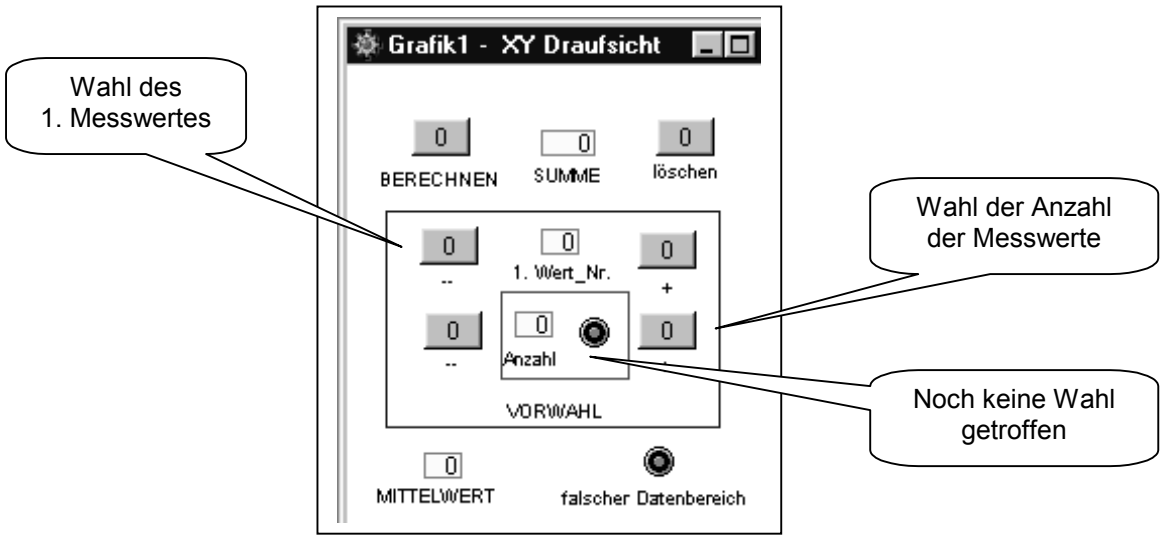
Datenbaustein

Zuerst kann der Datenbaustein erstellt werden unter <SPS> <NEU> <DB> bzw. - falls schon vorhanden - unter <SPS> <Öffnen> <DB...> (markieren) und mit <OK> auswählen. Bei aktivem DB-Fenster kann unter <Ansicht> <Deklarationsansicht> in die erste Zeile ein Name eingetragen werden, unter dem die "Zeile"- das Wort- später aufgerufen werden kann.
Wichtig ist der Eintrag des Datentyps (hier: INT). Sie können dann Anfangswerte eintragen, mit denen das Programm starten soll, und die später vom Programm auch geändert werden können. In obigem Beispiel sind links die Wortadressen zu erkennen. Sie können das Wort der Zeile 3 (Adresse 4.0) z.B. mit „L DB 1.DBW4" auslesen, oder für dieselbe Zeile auch „L DB 1.WertNr_2" eingeben. Der Name muss dabei exakt so geschrieben werden, wie in der DB-Tabelle eingetragen.
Gekennzeichnet ist die Adresse in der Tabelle des DB 1 jeweils durch das erste Bit, für das DBW4 also durch „4.0“.

Symboltabelle

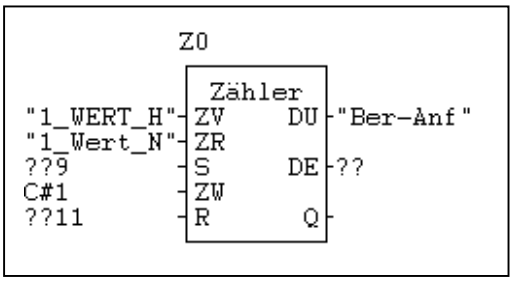
Symbolische Adresse	Operand	Datentyp	Kommentar
Ber-Anf	AW 512	INT	Bereichs-Anfang, 1. Wert_Nr.
WIEVIEL	AW 514	INT	Bereichs-LÄNGE (wieviel Werte)
MITTELW	AW 516	INT	Mittelwertanzeige
STELLZHL	MW 30	INT	Kontrolle der Wortstellen im DB1
Berechne	E 0.0	BOOL	Beginn der Rechenoperation
löschen	E 0.1	BOOL	Teil-Summe im DB 1 löschen
MESSW_W	E 0.4	BOOL	weniger Messwerte
MESSW_M	E 0.5	BOOL	mehr Messwerte
1_WERT_H	E 0.3	BOOL	1. Wert höher
1_Wert_N	E 0.2	BOOL	1. Wert niedriger
Überlauf	A 0.0	BOOL	falscher Wort-Bereich
NULLWAHL	A 0.1	BOOL	"Anzahl" >0 wählen!

Anlage



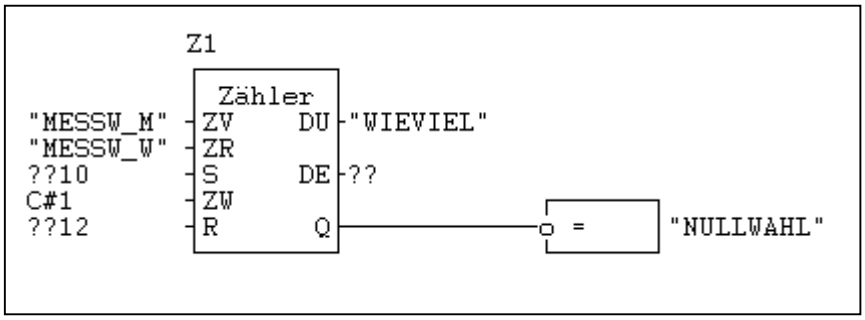
TrySim gestattet es, ein DB-Wort direkt auf die Anzeige zu legen. Das wird bei der Anzeige von <SUMME> angewendet, deren Zielbezeichnung direkt „DB1.DBW20“ ist. Im Normalfall müsste dafür natürlich ein "AW..." zwischengeschaltet werden, falls eine Anzeige außerhalb des Automatisierungsgerätes gewünscht würde.

OB 1; Netzwerk 1: Vorwahl des ersten Messwertes (Tabellen- /Zeilen-Nr.)



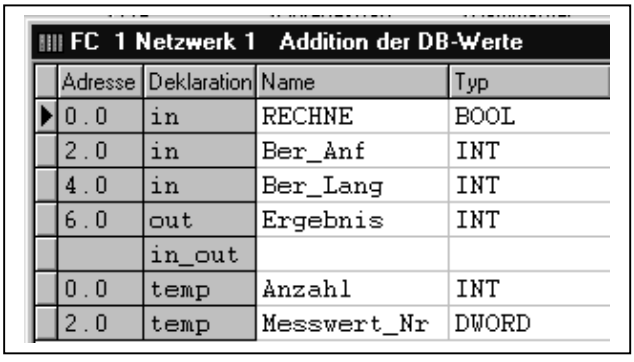
Durch Tasterbetätigung wird der Zählerwert verändert. Der Wert wird an die Digital-Anzeige geleitet.

OB 1; Netzwerk 2: Wie viele Messwerte sollen erfasst werden?



Durch Tasterbetätigung wird der Zählerwert verändert. Der Wert wird an die Digital-Anzeige geleitet.
Wenn **kein** Wert (= 0) gewählt wird, führt <NULLWAHL> "1". Die Hinweis-LED blinkt dann. Es soll dann kein BERECHNEN möglich sein (siehe FC 1).
Bevor das nun folgende Netzwerk erstellt werden kann, muss die FC 1 – zumindest mit den Kopfdaten / Deklarationen – angelegt worden sein mit <SPS>|<NEU>|<FC>|<OK> und <Projekt>|<Alles speichern>.
Anschließend können Sie die FC 1 aufrufen unter <SPS>|<Öffnen>|<FC1> (markieren) und <OK>.

Wechsel zu FC 1

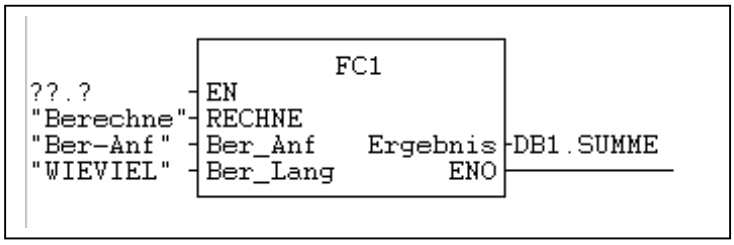

FC 1 Netzwerk 1 Addition der DB-Werte

	Adresse	Deklaration	Name	Typ
▶	0.0	in	RECHNE	BOOL
	2.0	in	Ber_Anf	INT
	4.0	in	Ber_Lang	INT
	6.0	out	Ergebnis	INT
		in_out		
	0.0	temp	Anzahl	INT
	2.0	temp	Messwert_Nr	DWORD

Die FC-Adressen können mit ihren Namen angesprochen werden (siehe unten OB 1, Netzwerk 3). Bei jedem Aufruf einer FC (z.B. im OB 1) müssen die "Anschlüsse" der restlichen Steuerung übergeben werden.
Siehe hierzu auch unter TrySim-<Hilfe>|<Index> "FC".

Nachdem Sie eine Zeile ausgefüllt haben, können Sie weitere Parameter des gleichen Typs mit der <ENTER>-Taster öffnen. Unter <temp> werden die Variablen eingetragen, die nur innerhalb der FC verändert werden. Diese haben also keine Ein- oder Ausgänge am FC-Block.

OB 1; Netzwerk 3: Aufruf des Rechenprogramms

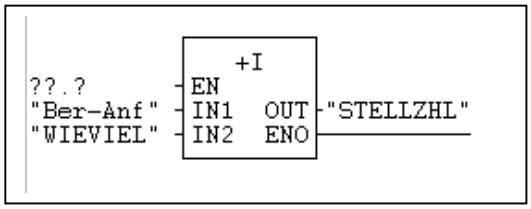



```
U  "Berechne"
CALL FC 1
RECHNE   := "Berechne"
Ber_Anf  := "Ber-Anf"
Ber_Lang := "WIEVIEL"
Ergebnis := DB  1.SUMME
```

Netzwerk in FUP.... ...und AWL

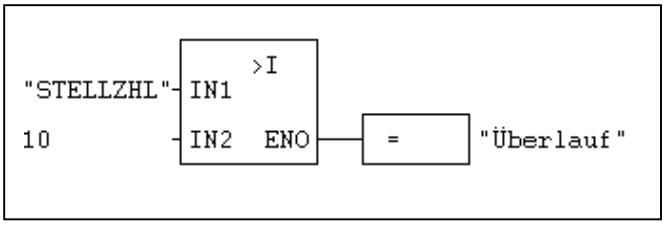
Wenn FC 1 im FUP erstellt wurde und dann in AWL umgeschaltet wird, sieht der Text etwas anders aus. Die zusätzlichen Zeilen entstehen automatisch durch das Programm und haben keinen Einfluss auf die Logik. Sie können in AWL die Zeilen auch wieder wie hier abgebildet ändern und auch dann wieder in FUP umschalten. Seien Sie bitte nicht irritiert; dieses Phänomen ist bedeutungslos.

OB 1; Netzwerk 4: Addition der 1. Stelle und der Anzahl (MAX.-Wert = 9)



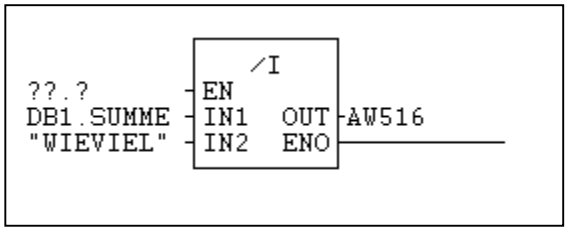
Wichtig ist die Kontrolle der letzten aufzurufenden Zeile im DB. Sie muss im gewünschten Datenbereich liegen. Es sind insgesamt 10 Einträge abfragbar, d.h. je nach Startposition (z.B. WertNr_6) könnten noch 3 weitere Messwerte (also insgesamt 4 Messwerte) erfasst werden. Danach kommt unter DB 1.DBW20 die Summe, die von der FC 1 als Ergebnis dort eingetragen wird. Dieser Wert soll aber laut Arbeitsanweisung bestimmt nicht für die Mittelwertsbildung verwendet werden.

OB 1; Netzwerk 5: Kontrolle der richtigen Bytes



Es wird verglichen, ob die oben errechnete Zeilenzahl (STELLZHL) größer ist als die Zahl 10. Dieser Fall würde durch eine Blink-LED ("Überlauf") signalisiert. In FC 1 ist dann keine Berechnung möglich (vgl. dort).

OB 1; Netzwerk 6: Mittelwert



Die im DB 1 abgelegte Summe wird durch die gewählte Anzahl der Messungen geteilt. Dieser Mittelwert wird über AW 516 ("MITTELW") an die Digitalanzeige gelegt.

Das eigentliche Programm wird in der FC erstellt.
Durch Tastendruck auf <BERECHNEN> werden die jeweiligen DB-Werte mittels „LOOP“ addiert und an den DB bzw. die Anzeige geleitet.

FC1; Netzwerk 1:

Grundsätzlich kann ein Programm in FUP, KOP oder AWL geschrieben werden. Aber in AWL sind mehrere Anweisungen in einem Netzwerk möglich. Dadurch erkennt man besser die Zusammenhänge. Deshalb wurde hier AWL gewählt.

```
U  #RECHNE      //Wenn die Taste <BERECHNEN> betätigt wird...
FP M   10.0     ///(gilt nur einmal (Flanke)
```

Wenn der Eingang eine steigende (**P**ositive) **F**lanke hat, wird der Ausgang für einen Zyklus auf „1“ gesetzt. Achten Sie unbedingt darauf, dass der Flankenmerker (hier: M 10.0) nirgendwo anders (d.h. weder im Programm noch in der Anlage) verwendet wird! Merker M10.0 wird nirgends explizit aufgerufen. Er verhindert aber programmintern, dass <BERECHNEN> im nächsten Zyklus noch einmal durchgeführt wird (was zu einem falschen Wert in <#Ergebnis> führen würde).

```
UN "Überlauf"   //...und wenn der Datenbereich ok ist...
```

Es muss bei der Vorwahl sicher gestellt sein, dass alle Daten in einem richtigen Datenwort gespeichert sind. Würde durch eine falsche Vorwahl z. B. das DBW20 (Speicherplatz der Teilsumme) abgefragt werden, wäre die Aussage unbrauchbar und evtl. gefährlich falsch (falls dieser Wert woanders weiterverarbeitet würde). Diese Bedingung wird im OB 1, Netzwerk 5 überprüft.

```
U Z  1          //...und wenn der Ausgang des Zählers Z1 eine "1" liefert.
                //(Das ist immer der Fall, wenn Zählerstand >0 ist).
SPBN  end       /Falls vorstehende Bedingungen nicht erfüllt sind, springe zur Marke
                // <end:>
L  0            //sonst lade den Wert "0" ...
T  #Ergebnis    //...und transferiere ihn zum FC 1-Parameter <#Ergebnis>, der ihn in
                // den DB1.DBW20 schreibt.
```

Dieser Schritt an dieser Stelle (!) verhindert, dass bei erneut gedrücktem Taster <BERECHNEN> die Summe zum vorherigen Wert hinzuaddiert wird. Wie Sie gleich sehen werden, steht im DBW20 das jeweilige Zwischenergebnis. Dort bleibt das Ergebnis bis zum Ende der Berechnung erhalten. Würde der Wert nicht bei einer Neuberechnung auf 0 gesetzt, wäre der Anfangswert für die neue Berechnung (zu dem jeweils ein neuer Wert addiert werden soll) gleich dem Wert der alten Berechnung

```
AUF DB  1       //...DB1 aufgeschlagen.
```

Vgl. TrySim <Hilfe>|<Index> „DB“ und dort <aufschlagen>.

Wenn Sie innerhalb eines Programmteils häufig Daten aus dem gleichen Datenbaustein benötigen, dann „schlagen“ Sie diesen „auf“. Im folgenden brauchen Sie bei der Adressierung von Datenwörtern nicht mehr anzugeben, in welchem DB sich diese befinden, die SPS nimmt sie automatisch aus dem aufgeschlagenen Datenbaustein.

```
L   P#0.0          // Der Pointer wird auf das erste Byte im Datenbaustein gesetzt.
```

Pointer sind 4 Bytes groß und werden verwendet, um auf Bits im E- A- oder M- oder D-Bereich zu zeigen. In ihm werden keine Prozessdaten gespeichert, sondern eine Adresse innerhalb der SPS. Vgl. TrySim <Hilfe>|<Index> „Pointer“ .

```
L   #Ber_Anf       //Lade die Nummer des Datenwortes, bei dem die Auswertung
                   //beginnen soll.
                   //(nicht die Byte-Adresse im DB 1, sondern die
                   // "Zeilennummer", angefangen bei 0).
SLW 4              //Schiebe 4 Bits nach links. Dadurch wird die Zahl in das Pointer-
                   //Format umgewandelt.
                   //(Siehe Erläuterungen unter "Breakpoint" in diesem Projekt).
+I                 //Zähle diesen Wert zur Pointer-Position hinzu...
T   #Messwert_Nr   //...und transferiere ihn an die temporäre (flüchtige, nur intern
                   //benutzte) Variable.
```

Durch das Pointer-Format ist es sehr einfach, bezogen auf eine "Start"-Adresse jede gewünschte Zeile durch einen addierten Versatz zu beschreiben.

```
     L   "WIEVIEL"     //Zahl der Messwerte, die addiert werden sollen.
next: T   #Anzahl      //Sprungmarke. Transferiere "WIEVIEL" an die temporäre Variable.
                       //Der Wert entspricht der Zahl der noch verbleibende Zyklen in der
                       //LOOP-Schleife. Dieser Wert wird - siehe unten - von "LOOP"
                       //heruntergezählt.

L   DBW [#Messwert_Nr] //Lade das DB-Wort, das in der entsprechenden Zeile steht.
```

Diese Methode nennt man "speicherindirekte Adressierung". Der Aufruf eines Wortes "L DBW..." gilt allgemein. Variabel aber ist, welches Wort gesucht wird. Es wird also z.B. nicht "DBW4" fest vorgegeben, sondern in der eckigen (!) Klammer steht die temporäre Variable, in der die gewünschte Zeile im Pointer-Format steht.

```
L   #Ergebnis      //Über den  Parameter "#Ergebnis" wird der Wert von DB 1.DB 20
                   //(SUMME) geladen.
+I                 //beide Werte werden addiert...
T   #Ergebnis      //...und über <#Ergebnis> an DB 1.SUMME transferiert.
L   #Messwert_Nr   // Die bisherige Zeilenzahl wird in den AKKU geladen
L   P#2.0          // Byte 2, Bit 0 wird im Pointer-Format geladen.
+I                 //Beide Werte werden addiert, d.h. die alte Adresse wird um 2 Byte
                   //vergrößert.
```

Vergleichen Sie bitte die Adressen im DB 1. Sie sind wortweise angegeben, d.h., die Adressen springen immer um 2 Byte weiter. Somit wird die nächste Zeile ausgelesen, wenn mit "P#2.0" um 2 Byte gesprungen wird.

```
T   #Messwert_Nr   //Der neue Adresswert wird in der alten Temp-Variablen
                   //gespeichert.
L   #Anzahl        //Lade den Wert der bisher noch nötigen <next>-Schleifen.
```

Vor der <LOOP>-Anweisung muss noch einmal die alte Zahl stehen, die runtergezählt werden soll.

```
LOOP  next         //Reduzier- und Sprungbefehl zurück zur Marke <next>.
```

Die Zahl im Akku1 wird um 1 erniedrigt. Wenn Akku1 danach ungleich Null ist, wird automatisch zu der angegeben Sprungmarke gesprungen.

Im zweiten Durchgang von <next:> ist eingangs <#Anzahl> jetzt um 1 erniedrigt. Ebenfalls ist <#Messwert_Nr> für die Auswertung der nächsten Zeile schon eingestellt: der neue Wert (dieser neuen Zeile) wird geladen und zur alten DB 1.SUMME addiert. Der neue Wert überschreibt DB 1.SUMME, usw.

Erst wenn <#Anzahl> durch <LOOP> auf "0" gesetzt wird, wird das Programm normal fortgesetzt.

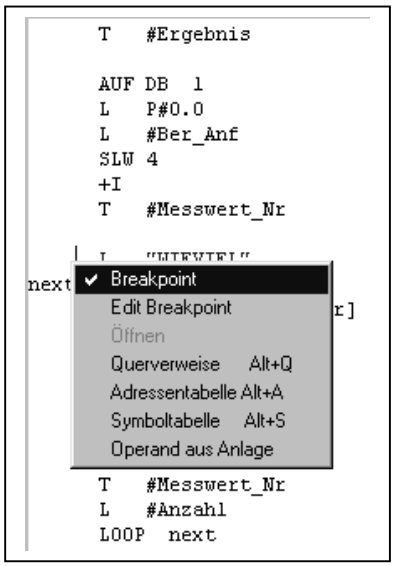
```
end:  U  "löschen"      //Wenn <löschen> betätigt wird...
      O  "1_Wert_N"     //...oder...
      O  "1_WERT_H"     //          ...einer...
      O  "MESSW_W"      //                    ...dieser...
      O  "MESSW_M"      //                              ...Operanden...
      SPBN  end2        //(falls doch nicht, springe zur Marke <end2:>)
      L   0             //...dann lade den Wert 0 ...
      T   #Ergebnis     //...und transferiere ihn über <#Ergebnis> an DB 1.SUMME.

end2: NOP 0             //Hier passiert nichts mehr. Diese Zeile ist lediglich als Sprungziel
                        //erforderlich, wenn DB 1.SUMME nicht auf "0" zurückgesetzt werden
                        //soll.
```

Hier noch einmal das FC 1-Programm im Zusammenhang:

```
      U  #RECHNE
      FP M   10.0
      UN "Überlauf"
      U  Z   1
      SPBN  end
      L   0
      T   #Ergebnis
      AUF DB  1
      L   P#0.0
      L   #Ber_Anf
      SLW 4
      +I
      T   #Messwert_Nr
      L   "WIEVIEL"

next: T   #Anzahl
      L   DBW [#Messwert_Nr]
      L   #Ergebnis
      +I
      T   #Ergebnis
      L   #Messwert_Nr
      L   P#2.0
      +I
      T   #Messwert_Nr
      L   #Anzahl
      LOOP  next

end:  U  "löschen"
      O  "1_Wert_N"
      O  "1_WERT_H"
      O  "MESSW_W"
      O  "MESSW_M"
      SPBN  end2
      L   0
      T   #Ergebnis

end2: NOP 0
```

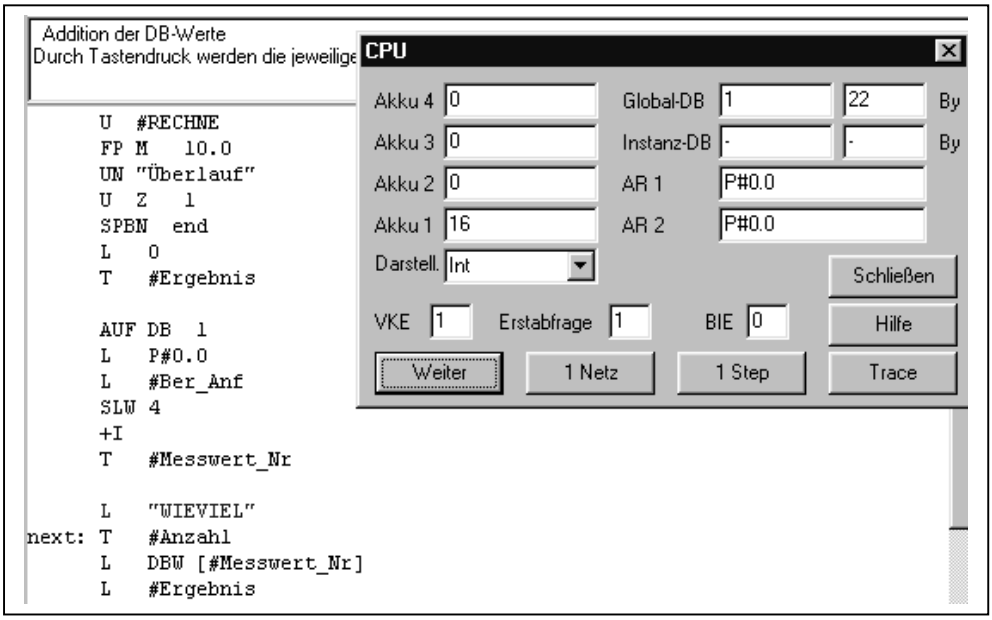
Dieses Projekt stellt eine gute Gelegenheit dar, ein wichtiges Hilfsmittel kennenzulernen: den **Breakpoint**.

Durch Breakpoints (Haltepunkte) können Sie die Bearbeitung des SPS-Programms an einer beliebigen Stelle unterbrechen. Die Bearbeitung der Simulation wird dann ebenfalls gestoppt. Vergleichen Sie bitte zu diesem Thema die TrySim-<Hilfe>|<Index> "Breakpoint".
Sie setzen einen Breakpoint, indem Sie im Editor auf die gewünschte Zeile mit <rM> klicken und aus dem Kontext-Menü „Breakpoint“ wählen. Die entsprechende Zeile wird dann blau markiert. (Eine derartige Markierung kann auch relativ leicht ungewollt passieren, wenn man mit <rM> das Kontextmenü aufruft und – schwupps - <LM> aus Versehen bei <Breakpoint> drückt).



Die Zeile vor der Sprungmarke <next:> wurde markiert.

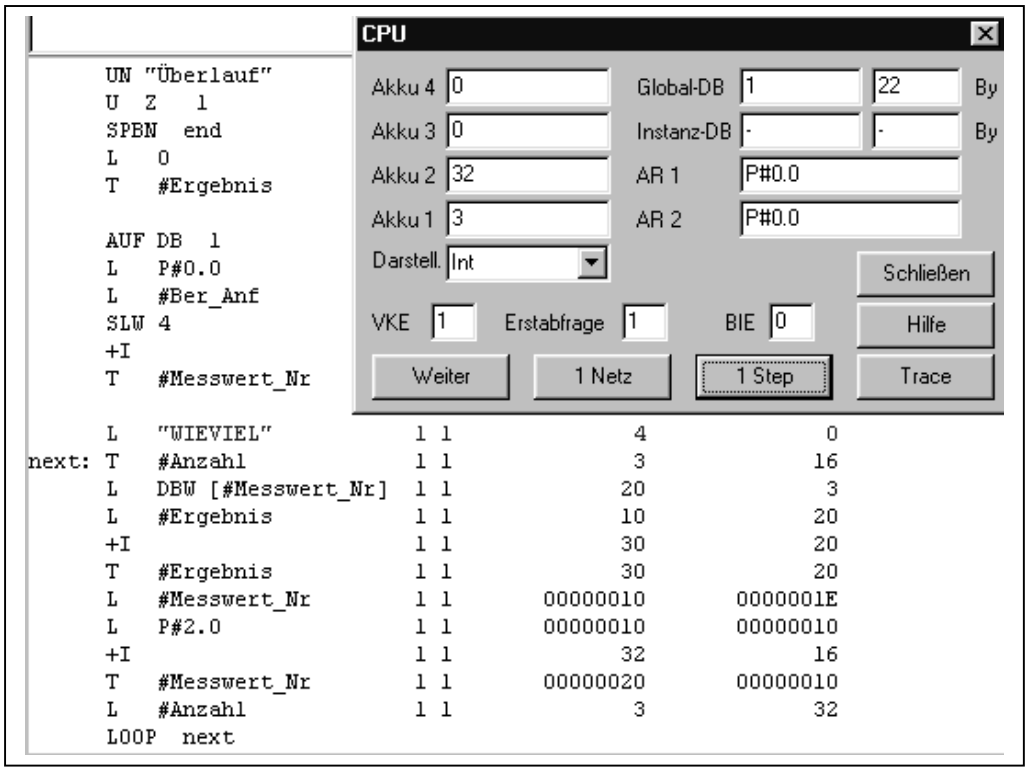
Wenn die Bearbeitung des SPS-Programms an diesen Punkt gelangt, wird das Programm angehalten, der Baustein wird geöffnet, das CPU-Fenster wird eingeblendet und die LED links unten wird gelb.
Die Bearbeitung des Programmes wird vor der markierte Zeile unterbrochen.



Tasten Sie bei aktivem FC 1-Fenster und eingeschaltem Auge-Icon mit <1 Step> Zeile für Zeile durch.

Sie erkennen in der AKKU1-Anzeige, dass 4 Messwerte mit "WIEVIEL" vorgewählt wurden. Der Wert wurde in der Zeile der Sprungmarke <next:> nach "<#Anzahl> transferiert.
Es wurde dann der Messwert der ersten abzufragenden Zeile (das erste Datenwort = "10") geladen. (Das könnte auch jede andere Zeile sein). Im DB 1.SUMME liegt - über <#Ergebnis> abgefragt - noch die "0" an. Dann wird "10" über <#Ergebnis> in DB 1.SUMME gelegt, was man aber dort nicht beim <Breakpoint> verfolgen kann. Das alte Startbit für die Information der ersten Zeile mit der Adresse 0.0 ist unter <#Messwert_Nr> hexadezimal mit "00000000" (= 0) angegeben. Dazu werden mit "P#2.0" HEX 10 (= 16 Bits = 2 Bytes = 1 WORD) dazugezählt. Die nächste <#Messwert_Nr> (HEX 10) bezieht sich also auf das DB-Wort ab 2.0, also auf die nächste Zeile.
Vor der <LOOP>-Anweisung muss noch einmal die alte Zahl stehen, die runtergezählt werden soll.

Nach "LOOP next" beginnt der Zyklus neu, jedoch - wie Sie sehen können, mit verringerter <#Anzahl>.



In der zweiten Runde ist die <#Anzahl> auf "3" reduziert, der neue Messwert der zweiten Zeile ist mit "20" ermittelt worden, das Zwischenergebnis ist "10". Beide Werte werden addiert und mit "30" an <#Ergebnis> übertragen. Zur <#Messwert_Nr> werden wieder 16 Bits addiert, so dass der Pointer vom Ursprung aus jetzt 32 Bits = 2 Worte weiterspringt, d.h., zur DB-Adresse 4.0.
Sie können die Entwicklung bis zum automatischen Ende der LOOP-Schleife weiter verfolgen und so die Wirkung und Richtigkeit des Programms kontrollieren.
Ohne Breakpoint könnten Sie die Zwischenwerte niemals wahrnehmen.
Nachdem das FC-Programm vollständig durchgetaktet wurde, wird das neue Ergebnis auch im DB angezeigt, wie man bei aktivem DB 1-Fenster und der Datenansicht (<Ansicht>|<Datenansicht> und Auge-Icon einschalten!) verfolgen kann.

DB1

	Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
▶	0.0	WertNr_0	INT	10	10	Wert 0 Byte 0
	2.0	WertNr_1	INT	20	20	Wert 1 Byte 2
	4.0	WertNr_2	INT	30	30	Wert 2 Byte 4
	6.0	WertNr_3	INT	5	5	Wert 3 Byte 6
	8.0	WertNr_4	INT	50	50	Wert 4 Byte 8
	10.0	WertNr_5	INT	5	5	Wert 5 Byte 10
	12.0	WertNr_6	INT	3	3	Wert 6 Byte 12
	14.0	WertNr_7	INT	2	2	Wert 7 Byte 14
	16.0	WertNr_8	INT	6	6	Wert 8 Byte 16
	18.0	WertNr_9	INT	4	4	Wert 9 Byte 18
	20.0	SUMME	INT	0	65	(Teil)-Summe

Sie können <Breakpoint> jederzeit mit <CPU>|<Schließen> beenden. Allerdings müssen Sie dann auch für den "Normalbetrieb" auch den gesetzten Breakpoint löschen, indem Sie aus dem Kontextmenü erneut die nun mit einem Häkchen versehene Zeile „Breakpoint“ wählen und damit das Häkchen zurücksetzen.

Beim Breakpoint-Betrieb sind in der Icon-Leiste die Icons für <Anlage starten> **und** <Anlage stoppen> aktiv.
Sie müssen noch auf eines der Icon mit <LM> klicken, um in den Normalbetrieb zu kommen.

vgl. TrySim-Projekt <LOOP> im Verzeichnis <Lösungen>

Zusammenfassung:

Sie müssen nicht schon beim Schreiben Ihres Programmes die Operanden endgültig festlegen, sondern können Sie erst während der Laufzeit bestimmen lassen. Nützlich ist dies, wenn immer wiederkehrende Operationen mit verschiedenen Operanden durchgeführt werden sollen. Wenn Sie z.B. 14 verschiedene Rezepte in den Datenbausteinen DB 1 – 14 gespeichert haben, und die Nummer des aktuellen Rezepts im MW 20 gespeichert ist, programmieren Sie: AUF DB[MW 20]

Vgl. TrySim <Hilfe>|<Index> „indirekte Adressierung“.

Sie haben gelernt:

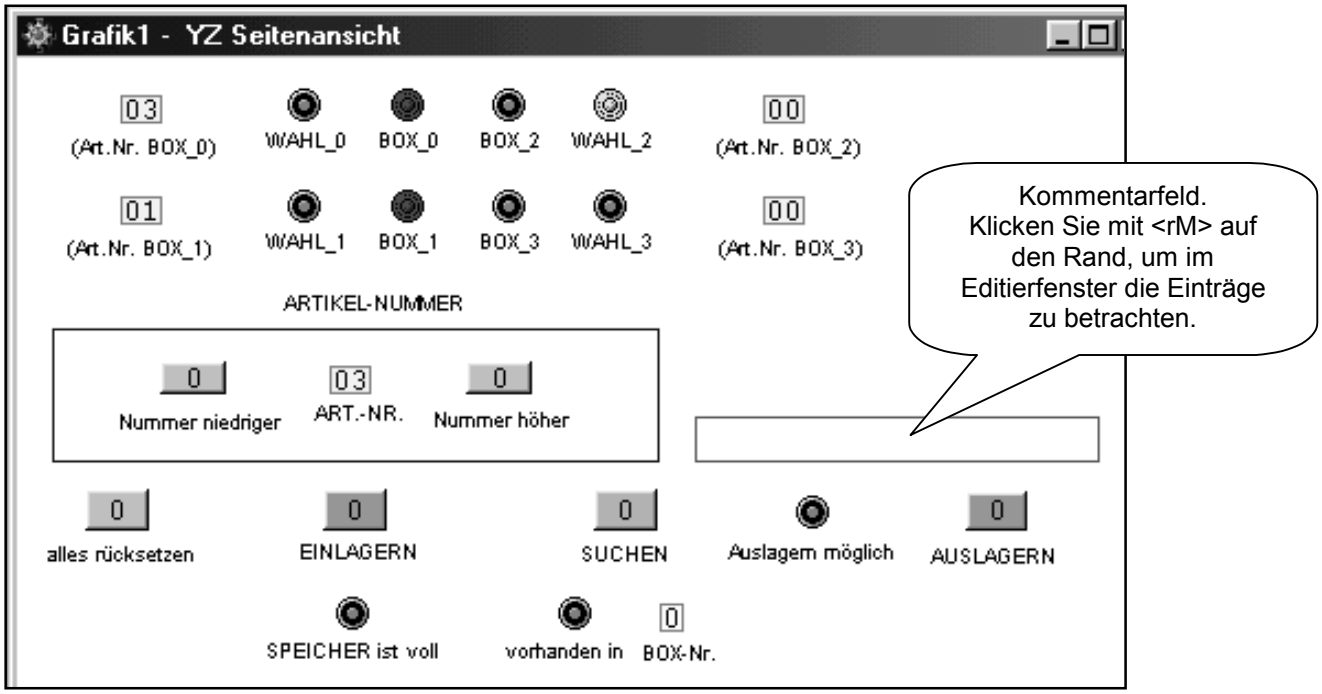
- Sie können Operationen, bzw. Programme auf variable Datenbereiche mit dem Pointer anwenden.
- Sie können wiederkehrende Schleifen mit „LOOP“ bilden.
- Sie können mit dem Breakpoint schrittweise das Programm verfolgen.
- Sie können mathematische Operationen durchführen und als AW ausgeben.
- Sie können temporäre Variablen deklarieren und anwenden.
- Sie können einen Datenbaustein aufschlagen.

Projekt 34: Regallager

Schwerpunkte: LOOP, FC, DB, AUF, +I, <>I, ==I, SPB, SPA, SLW, indirekte Adressierung, ARRAY, Pointer

Aufgabe: Mini-Regallager mit selbständiger Suche der nächsten freien Box.

Ein kleines Regallager besteht aus 4 Speicherboxen, in die jeweils ein Paket automatisch gelegt und auch wieder abgerufen werden kann. Für jedes Paket kann bzw. muss eine Artikel-Nummer vergeben werden. Unter dieser Codenummer wird die Ware automatisch mit <EINLAGERN> in der nächsten freien Regalbox gespeichert. Die gefüllten Boxen werden durch LEDs und die dazugehörigen Artikelnummern als Digitalanzeigen dargestellt.
Zum Auslagern wird der gewünschte Artikel über die Eingabe <Art.Nr.> vorgewählt und über 2 Schritte ausgelagert = (gelöscht): 1. <SUCHEN>, 2. <AUSLAGERN>. Es wird also nicht eine Box aufgerufen, sondern der Artikel wird gesucht. Das Programm ermittelt, wo der Artikel liegt.
Für die Platzsuche der gewünschten Artikel wird die „indirekte Adressierung“ gewählt. Dieses Verfahren wird in diesem Projekt erläutert.
Hinweis: Art.-Nr. 00 kann nicht vergeben werden, denn sie bedeutet "leeres Fach".
Zum Verständnis sollte vorher das Projekt <LOOP> bearbeitet werden.
Dieses Beispiel erhebt nicht den Anspruch, eine industriell ausgereifte Lösung für ein Großlager zu sein. Es ist gewählt worden, um unabhängig von reinen Steuerungsaufgaben einen Einblick in die vielfältigen Möglichkeiten der erweiterten Befehle zu geben.
Es wird hier nur die Lager-Logik programmiert und nicht die mechanische Lösung.



So wird zum Schluss die Übersicht der Bausteine aussehen:

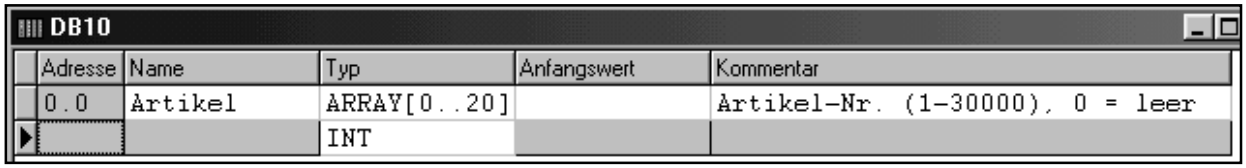
Das Projekt ist vorbereitet unter <Vorlagen> <LagerLogic_V>. Sie können es nach dem Öffnen in Ihr Arbeitsverzeichnis kopieren und dort bearbeiten.

Symbolische Adresse	Operand	Datentyp	Kommentar
Wahl_0	A 0.0	BOOL	freie Box 0
Wahl_1	A 0.1	BOOL	freie Box 1
Wahl_2	A 0.2	BOOL	freie Box 2
Wahl_3	A 0.3	BOOL	freie Box 3
BOX_0	A 0.4	BOOL	Box 0 ist belegt
BOX_1	A 0.5	BOOL	Box 1 ist belegt
BOX_2	A 0.6	BOOL	Box 2 ist belegt
BOX_3	A 0.7	BOOL	Box 3 ist belegt
IST_IN	A 1.0	BOOL	LED Artikel vorhanden
voll	A 4.4	BOOL	alle Boxen sind besetzt
LED_AUSL	A 5.0	BOOL	LED "Auslagern möglich"
ART_POS0	AW 512	WORD	Artikelnummer in Box 0
ART_POS1	AW 514	WORD	Artikelnummer in Box 1
ART_POS2	AW 516	WORD	Artikelnummer in Box 2
ART_POS3	AW 518	WORD	Artikelnummer in Box 3
SUCHEN	E 0.6	BOOL	Artikelnummer suchen in BOX...
NR_niedr	E 0.7	BOOL	Artikelnummer kleiner vorgeben
NR_höher	E 1.0	BOOL	Artikelnummer größer vorgeben
AUSLAGRN	E 1.2	BOOL	BOX leeren
LEEREN	E 2.2	BOOL	alle Boxen leeren
EINLAGRN	E 2.3	BOOL	Artikel einlagern in gewählte Box
BOX_NR	MW 10	WORD	aktuelle BOX-Nummer
ART_NR	AW 520	WORD	Digitalanzeige Artikelnummer
SUCHMERK	M 2.2	BOOL	speichert Suchimpuls
WARNSUCH	M 2.3	BOOL	Hinweis: erst SUCHEN, dann AUSLAGERN
FMEinlag	M 1.1	BOOL	FlankenMerker: nur einmal einlagern
AusLMoeg	M 1.2	BOOL	Auslagern möglich
ART_NR_0	M 30.3	BOOL	Artikel-Nr. = 0
F_MERK_1	M 30.4	BOOL	Flankenmerker

Die relevanten Daten sollen im DB 10 gespeichert und verwaltet werden.

DB 10

Den DB legen Sie unter <SPS>|<Neu>|<DB> mit „10“ im Auswahlfenster und <OK> an. Speichern Sie das bisherige Programm und rufen Sie den neuen DB auf mit <SPS>|<Öffnen>|<DB 10> (markieren) und <OK>. Dort können Sie unter <Typ> ein Array nach folgender Schreibweise anlegen:



Dadurch könnten Sie das Lager schnell auf 20 (oder mehr) Boxen erweitern.
Unter <Ansicht>|<Datenansicht> ergibt sich folgende Darstellung:

DB10

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	Artikel[0]	INT	0	0	
2.0	Artikel[1]	INT	0	0	
4.0	Artikel[2]	INT	0	0	
6.0	Artikel[3]	INT	0	0	
8.0	Artikel[4]	INT	0	0	
10.0	Artikel[5]	INT	0	0	
12.0	Artikel[6]	INT	0	0	
14.0	Artikel[7]	INT	0	0	
16.0	Artikel[8]	INT	0	0	
18.0	Artikel[9]	INT	0	0	
20.0	Artikel[10]	INT	0	0	
22.0	Artikel[11]	INT	0	0	
24.0	Artikel[12]	INT	0	0	
26.0	Artikel[13]	INT	0	0	
28.0	Artikel[14]	INT	0	0	
30.0	Artikel[15]	INT	0	0	
32.0	Artikel[16]	INT	0	0	
34.0	Artikel[17]	INT	0	0	
36.0	Artikel[18]	INT	0	0	
38.0	Artikel[19]	INT	0	0	
40.0	Artikel[20]	INT	0	0	

Durch die Änderung der „20“ im Typ-Feld der Deklarationsansicht ändert sich automatisch die Zeilenzahl in der Datenansicht.

Bevor die Funktionen im OB 1 aufgerufen werden können, müssen die FCs – zumindest als Baustein (noch ohne Programm) – angelegt worden sein. In der folgenden Erläuterung werden sie jedoch schon vollständig programmiert.

FC 1

FC 1 Netzwerk 1 Diese FC wird aufgerufen, wenn ein Artikel eingelagert wird

Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
0.0	in	Artikel	INT		
	out				
	in_out				
0.0	temp	PlatzIdx	DWORD		Lagerplatz - Index
4.0	temp	Idx	INT		Schleifen - Index

An den In-Parameter #Artikel wird – von außen - der Artikel (die Artikel-Nummer) übergeben, der eingelagert werden soll. Der Artikel wird in der ersten freien Box eingelagert. Diese muss gesucht werden.
Vor dem Aufruf muss sichergestellt sein, dass noch ein Lagerplatz frei ist.
Die Variablen <PlatzIdx> und <Idx> werden nur intern – temporär - in dieser FC benötigt. Deshalb werden sie unter <temp> deklariert.

FC 1; Netzwerk 1: Diese FC wird aufgerufen, wenn ein Artikel eingelagert wird

```
AUF DB  10          // Der DB 10 der Lager-Belegung wird auf geschlagen
```

Wenn Sie innerhalb eines Programmteils häufig Daten aus dem gleichen Datenbaustein benötigen dann „schlagen“ Sie diesen „auf“. Im folgenden brauchen Sie bei der Adressierung von Datenwörtern

nicht mehr anzugeben, in welchem DB diese sich befinden, die SPS nimmt sie automatisch aus dem aufgeschlagenen Datenbaustein. (vgl. TrySim <Hilfe> <Index> „Aufschlagen, DB“>).

Jetzt wird eine Routine entwickelt, um den ersten freien Platz zu finden. Im Prinzip muss jede Box abgefragt werden. Das Programm kann etwas „automatisiert“ werden. Jedes Mal wird gefragt, ob der Inhalt der DB-Zeile gleich 0 ist. Es ändert sich lediglich die DB-Zeile. Mit der Pointer-Operation lassen sich diese unterschiedlichen Werte nacheinander an dieselbe Stelle der Vergleichsabfrage einbauen.

```
      L   P#0.0            // Der Pointer wird auf die Adresse der ersten Box im aufgeschlagenen DB 10
                           // gesetzt.
      T   #PlatzIdx        // Die Adresse wird an die temporäre Variable transferiert
      L   4                //Lade die Zahl 4, da insgesamt 4 Schleifendurchläufe (für 4 Boxen) geplant
                           // werden.
loop: T   #Idx             // „loop:“ ist eine - beliebige – Sprungmarke und nicht mit der Operation
                           // <LOOP> zu verwechseln. Hierher wird später zurückgesprungen.
                           // „4“ wird an <#Idx> transferiert. So beginnt immer die <LOOP>-Schleife.
      L   DBW [#PlatzIdx]  //Der DB 10 ist oben schon aufgeschlagen worden, deshalb wird hier
                           //nur noch das entsprechende Wort geladen, das an der Adresse
                           //steht, die hier übergeben wurde.
      L   0                //Lade die Zahl 0
      <>I                  //Die beiden Werte werden verglichen.
                           //Wenn schon ein Artikel vorhanden ist, sind sie ungleich; dann ist...
      SPB   voll           // ...diese Box voll => nächsten prüfen
      L   #Artikel         //sonst lade die Artikel-Nr.
      T   DBW [#PlatzIdx]  // und lege sie im Datenwort ab, dessen Adresse #PlatzIDX angibt.
      SPA   NWE1           // Schleifen-ABBRUCH, die FC hat ihre Aufgabe erfüllt.
voll: L   #PlatzIdx        // Der alte Platz-Index wird geladen...
      L   P#2.0            // ...und um zwei Bytes...
      +I                   // ...erhöht.
      T   #PlatzIdx        //<#PlatzIdx> weist jetzt auf die neue Adresse hin
      L   #Idx             // Der Schleifen-Index muss immer vor LOOP geladen werden.
      LOOP  loop           // Springe nach <loop:> und verringere den alten Wert <#Idx> um 1.
                           //Wenn <#Idx> = 0, wird das Programm ohne Rücksprung fortgesetzt.
NWE1: NOP 0                //Dieses Netzwerk ist beendet.
```

FC 2

FC 2 Netzwerk 1 Wenn diese FC aufgerufen wird, werden alle Lagerplätze als LEER

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
▶		in				
		out				
		in_out				
	0.0	temp	Idx	INT		Schleifen-Index
	2.0	temp	PlatzIdx	DWORD		Zeiger auf Speicherplatz im DB

FC 2; Netzwerk 1: Wenn diese FC aufgerufen wird, werden alle Lagerplätze als LEER gekennzeichnet (Art-Nr. = 0)

Dieses Programm ähnelt sehr der FC 1. Ausgehend von der ersten Wortadresse wird jedes Wort auf den Wert „0“ gesetzt. Der Schleifenzähler wird durch <LOOP> runtergezählt. Gleichzeitig wird die Wortadresse jeweils um ein Wort erhöht.

```
      AUF DB  10           //Aufschlagendes DB 10
      L   P#0.0            //Pointer auf Bit 0.0
      T   #PlatzIdx        //Transfer
      L   4                //Lade Zahl 4 (Schleifendurchläufe)
```

```
loop:  T   #Idx          //Transfer an Temp.-Variable #Idx
       L   0             //Lade Zahl 0
       T   DBW [#PlatzIdx]  //Transfer an das Wort mit der Adresse des Pointers.
       L   #PlatzIdx     //Lade Pointeradresse
       L   P#2.0         //Lade 16 Bit = 1 Wort = 2 Byte weiter
       +I                //Addiere beide Werte
       T   #PlatzIdx     //Transferiere den neuen Adresswert an <#PlatzIdx>
       L   #Idx          //Lade vor <LOOP> die Schleifenzahl
       LOOP  loop        //Verringere den alten Wert <#Idx> um 1 und springe nach <loop:>.
```

Die deklarierten Namen gelten nur innerhalb der FC. „Idx“ der FC1 hat mit der gleichnamigen Bezeichnung in der FC2 nichts zu tun. Die Namen gelten also nur lokal.

FC 3

FC 3 Netzwerk 1 Suchen

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
	0.0	in	Artikel	INT		
	2.0	out	Box_Nr	INT		
		in_out				
	0.0	temp	Idx	INT		
▶	2.0	temp	PlatzIdx	DWORD		

FC3; Netzwerk 1: Suchen eines Artikels
Diese FC sucht den ersten Platz, der mit <#Artikel> belegt ist. Die Nr. des Platzes wird in Box_Nr. abgelegt. Wird der Artikel nicht gefunden, wird "-1" ausgegeben

```
      AUF DB  10
      L   P#0.0
      T   #PlatzIdx
      L   4
loop: T   #Idx
      L   #Artikel
      L   DBW [#PlatzIdx]
      ==I               // Wenn die gesuchte Artikelnummer mit dem Datenwort übereinstimmt…
      SPB   Ok          // …springe zur Sprungmarke <Ok:>. Sonst...
      L   #PlatzIdx     // …lade den alten Platzindex und…
      L   P#2.0         // …erhöhe die Adresse…
      +I                // …durch Addition um 2 Byte.
      T   #PlatzIdx     // Die neue Adresse wird #PlatzIdx zugewiesen.
      L   #Idx          // Der alte Wert #Idx muss hier vor <LOOP> geladen werden.
      LOOP  loop        // Fall #Idx nicht 0 ist, springe zur Marke <loop :>. Sonst setze fort:
      L   -1            // Schleife ist durchgelaufen,
      T   #Box_Nr       // nichts gefunden, also: Box_Nr := -1
      BEA               //Der Baustein wird ohne Sprung und ohne weitere Bedingungen beendet.

Ok:   L   #PlatzIdx     //Lade den aktuellen Wert der gerade gefundenen Box-Adresse.
      SRW 4     //Schiebe die Stellen um 4 nach rechts =>Teile diesen Wert durch 16
      T   #Box_Nr       //Zeige die gefundene Box an.
```

Zur Erklärung von „SRW 4“:
Bei jeder <LOOP>-Schleife wurde der Pointer um 16 Bits höher gesetzt. Der aktuelle Wert entspricht also einem Vielfachen von 16. Ein Verschieben des dualen Bitmusters um eine Stelle nach rechts ist gleichzusetzen mit einer Division durch 2. Das gleiche gilt für eine Wiederholung. 4 mal um eine Stelle verschieben bedeutet also, den Wert viermal zu halbieren, was einem Sechzehntel entspricht.

FC 4

FC 4 Netzwerk 1 Auslagern eines Artikels

	Adresse	Deklaration	Name	Typ
▶	0.0	in	BoxNr	INT
		out		
		in_out		
	0.0	temp	TBoxNr	DWORD

FC 4; Netzwerk 1: Auslagern eines Artikels
Das Auslagern besteht eigentlich nur darin, dass der Artikel in <#BoxNr> auf Null gesetzt wird.
Mit „SLW 4“ wird das Umgekehrte zu FC 3 erreicht: Aus der <#BoxNr> wird durch die Multiplikation mit 16 das Format erreicht, das dem Bit der Adresse entspricht, unter der die Information „0“ gespeichert werden soll.

```
AUF DB  10
L   #BoxNr
SLW 4
T   #TBoxNr
L   0
T   DBW [#TBoxNr]
```

FC 5

FC 5 Netzwerk 1 Speicher Voll?

	Adresse	Deklaration	Name	Typ
▶		in		
	0.0	out	Voll	BOOL
		in_out		
	0.0	temp	Idx	INT
	2.0	temp	BoxIdx	DWORD

FC 5; Netzwerk 1: Speicher Voll?
Diese FC setzt des Ausgang #Voll, wenn alle Plätze eine von "0" verschiedene Artikel-Nr. enthalten.
Hier wird zur Abwechselung der Pointer nicht hochgezählt, sondern runter. Die Bedeutung von „SLW 4“ wurde bereits erwähnt.

```
        AUF DB  10
        CLR                 //“CLR“ bewirkt das Zuweisen einer „0“ ohne Verknüpfungsabfrage.
        =  #Voll            //<#Voll> wird auf „0“ gesetzt
        L   4
loop:   T   #Idx
        L   1
        -I
        SLW 4
        T   #BoxIdx
        L   DBW [#BoxIdx]   //Es wird geprüft, ob ein Wort „0“ (= leer) ist.
        L   0               //Lade „0“
        ==I                 //Vergleich
        BEB                 // EXIT, der Speicher ist nicht voll
```

```
L   #Idx          //Lade den aktuellen Schleifenwert
LOOP  loop        //Verringere den alten Wert <#Idx> um 1 und springe nach <loop:>.

SET               // Schleife ist durchlaufen, ohne einen leeren
= #Voll           // Platz zu finden -> Speicher ist voll
BEA
```

Hinweis: „SET“ bewirkt eine „1“, ohne dass dafür ein Verknüpfungsergebnis (VKE) erforderlich war.

Die weiteren Programmteile werden alle im OB1 angelegt. Ebenso erfolgt vom OB1 der Aufruf der bisher schon angelegten FC-Netzwerke.

OB 1

OB 1; Netzwerk 1: Abfrage der DBs (Artikelnummer)
Transfer der Artikelnummern an die Anzeigen

```
L   DB  10.Artikel[0]
T   "ART_POS0"
L   DB  10.Artikel[1]
T   "ART_POS1"
L   DB  10.Artikel[2]
T   "ART_POS2"
L   DB  10.Artikel[3]
T   "ART_POS3"
```

Die Speicherzellen des DB 10 werden über symbolische Adressen abgefragt und deren Inhalte werden direkt auf die Anzeigen „geschaltet“.

OB 1; Netzwerk 2: Anzeige der Belegung der Boxen

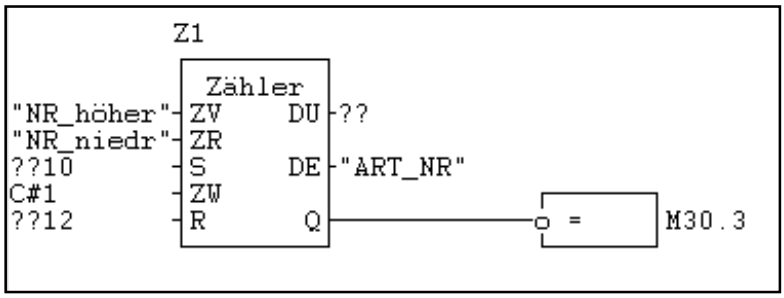
```
L   DB  10.Artikel[0]       //Lade die Artikelnummer der Box 0
L   0                       //Lade die Zahl 0
<>I                         //Wenn die Werte unterschiedlich sind,
=  "BOX_0"                  //ist die Box belegt. Die LED soll leuchten.
L   DB  10.Artikel[1]       //Lade die Artikelnummer der Box 1 ...
L   0
<>I
=  "BOX_1"
L   DB  10.Artikel[2]
L   0
<>I
=  "BOX_2"
L   DB  10.Artikel[3]
L   0
<>I
=  "BOX_3"
```

OB 1; Netzwerk 3: Anzeige der ersten leeren Box (Wahl_x)

```
CLR                         //Dieser Befehl setzt das VKE ohne weitere Bedingung auf „0“.
 =  "Wahl_0"                //Alle Wahl-Boxen werden zuerst auf „0“ gesetzt.
 =  "Wahl_1"
 =  "Wahl_2"
 =  "Wahl_3"
 L   DB  10.Artikel[0]
```

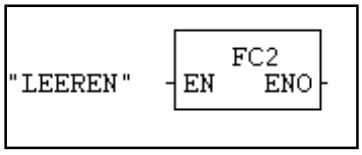
```
L  0
==I                     //Vergleiche die Art.-Nr. mit 0.
=  "Wahl_0"             //Wenn gleich, dann leer => Anzeige
SPB  NWE3               //Nach der ersten freien Box springe zum Ende
L  DB  10.Artikel[1]    //...sonst frage die nächste Box ab...
L  0
==I
=  "Wahl_1"
SPB  NWE3
L  DB  10.Artikel[2]
L  0
==I
=  "Wahl_2"
SPB  NWE3
L  DB  10.Artikel[3]
L  0
==I
=  "Wahl_3"
SPB  NWE3

NWE3: NOP 0             //Dieses ist nur das Sprungziel; ohne Operation
```

OB 1; Netzwerk 4: Artikel-Nummer



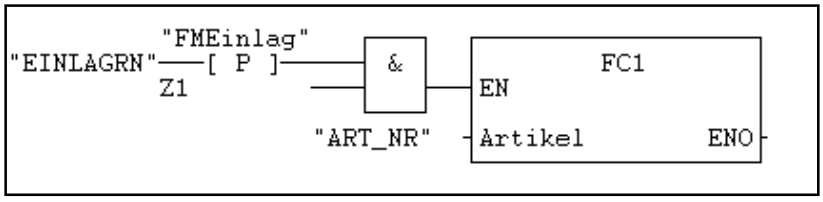
Mit dem Vorwärts- /Rückwärts-Zähler wird die Artikelnummer ausgewählt.
M 30.3 gibt „1“-Signal, wenn „0“ am Zähler ansteht. Das bedeutet: „leer“. Diese Art.-Nr. darf nicht zugewiesen werden.

OB 1; Netzwerk 5: DB "alle Boxen leeren"



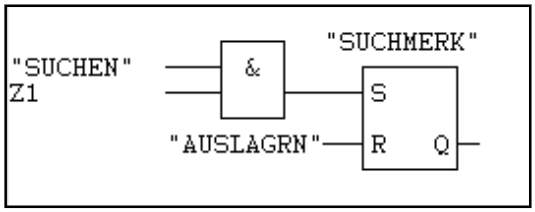
Nur wenn <alles rücksetzen> betätigt wird, wird FC2 über den „EN“-Eingang aktiviert. Es muss also intern in der FC2 keine Startbedingung erfüllt sein.

OB 1; Netzwerk 6: Ablage



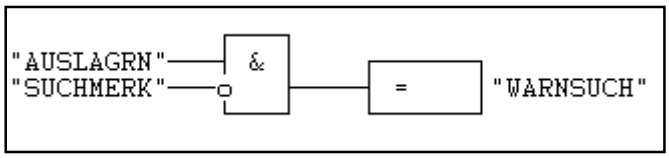
„Z1“ ist gleichbedeutend mit dem Ausgang Z1, Klemme Q. Wenn der Zählerwert > 0 ist, wird eine „1“ geliefert. Also nur bei entsprechender Codierung kann eingelagert werden.
Der Flankenmerker <FMEinlag> verhindert, dass <Einlagern> zum mehrmaligen Einlagern führt.

OB 1; Netzwerk 7: Such-Merker

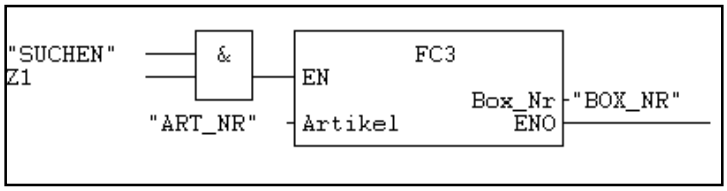


Der Such-Merker ermöglicht im nächsten Netzwerk den Hinweis, dass Auslagern nur nach vorherigem Suchen möglich ist. Nur wenn eine Art.-Nr. > 0 vorgewählt wurde (Z1 = „1“), kann gesucht werden.

OB 1; Netzwerk 8: Auslagern nur nach "SUCHEN" möglich
M2.3 (WARNSUCH) erzeugt einen Hinweis im Textfenster.
Mit <rM> auf den Textfenster-Rahmen öffnet sich das Editierfenster. Dort sehen Sie die anderen Einträge und Aufrufoperanden für die Meldungen.



OB 1; Netzwerk 9: Suchen
Wenn auf den Taster <Suchen> gedrückt wird, soll der erste Platz, der den Artikel in der Anzeige "Art.-Nr." enthält, in "Box-Nr." angezeigt werden. "-1" steht dafür, dass der Artikel nicht vorhanden ist.



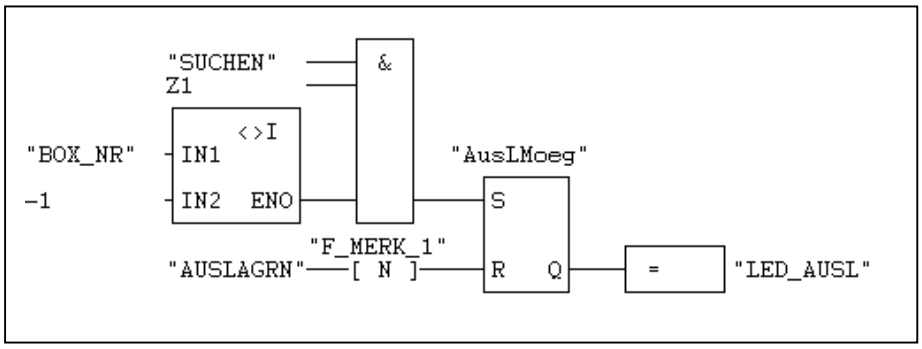
Wenn sie das Auge-Icon aktiviert haben, in dem Netzwerk die Werte von „ART_NR“ und „BOX_NR“ angezeigt. Wenn Sie allerdings die Artikelnummer ändern, „versagt“ die Anzeige in diesem Netzwerk. Wenn Sie sich dieses Netzwerk in AWL anzeigen lassen, wird Ihnen schnell klar, warum.

Das Netzwerk wird folgendermaßen übersetzt:

```
      U  "SUCHEN"
      U  Z   1
      SPBNB _142
      CALL FC 3
      Artikel := "ART_NR"
      Box_Nr  := "BOX_NR"
_142: NOP 0
```

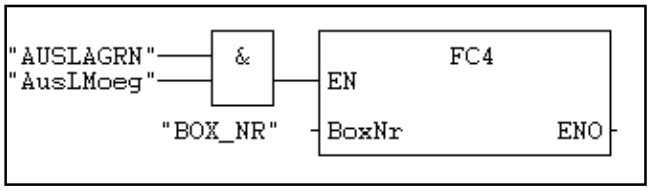
Wenn die UND-Verknüpfung von „SUCHEN“ und „Z1“ nicht gegeben ist, wird der Aufruf übersprungen und der alte Wert wird weiterhin angezeigt. Dass alles richtig arbeitet, merken Sie, wenn an „EN“ die „1“-Bedingung erfüllt ist. Dann ändern sich auch die aktuellen Werte.

OB 1; Netzwerk 10: Auslagern möglich

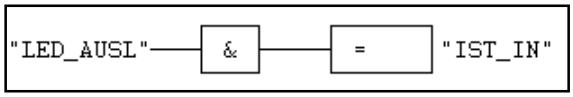


Wenn <Suchen> betätigt wird, die Code-Nr. > 0 ist (Z1 = 1) und eine Box_Nr. > -1 ausgewählt wird (ein Artikel vorhanden ist), wird das Auslagern möglich. Zurückgesetzt wird der SR-Merker beim <Auslagern>. „AusLMoeg“ ist eine Vorbedingung für das Auslagern. Deshalb darf das SR-FF nicht zu früh zurückgesetzt werden. Irgendwo muss aber der Rücksetzbefehl herkommen. Deshalb wird dafür die negative Flanke benutzt, die erst beim Loslassen von <AUSLAGERN> entsteht.

OB 1; Netzwerk 11: Auslagern
Beim Auslagern wird das zugehörige Datenwort auf Null gesetzt

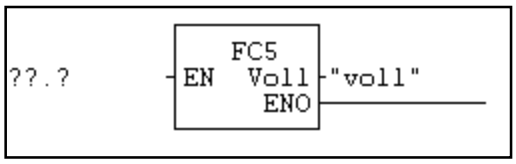


OB 1; Netzwerk 12: LED "aktuell"



Eine zusätzliche LED zeigt an, dass der gewünschte Artikel vorhanden ist.

OB 1; Netzwerk 13: Anzeige Speicher Voll
Diese FC setzt den Ausgang #Voll auf "1", wenn keine Box "0" enthält.



Vgl. TrySim-Projekt <LagerLogic> im Verzeichnis <Lösungen>.

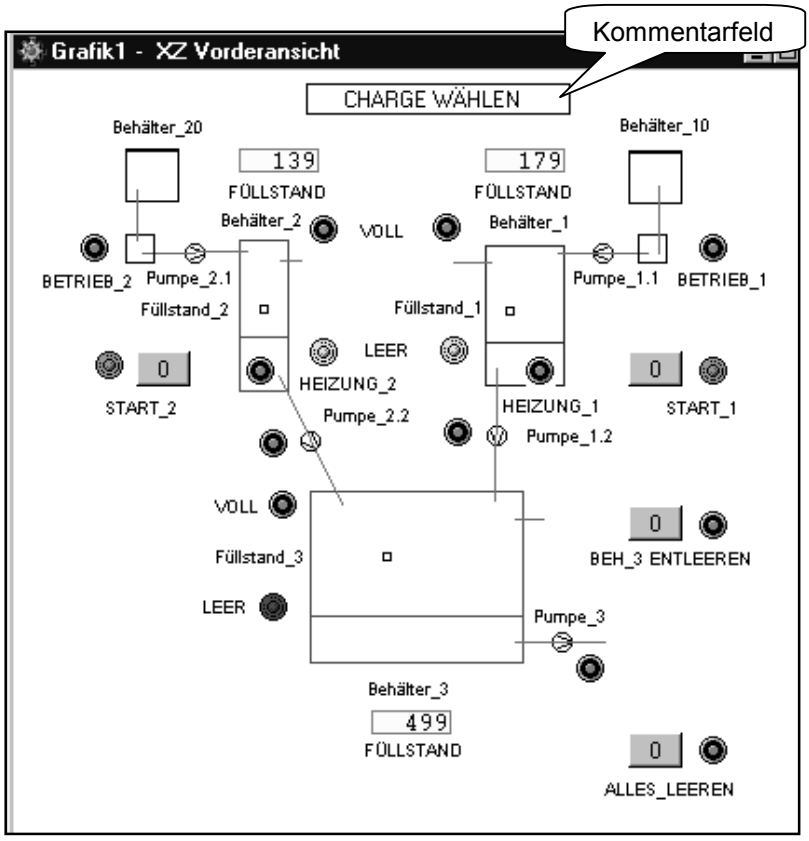
Projekt 35: Mischanlage

Inhalte: WORD, <I, Impuls, Flanke, Sprünge,

Aufgabe: 2 Flüssigkeiten können alternativ über Behälter 1 oder Behälter 2 aufbereitet und im Behälter 3 gesammelt werden, bis Behälter 3 voll ist. Der Ablauf, der also für beide Flüssigkeiten gilt, wird hier für den rechten Pfad der unten abgebildeten Anlage erläutert.
Aus der Quelle 1 (Behälter_10) wird mit der Pumpe_1.1 der Behälter_1 gefüllt. Bei max. Füllung wird die Flüssigkeit erhitzt. Der Füllstand wird mit Füllstand_1 ermittelt. Anschließend fördert Pumpe_1.2 die Flüssigkeit in den größeren Behälter_3. Dieser Vorgang wiederholt sich automatisch, bis Behälter_3 gefüllt ist. Auch hier wird der Füllstand mit Füllstand_3 ermittelt. Anschließend kann mit <BEH_3 ENTLEEREN> Pumpe_3 eingeschaltet werden, bis Behälter_3 leer ist. Danach kann mit <START_1> oder <START_2> der neue Füllvorgang ausgewählt werden. Mit <ALLES_LEEREN> können jederzeit alle Behälter geleert werden.

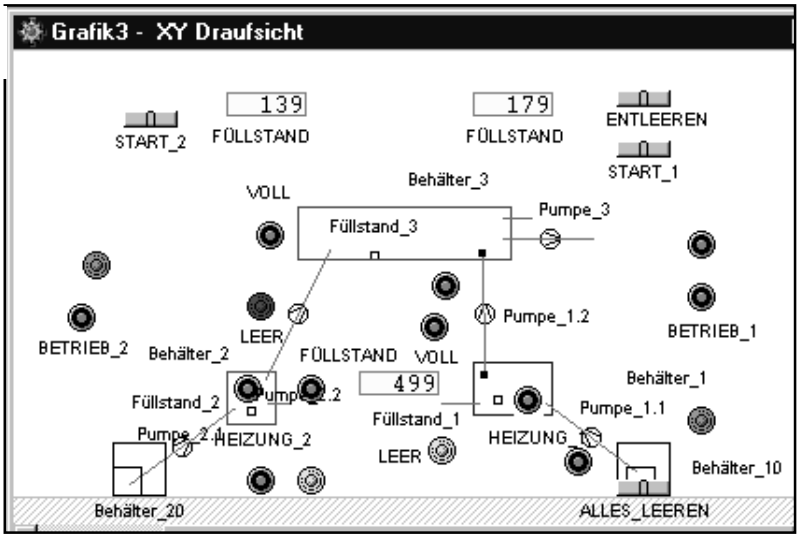
Durchführung des Projektes mit TrySim:
Im Verzeichnis <Vorlagen> ist das Projekt unter <Mischer_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Anlage und die Symboltabelle sind bereits vorbereitet.



Falls Sie derartige Anlagen selbst entwerfen wollen, sei vermerkt, dass die Komponenten in der Vorderansicht plaziert wurden. Nur dann können Sie erkennen, wie die Flüssigkeit in den Behältern steigt und fällt. Schalten Sie auch unter <Grafik> die vielleicht auf den ersten Blick etwas abschreckend wirkende Ansicht <Grafik 3, XY Draufsicht> ein. Hier sehen Sie, dass die Leitungen dreidimensional zueinander passen müssen. Es sind für diese Ansicht nicht alle Elemente, die in der Vorderansicht eingesetzt wurden, in der Lage optimiert. Wichtig ist nur, dass die Anschlüsse räumlich zueinander passen. Sie können in jeder Ansicht – bei ausgeschalteter Anlage! – jedes Element anklicken und in der Lage verändern, bzw. in der anderen Ansicht verfolgen. (In dieser Vorlage sind alle Komponenten fixiert, damit kein „Leck“ durch ungewolltes Verschieben mit der Maus entsteht).

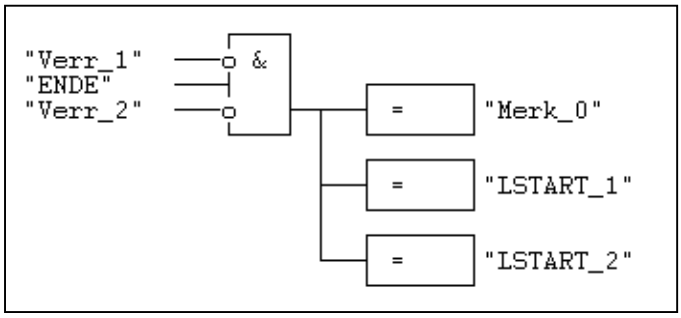
<Grafik 3, XY Draufsicht>



Symboltabelle

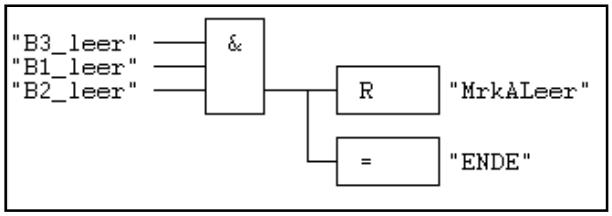
Symbolische Adresse	Operand	Datentyp	Kommentar
PUMPE_11	A 0.0	BOOL	Antrieb Pumpe 1.1
PUMPE_12	A 0.1	BOOL	Antrieb Pumpe 1.2
PUMPE_3	A 0.2	BOOL	Antrieb Pumpe 3
HEIZ_1	A 0.3	BOOL	LED Heizung 1
VOLL_3	A 0.4	BOOL	LED Behälter 3 voll
LLEEREN	A 0.5	BOOL	LED Bereit zum Entleeren
LEER_1	A 0.6	BOOL	LED Behälter 1 ist leer
VOLL_1	A 0.7	BOOL	LED Behälter 1 ist voll
LBETR_1	A 1.0	BOOL	LED Pumpe 1.1 ist in Betrieb
LPUMP_12	A 1.1	BOOL	LED Pumpe 1.2 ist in Betrieb
LPUMP_3	A 1.2	BOOL	LED Pumpe 3 ist in Betrieb
LEER_3	A 1.3	BOOL	LED Behälter 3 ist leer
LSTART_1	A 1.4	BOOL	LED START 1
VOLL_2	A 1.5	BOOL	LED Behälter 2 ist voll
LEER_2	A 1.6	BOOL	LED Behälter 2 ist leer
PUMPE_21	A 1.7	BOOL	Antrieb Pumpe 2.1
LBETR_2	A 2.0	BOOL	LED Pumpe 2.1 ist in Betrieb
PUMPE_22	A 2.1	BOOL	Antrieb Pumpe 2.2
LSTART_2	A 2.2	BOOL	LED START 2
HEIZ_2	A 2.3	BOOL	LED Heizung 2
LPUMP_22	A 2.4	BOOL	LED Pumpe 2.2 ist in Betrieb
BLINKLED	A 2.5	BOOL	Hinweis: Behälter_3 leeren
START_1	E 0.0	BOOL	Starttaster 1 (Schließer) Prog. 1
START_2	E 0.1	BOOL	Starttaster 2 (Schließer) Prog. 2
ENTLEER	E 0.2	BOOL	Taster "Entleeren" (Schließer)
AL_LEER	E 0.3	BOOL	Alle Behälter leeren
HeizEnd1	M 2.0	BOOL	Heizzeit 1 zu Ende
B3_n_voll	M 2.1	BOOL	Behälter 3 ist noch nicht voll
B3_leer	M 2.2	BOOL	Behälter 3 ist leer
B1_leer	M 2.3	BOOL	Behälter 1 ist leer
B3leeren	M 2.4	BOOL	Behälter 3 muss geleert werden
ENDE	M 2.5	BOOL	Zyklus ist zu Ende (neu starten)
HeizEnd2	M 3.0	BOOL	Heizzeit 2 zu Ende
B2_leer	M 3.3	BOOL	Behälter 2 ist leer
Verr_1	M 4.0	BOOL	Verriegelung Charge / Prog. 1
Verr_2	M 4.1	BOOL	Verriegelung Charge / Prog. 2
Merk_0	M 4.2	BOOL	kein Programm gewählt
MrkALeer	M 20.0	BOOL	Merker Taster <ALLES LEEREN> betätigt
R_IMP	M 30.0	BOOL	Rücksetzimpuls für Programmwahl
H_Zeit_1	T 1	TIMER	Heizzeit Behälter 1
H_Zeit_2	T 2	TIMER	Heizzeit Behälter 2
L_IMP	T 3	TIMER	Timer Rücksetzimpuls für Programmwahl

OB 1; Netzwerk 1: Es wurde noch kein Pfad gewählt



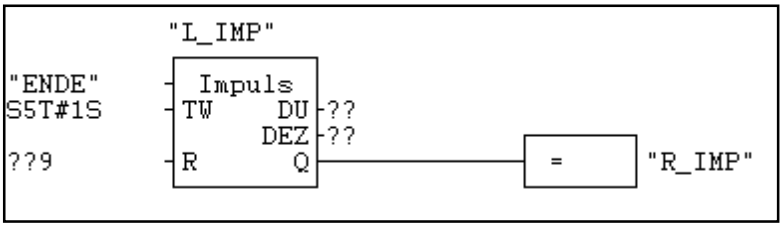
Im Zusammenhang mit NW 2 leuchten die START-LEDs, wenn alle Behälter leer sind (Ende des Programms, NW 3) und noch kein <START>-Taster gedrückt wurde. Siehe auch Netzwerk 4 und 5, in denen aus den Taster-Impulsen eine Dauer-„1“ entsteht.

OB 1; Netzwerk 2: Alle Behälter sind leer.



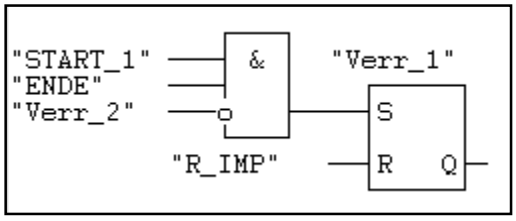
Wenn alle Behälter leer sind, wird der Merker, der mit <ALLES_LEEREN> gesetzt wurde, zurückgesetzt. Außerdem wird dadurch das Ende des Programmzyklus erkannt.

OB 1; Netzwerk 3: Rücksetzimpuls für Verriegelungen



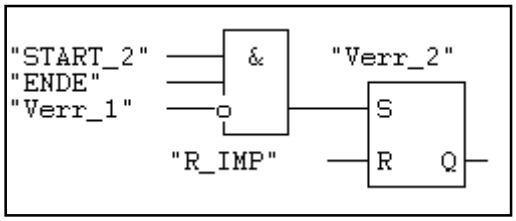
Wenn alle Behälter leer sind (<ENDE> = 1), wird mit einem Impuls im nächsten Netzwerk das vorgewählte Programm zurückgesetzt.

OB 1; Netzwerk 4: Nur Charge 1



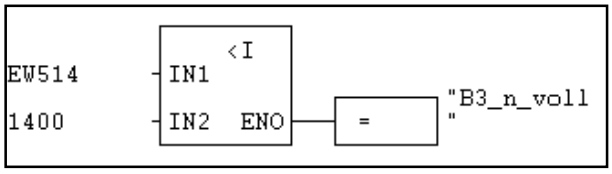
Sofern noch nicht die 2. Charge (<Verr_2>) gewählt wurde, kann mit dem Taster <START_1> der Verriegelungs-Merker für die 1. Charge gesetzt werden, wenn vorher die 3 Behälter leergefahren wurden. <Verr_1> verhindert bis zum Ende des Zyklus, dass das Programm gewechselt werden kann.

OB 1; Netzwerk 5: Nur Charge 2



Sofern noch nicht die 1. Charge (<Verr_1>) gewählt wurde, kann mit dem Taster <START_2> der Verriegelungs-Merker für die 2. Charge gesetzt werden, wenn vorher die 3 Behälter leergefahren wurden. <Verr_2> verhindert bis zum Ende des Zyklus, dass das Programm gewechselt werden kann.

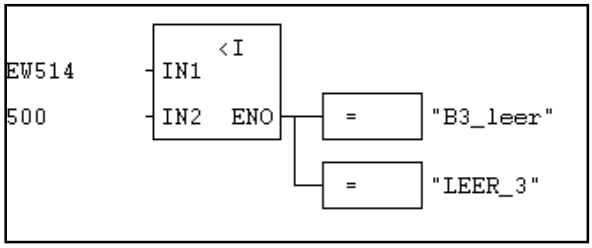
OB 1; Netzwerk 6: Wenn Behälter_3 nicht voll ist...



Falls der Wert von EW 514 kleiner als 1400 ist, bedeutet eine „1“ am Ausgang, dass der Behälter 3 noch nicht voll ist.

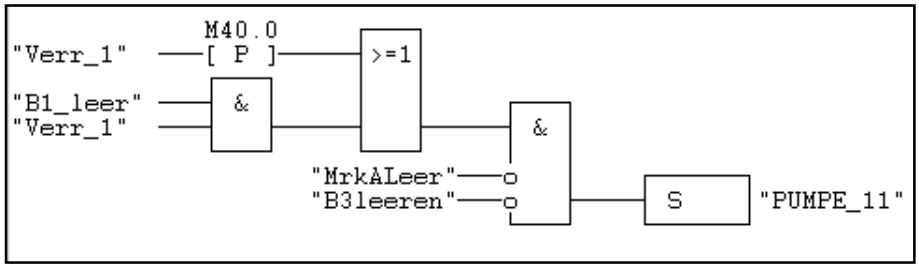
EW 514 ist der Messwert des Füllstandsmessers im Behälter 3. (Öffnen Sie bei ausgeschalteter Anlage mit <rM> auf den Füllstandsmesser oder den Text <Füllstand_3> das Editierfenster, um den Operanden zu sehen oder evtl. zu ändern).

OB 1; Netzwerk 7: Wenn Behälter 3 leer ist...



Der Wert des Füllstandsmessers wird mit der vorgegebenen Füllhöhe verglichen. Wenn die Füllhöhe < 500 ist, führt <B3_leer> eine „1“. Zusätzlich wird die LED eingeschaltet.

OB 1; Netzwerk 8: Lauf Pumpe 1.1 / Hand-Start und Automatik, wenn Behälter 3 nicht voll ist

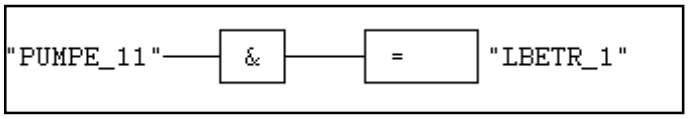


Aus dem Taster-Befehl <START_1> wurde in „Verr_1“ eine Dauer-„1“. Mit „Verr_1“ wird nicht nur die Charge 1 vorgewählt, sondern hier auch <Pumpe_11> auf „1“ gesetzt, sofern

- der Behälter 3 noch nicht voll und
- der Merker für den Vorgang <ALLES LEEREN> nicht gesetzt ist.

„Verr_1“ wird hier jedoch nicht als Dauer-„1“ benötigt, sondern soll nur 1x („von Hand“) den Füllvorgang starten. Die positive Flanke verhindert, dass die Pumpe immer wieder eingeschaltet wird. Ausgeschaltet wird die Pumpe im Netzwerk 10.
Der Füllvorgang beginnt automatisch wieder (ODER), wenn der Behälter 1 leer ist und mit „Verr_1“ die Charge 1 vorgewählt wurde.

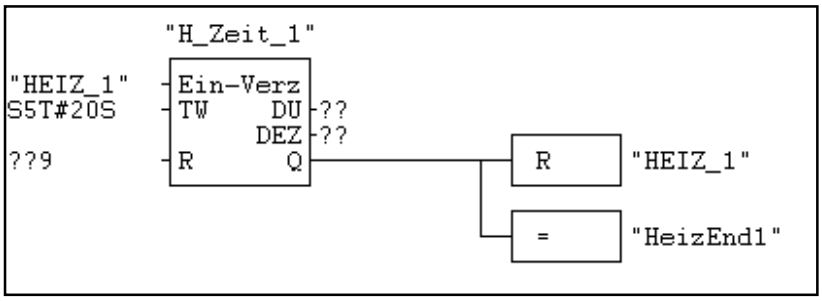
OB 1; Netzwerk 9: Anzeige Pumpe 1.1 in Betrieb



OB 1; Netzwerk 10: HEIZEN_1, wenn Behälter 1 voll ist; zeitbegrenzt

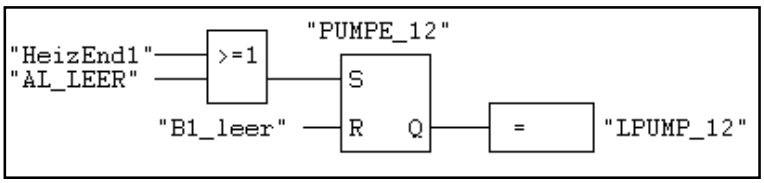
```
    SET              //Bedingungsloses Setzten des VKE auf „1“
    R  "HEIZ_1"      //Grundstellung: Heizung 1 AUS
    R  "VOLL_1"      //Grundstellung: LED <Behälter 1 ist voll> auf „0“
    L   EW  512      //Lade das Wort des Füllstandsmessers
    L   430          //Lade die Zahl 430
    >I               //Wenn der Füllstand höher als 430 ist...
    SPBN  ende       //.. springe (bedingt) zur Marke <ende:>
    R  "PUMPE_11"    //Sonst schalte die Pumpe 11 aus...
    =  "VOLL_1"      //... und schalte die LED ein.
    UN "HeizEnd1"    //Wenn die Heizzeit noch nicht beendet ist...
    =  "HEIZ_1"      //... wird die Heizung eingeschaltet.
ende: NOP 0          //Sprungmarke. Ende des Netzwerkes.
```

OB 1; Netzwerk 11: Heizzeit 1



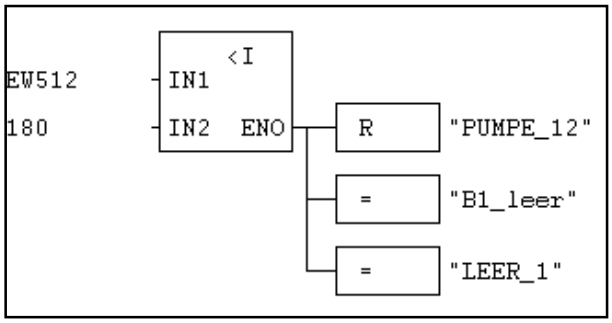
Bei eingeschalteter Heizung läuft die Zeit für die Heizdauer. Nach Ablauf der Zeit wird der Ausgang für die Heizung abgeschaltet und der Merkeroperand für das Ende der Heizzeit führt „1“.

OB 1; Netzwerk 12: Nach Heizende wird B_1 entleert



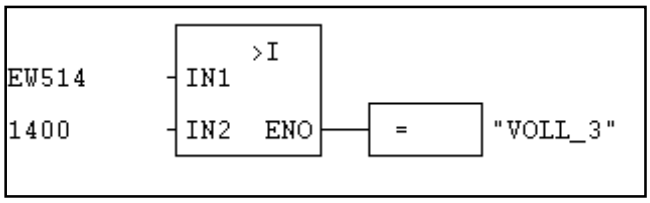
Wenn die Heizzeit abgelaufen ist oder wenn alle Behälter geleert werden sollen, wird die Pumpe 12 gesetzt, bis der Behälter 1 leer ist. Gleichzeitig wird die Pumpen-LED eingeschaltet.

OB 1; Netzwerk 13: B_1 ist leer



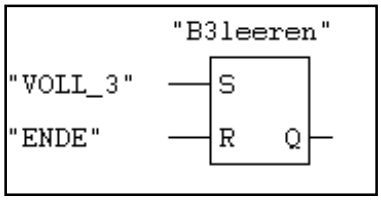
Wenn der Füllstandsmesser im Behälter 1 weniger als den Wert 180 misst, wird die Pumpe 12 ausgeschaltet. Der Merker zeigt mit „1“ an, dass der Behälter 1 leer ist. Zusätzlich zeigt die LED diesen Zustand an.
(Wenn Behälter 3 noch nicht voll ist, leuchtet die LED nur ganz kurz, weil sofort der Füllvorgang wieder beginnt und sich das Füllwort ändert).

OB 1; Netzwerk 14: B_3 ist voll



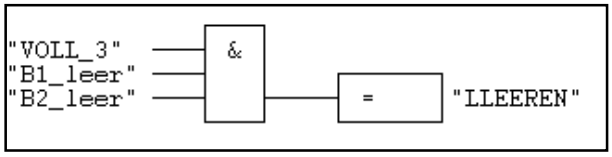
Der Wert des Füllstandsmessers wird mit der vorgegebenen Füllhöhe verglichen. Wenn die Füllhöhe >1400 ist, führt LED <VOLL_3> eine „1“.

OB 1; Netzwerk 15: Behälter_3 muss entleert werden



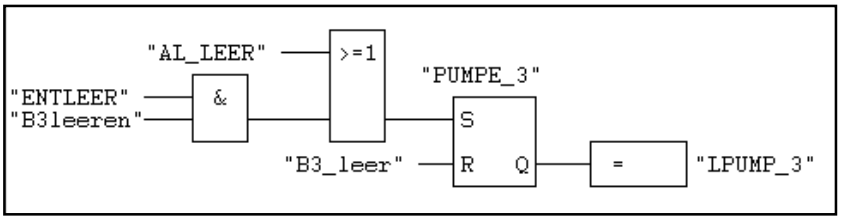
Die erreichte „1“ von <VOLL_3> wird gespeichert. Auch wenn der Behälter 3 geleert wird und <VOLL_3> erlischt, bleibt „B3leeren“ auf „1“, bis alle 3 Behälter leer sind, d.h., bis „ENDE“ eine „1“ liefert.

OB 1; Netzwerk 16: LED "ENTLEEREN möglich"



Wenn die Vorbehälter 1 und 2 leer sind und Behälter 3 voll ist, zeigt die LED an, dass jetzt Behälter 3 entleert werden kann / muss.
(Alternativ kann auch <ALLES LEEREN> betätigt werden.

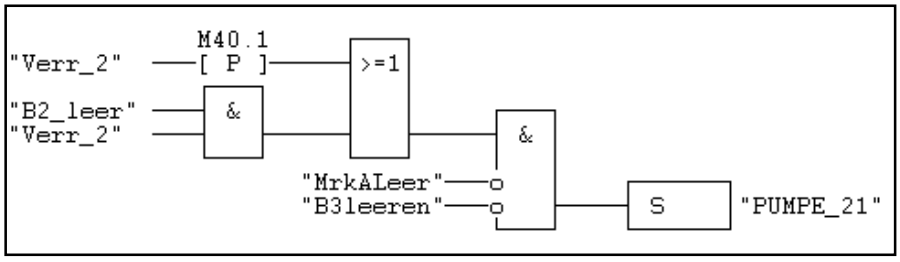
OB 1; Netzwerk 17: Entleeren mit Pumpe 3



Mit <PUMPE_3> wird der Behälter 3 geleert, bis <B3_leer> eine „1“ liefert, wenn

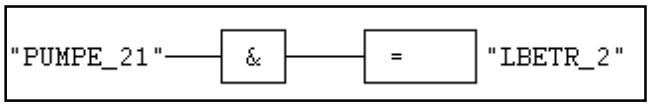
- <ALLES LEEREN> gedrückt wird, oder
- „B3leeren“ über „VOLL_3“ gesetzt wurde und <BEH_3 ENTLEEREN> gedrückt wird.

OB 1; Netzwerk 18: Lauf Pumpe 2.1 Hand-Start und Automatik, wenn Behälter 3 nicht voll ist



Für die 2. Charge wird sinngemäß zu NW 8 die erste Pumpe dieses Zweiges angesteuert.

OB 1; Netzwerk 19: Anzeige Pumpe 2.1 in Betrieb

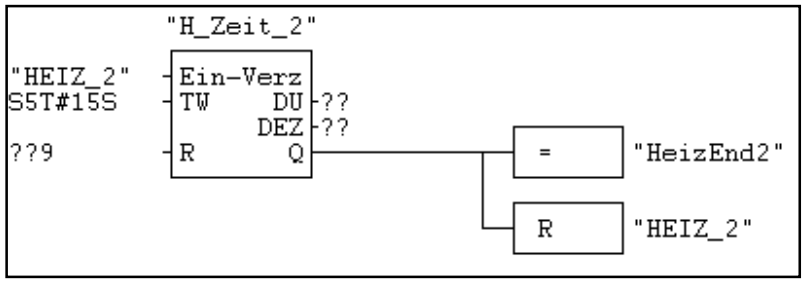


Bei eingeschalteter Pumpe leuchtet die LED.

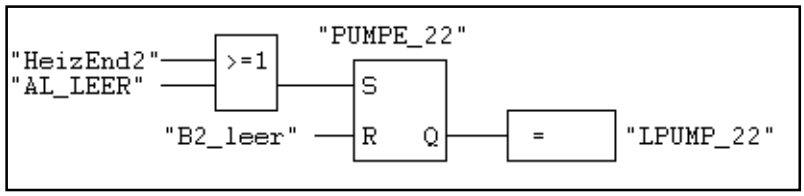
OB 1; Netzwerk 20: Heizen 2, wenn Behälter 2 voll ist; zeitbegrenzt

```
    SET                 // (vergl. NW 10)
    R  "HEIZ_2"
    R  "VOLL_2"
    L   EW  516
    L   330
    >I
    SPBN  end2
    R  "PUMPE_21"
    =  "VOLL_2"
    UN "HeizEnd2"
    =  "HEIZ_2"
end2: NOP 0
```

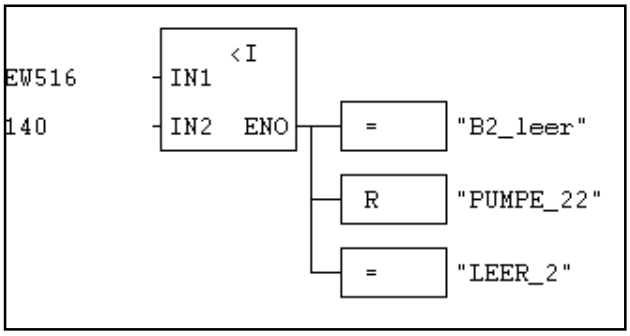
OB 1; Netzwerk 21: Heizzeit 2



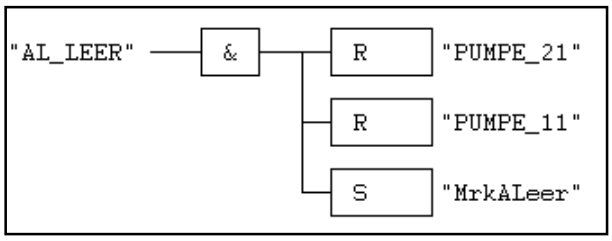
OB 1; Netzwerk 22: Anzeige Pumpe 2.2 läuft



OB 1; Netzwerk 23: Behälter 2 ist leer

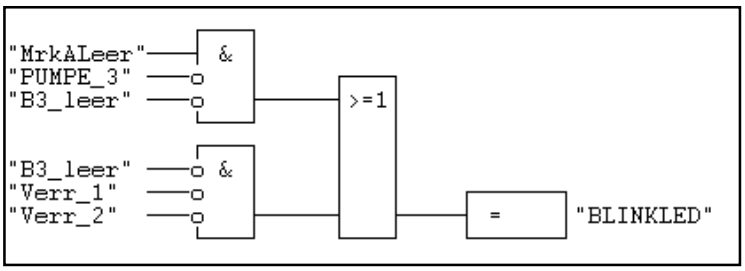


OB 1; Netzwerk 24: <ALLES_LEEREN>-R / S



Mit dem Taster <ALLES LEEREN> werden die Zulaufpumpen für die Vorbehälter ausgeschaltet und aus der Taster-„1“ wird eine Dauer-„1“. „MrkALeer“ verhindert, dass die Zulaufpumpen wieder anlaufen,wenn die Vorbehälter leer sind und die Chargen-verriegelung noch besteht.

OB 1; Netzwerk 25: Hinweis: Behälter 3 leeren

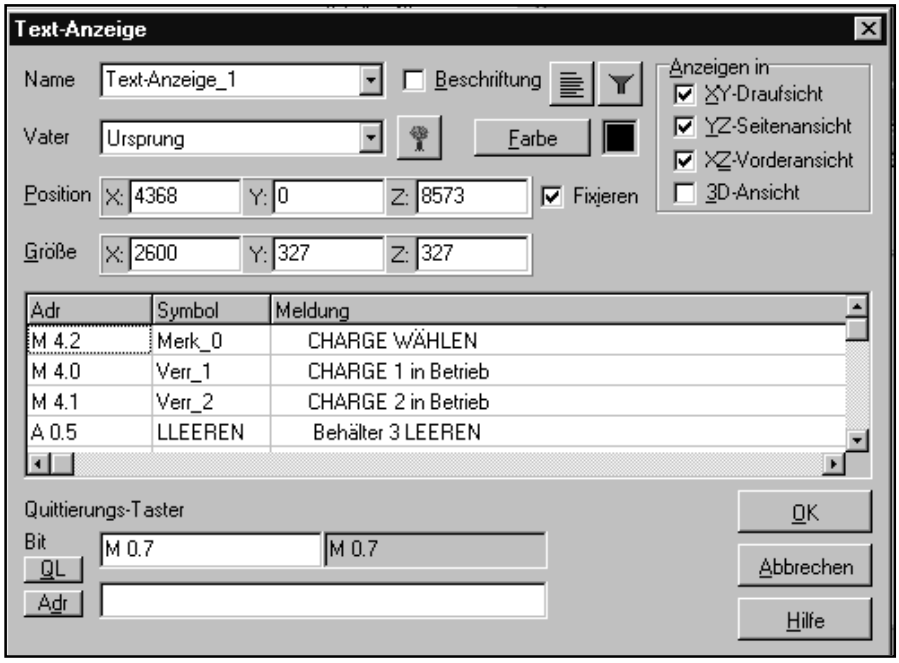


Wenn z.B. beim ersten Füllvorgang des Vorbehälters <ALLES LEEREN> betätigt wird und Behälter 3 noch leer ist, läuft Pumpe_3 nicht an. Von oben läuft aber die Flüssigkeit zu. Behälter 3 kann bei dieser Situation nicht leergepumpt werden. Es muss nochmals <ALLES LEEREN> gedrückt werden. Als Hinweis dafür wird die LED <BLINKLED> eingesetzt.
Die LED leuchtet, wenn

- die Taste <ALLES LEEREN> betätigt wurde, die Pumpe 3 steht und Behälter 3 nicht leer ist, oder
- Behälter 3 nicht leer ist und noch kein Chargenprogramm gewählt wurde (gewählt werden kann).

Diese Situation können Sie „einfrieren“, wenn Sie bei gefülltem Vorbehälter auf <ALLES LEEREN> und gleich danach [Umschalt+F2] zum Speichern drücken. Schließen Sie dann TrySim. Beim Neustart des Programms sind dann die Behälter – noch – gefüllt und der Hinweis auf <ALLES LEEREN> müsste erscheinen. Sonst würde der Bediener grübeln, warum die Anlage nicht zu starten ist.
Ich denke, an diesem und vielen anderen Projekten wird deutlich, wie unermesslich hilfreich TrySim beim Aufspüren von Detailproblemen sein kann.

Im Textfenster der Anlage werden situationsabhängig verschiedene Texte angezeigt. Die Texte werden mit entsprechenden Operanden dem Fenster zugeordnet. Mit <rM> auf den Rahmen öffnet sich das Editorfenster.



Wählen Sie dort <Hilfe>, um mehr zu erfahren.

Vgl. TrySim-Projekt <Mischer> im Verzeichnis <Lösungen>.

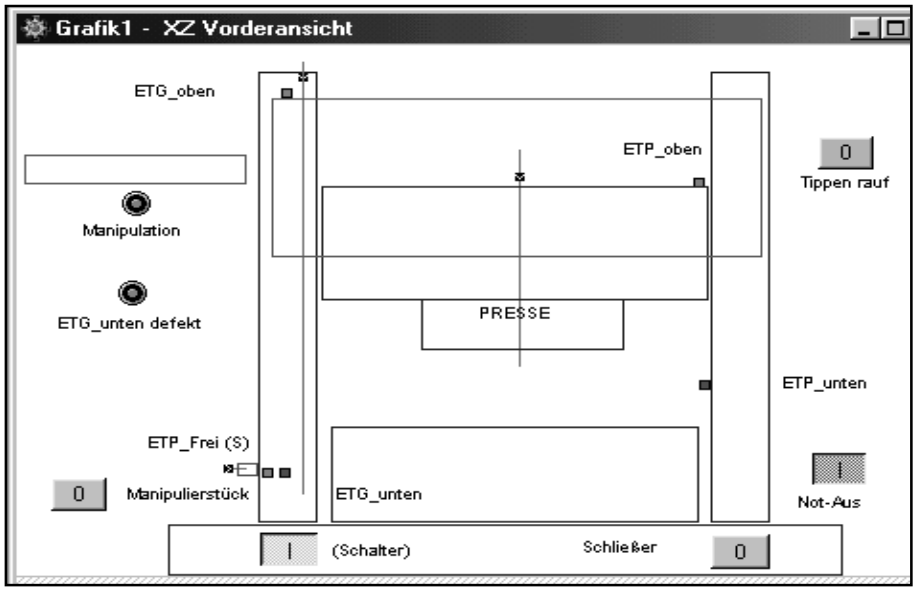
Projekt 36: Pressensteuerung mit Schutzgitter

Schwerpunkte: Vertiefung der Anwendung von Öffnern und Schließern. Sicherheitsbetrachtungen.

Aufgabe: Pressensteuerung mit Schutzgitter
Zu planen ist eine sichere Steuerung für den kompletten Zyklus einer Presse.

Theorie: Aus Sicherheitsgründen muss für eine Pressensteuerung eine Zweihand-Steuerung zwingend vorgesehen werden, damit der Akkordarbeiter nicht mit einer Hand das Teil einlegt, während er mit der anderen Hand schon den Stempel senkt. Da in der Simulation immer nur ein Taster mit der Maus bedient werden kann, wird ausnahmsweise der linke Taster (Schließer) einer 2-Hand-Schaltung in einen Schalter umgewandelt.

Bevor sich der Pressenstempel senken darf, muss zusätzlich zur Zweihand-Steuerung das Schutzgitter geschlossen sein. Somit wird eigentlich das Schutzgitter zweihandgesteuert. Nach dem Schließen des Schutzgitters senkt sich der Pressenstempel automatisch. Beim Erreichen des unteren Endtasters (Öffner) wird der Pressenantrieb abgeschaltet. Wird einer der beiden EIN-Taster losgelassen (oder beide), fährt der Stempel, gefolgt vom Gitter, wieder nach oben.

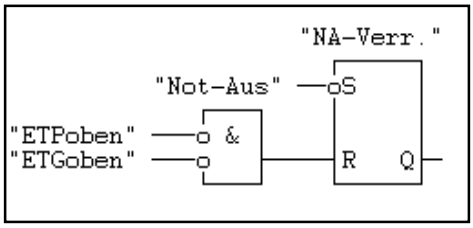


Symbolische Adresse	Operand	Datentyp	Kommentar
PRrunter	A 0.0	BOOL	Antrieb Presse nach unten
PRrauf	A 0.1	BOOL	Anntrieb Presse nach oben
G_runter	A 0.2	BOOL	Antrieb Gitter runter
G_rauf	A 0.3	BOOL	Antrieb Gitter rauf
Manipul	A 0.4	BOOL	Manipulation am Pressen-EIN-Taster
ETG_uTST	A 0.5	BOOL	Endtaster Gitter u. Überwachung
EIN_L	E 0.0	BOOL	Eintaster links (Schalter)
EIN_R	E 0.1	BOOL	Eintaster rechts (S)
ETPunten	E 0.3	BOOL	Endtaster unten (Ö)
ETPoben	E 0.4	BOOL	Endtaster oben (Ö)
ETGunten	E 0.5	BOOL	Endtaster Gitter unten (Ö)
ETGoben	E 0.6	BOOL	Endtaster Gitter oben (Ö)
ETP_Frei	E 0.7	BOOL	Endtaster Presse ist frei
Not-Aus	E 1.0	BOOL	Not-Aus
NA-Verr.	M 0.1	BOOL	Not-Aus-Verriegelung
Tipprauf	E 1.1	BOOL	Presse u. Gitter nach oben
PressMerk	M 2.0	BOOL	Stempel war unten

Zur Umsetzung der Aufgabe mit TrySim:

Wählen Sie bitte aus dem Ordner **<Vorlagen>** das Projekt **<Presse_V>** und speichern Sie es als neues Projekt in Ihrem eigenen Verzeichnis.

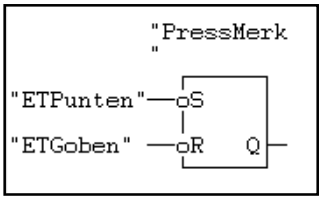
OB 1; Netzwerk 1: Not-Aus-Verriegelung



Zwingende Bedingung ist, dass bei Not-Aus die Anlage in jeder Stellung gestoppt wird.
Nach dem Öffnen der Raste im Not-Aus-Taster darf die Anlage nicht von allein anlaufen.

<NOT-AUS> ist in diesem Projekt als Schalter ausgeführt. Das mag etwas überraschen, ist aber logisch, wenn man die Bedienelemente von TrySim berücksichtigt: Man kann Schließer, Öffner und Schalter auswählen. Die Abfrage der <NOT-AUS>-Eingangsklemme muss wegen der Drahtbruchsicherheit auf "1" lauten. Dafür wurde bisher immer ein Öffner ausgewählt, wie es in der Praxis für die Funktion <AUS> auch richtig ist. Allerdings gibt es bei TrySim keinen "rastenden" Öffner, wie es ein NOT-AUS-Schalter ist. Deshalb wird hier als Variante einmal ein Schalter ausgewählt, der beim Starten der Anlage allerdings "0" anzeigt, also ausgeschaltet ist. Um einen NOT-AUS-Schalter darzustellen, muss er für den Betriebsfall erst einmal eingeschaltet werden. Im NOT-AUS-Fall wird dieser Schalter ausgeschaltet und bleibt bis zur Entriegelung in dieser Stellung. Das ist der Unterschied zum Öffner, der ja sofort nach dem Loslassen wieder logisch "1" vorgibt.
Auch Motorschutzschalter oder Sicherungsautomaten entsprechen diesem Prinzip: Sie müssen zuerst eingeschaltet werden und schalten bei Überlast dauerhaft aus.
Bei betätigtem NOT-AUS darf nichts zu schalten sein. Nach der Entriegelung darf nichts eigenständig anlaufen. Auch ein Neustart sollte nur aus der Grundstellung heraus möglich sein. Deshalb wird dieser Merker benutzt, der bei **und nach** NOT-AUS alle Funktionen blockiert - außer den Tippbetrieb in die Grundstellung.
Aufgehoben wird diese "Blockade" erst, wenn die Presse und das Schutzgitter in der Grundstellung sind. Beide Endtaster sind wegen der Drahtbruchsicherheit als Öffner ausgeführt. Bei der Betätigung liefern sie "0"-Signal.

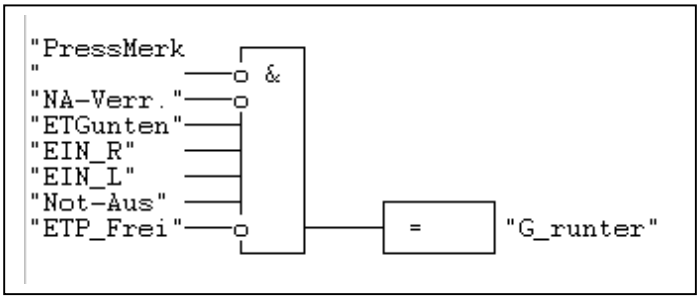
OB 1; Netzwerk 2: Stempel_unten_Merker



Dieser Merker verhindert, dass nach dem Pressen erneut der Pressenstempel oder das Gitter gesenkt werden können. Er wird durch den unteren Endtaster der Presse gesetzt und durch den oberen Endtaster des Gitters zurückgesetzt.

OB 1; Netzwerk 3: Gitter fährt runter

Zuerst scheint es auszureichen, wenn bei der Zweihandbetätigung das Schutzgitter bis zum unteren Öffner "ETG_unten" herunterfährt. Da es TrySim z.Z. nicht ermöglicht, 2 Taster innerhalb von 0.5 s gleichzeitig und dauerhaft zu betätigen - wie es die Berufsgenossenschaft vorschreibt - wird für die Simulation der <EIN>-Taster für die linke Hand ausnahmsweise als Schalter ausgeführt. **Folglich wird in dieser Simulation auch auf die zwingend vorgeschriebene Kontrolle des "0" / "1"-Überganges an beiden Tastern verzichtet!**
Programmiert wird immer die Einschalt- und nicht die Ausschaltbedingung. Somit werden diese Taster auf "1" abgefragt. Das gilt auch für "ETG_unten". Wird dieser Taster durch das Gitter betätigt, öffnet sich der Stromkreis. Falls an dieser Stelle ein Schließer eingebaut wäre, würde man bei Drahtbruch kein Signal "1" für das Ausschalten erhalten.



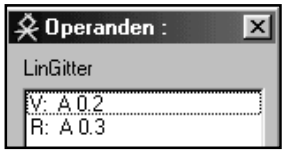
Das Gitter fährt runter, wenn

- "PressMerk" noch nicht vom unteren Pressenendtaster gesetzt wurde
- keine NOT-AUS-Situation vorlag (Verriegelung)
- der NOT-AUS-Taster "entriegelt" ist, d.h. in Grundstellung "1" liefert
- das Gitter noch nicht unten angekommen ist
- und beide <EIN>-Taster betätigt sind.

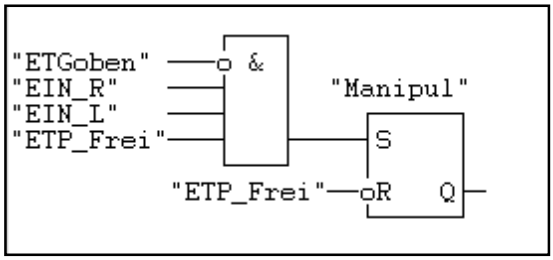
Der Taster "ETGunten" könnte prinzipiell auch das Signal für das Einschalten des Pressenvortriebs (senken) geben. Jedoch denken Sie bitte auch hier an einen Drahtbruch: "0" bei Drahtbruch würde gleichgesetzt mit "0" durch <Schutzgitter unten>. Das wäre dann das Startsignal für den Pressenstempel. Dieser könnte dann jederzeit bei geöffnetem Gitter herunterfahren!

Deshalb wird zur Freigabe des Pressenvortriebs ein zweiter Kontakt benötigt. Normalerweise wäre das durch eine zweite Schaltebene desselben Endtasters zu erreichen. Hier in der Simulation kann diese Bedingung durch einen zusätzlichen Schaltkontakt - auf gleicher Höhe angebracht - erreicht werden.
Zum Ausschalten dient der Öffner "ETG_unten" und zum Einschalten der Presse der Schließer "ETP_Frei". Die Abfrage von "ETP_Frei" wäre für das Gitter eigentlich nicht erforderlich. Sie verhindert aber schon das Absenken des Gitters, wenn an "ETP_Frei" eine Störung oder Manipulation vorliegt. Es könnte nämlich sein, dass ein vermeintlich pfiffiger Arbeiter "ETP_Frei" in gedrückter Stellung festklebt, um ein abgesenktes Gitter vorzutäuschen und somit die Presse schneller bedienen zu können. Der verantwortliche Planer muss all diese Möglichkeiten und auch den denkbar schlimmsten Störungsfall in die Sicherheitsüberlegungen mit einbeziehen. Hier muss der fehlenden Einsicht schon vorbeugend durch das nächste Netzwerk Einhalt geboten werden.

Hinweis zur Zuordnung der Ausgangsoperanden:
Die Zuordnung zum richtigen Antrieb der Anlage - also zum entsprechenden Ausgangsoperanden - erreichen Sie bei ausgeschalteter Anlage, indem Sie den Cursor auf den Strich des Linearantriebs "LinGitter" in der linken Säule legen. Mit <LM> können Sie den zugehörigen Operanden an den Ausgang des Neztwerkes legen. Es öffnet sich ein Auswahlfenster für beide Antriebsrichtungen. "Runter" ist hier "**V**orwärts" und "hoch" ist "**R**ückwärts". Klicken Sie auf "V: A 0.2".

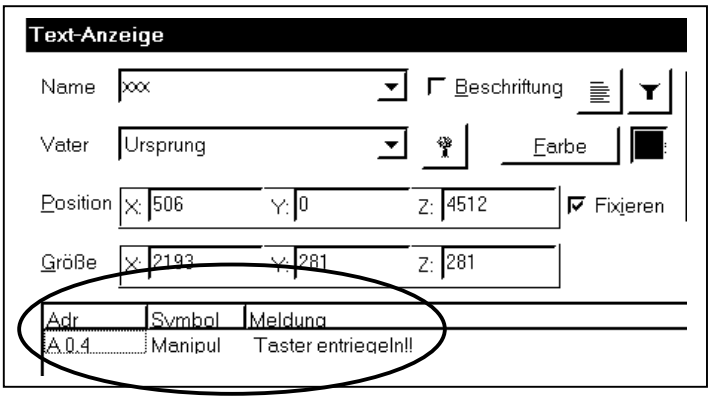


OB 1; Netzwerk 4: Feststellung der Manipulation am Freigabetaster (Gitter unten)



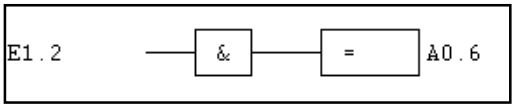
Falls "ETP_Frei" für den Pressenstempel schon "1" führt, wenn das Schutzgitter noch oben ist, wird der Manipulationsmerker gesetzt. Erst wenn die Manipulation beendet ist, wird der Merker zurückgesetzt.

Man kann sogar soweit gehen, diesen Manipulationsversuch anzuzeigen. Eine Startfreigabe ist nur möglich, wenn <ETP_Frei> "0" führt, d.h. zu Beginn unbetätigt ist. Bei einer Manipulation wird eine LED und ein Textfenster (oben links) eingeschaltet. (Textfenster-Rahmen <rM>-Klick. Operand und Text sind wählbar).



Um diese Manipulation testen zu können, ist in der Anlage schon eine kleine Einheit eingebaut worden. Diese finden Sie komplett programmiert im OB **2**, Netzwerk 1.
(vgl. die Erläuterungen zu den **O**rganisations**B**austeinen unter: TrySim <Hilfe> <Index> "OB").
Dieser Baustein OB 2 wird nur in der Simulation aufgerufen, muss also für die reale Anlage nicht entfernt werden.

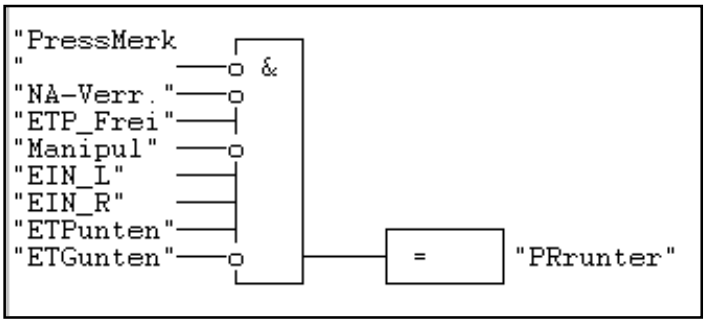
OB 2; Netzwerk 1: Manipulation des Schutzgitter-Tasters



Wenn der Schalter <Manipulierstück> (= E 1.2) betätigt wird, fährt der Linearbeweger <LinManipu> (= A 0.6) ein Kästchen auf "ETP_Frei".
Sie können mit dem bisherigen Programm diese Situation schon testen.

Bei abgesenktem Schutzgitter kann jetzt der Pressenantrieb freigegeben werden.

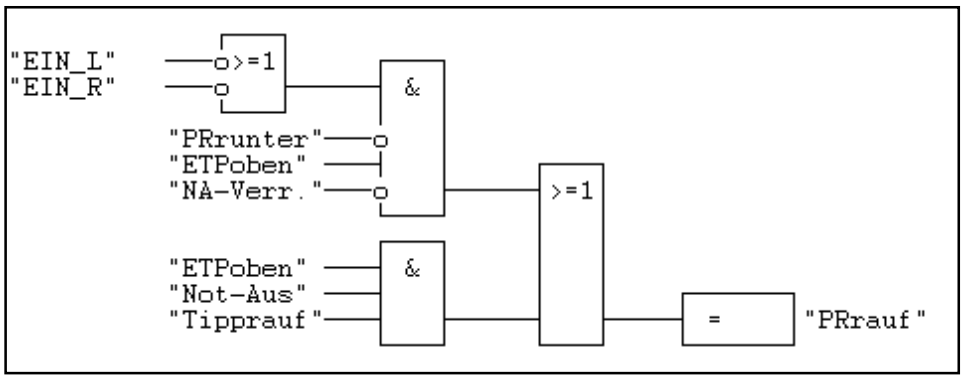
OB 1; Netzwerk 5: Presse fährt runter



Der Pressenstempel "LinPresse" fährt runter, wenn

- er noch nicht unten war
- keine NOT-AUS-Situation vorlag
- Der Freigabetaster vom Gitter angefahren wurde
- Keine Manipulation vorliegt
- Beide <EIN>-Taster gedrückt sind
- Das Schutzgitter unten ist
- und solange der untere Endtaster der Presse "ETPunten" nicht angefahren wurde.

OB 1; Netzwerk 6: Presse fährt rauf



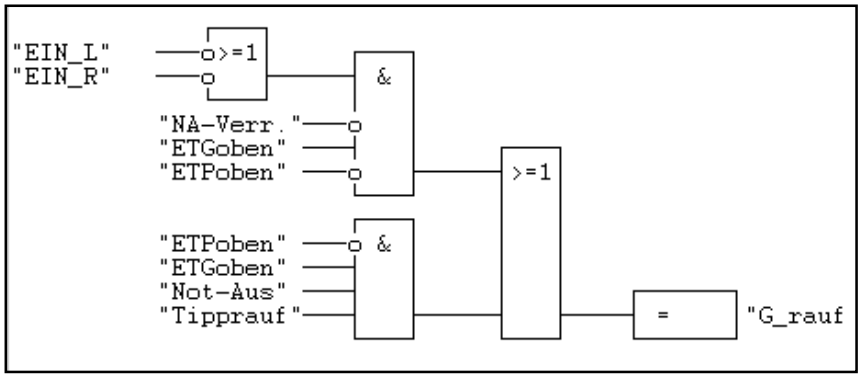
Der Pressenstempel fährt hoch, wenn

- einer oder beide >EIN>-Taster losgelassen werden
- der Antrieb nicht runter fährt
- der obere Endtaster noch nicht erreicht wurde
- und keine NOT-AUS-Situation vorliegt

oder wenn nach NOT-AUS

- <NOT-AUS> wieder entriegelt wurde
- <Tippen rauf> betätigt wird
- und der obere Endtaster noch nicht angefahren wurde

OB 1; Netzwerk 7: Gitter fährt rauf



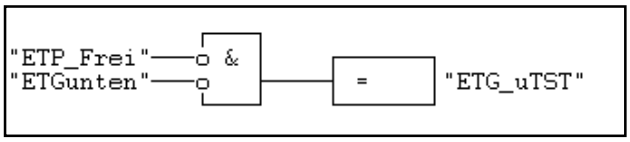
Das Gitter fährt nach oben, wenn

- einer oder beide >EIN>-Taster losgelassen werden
- und kein NOT-AUS-Fall vorlag
- und der Pressenstempel schon oben ist
- und das Gitter noch nicht oben ist

oder wenn nach NOT-AUS

- der Pressenstempel oben ist
- und das Gitter noch nicht oben ist
- und <NOT-AUS> wieder entriegelt wurde
- <Tippen rauf> betätigt wird

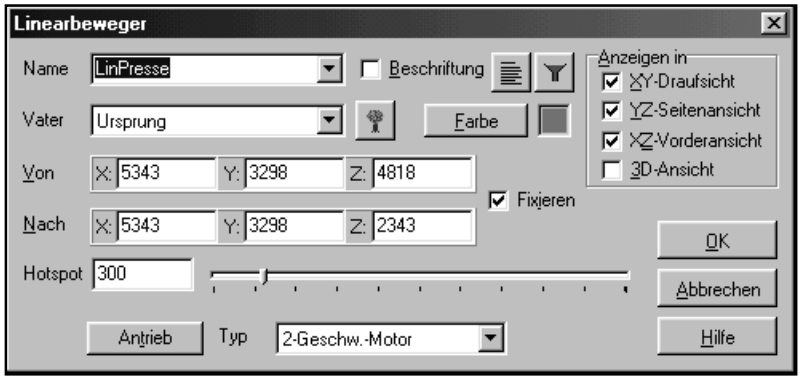
OB 1; Netzwerk 8: Test der Funktion von <ETGunten>



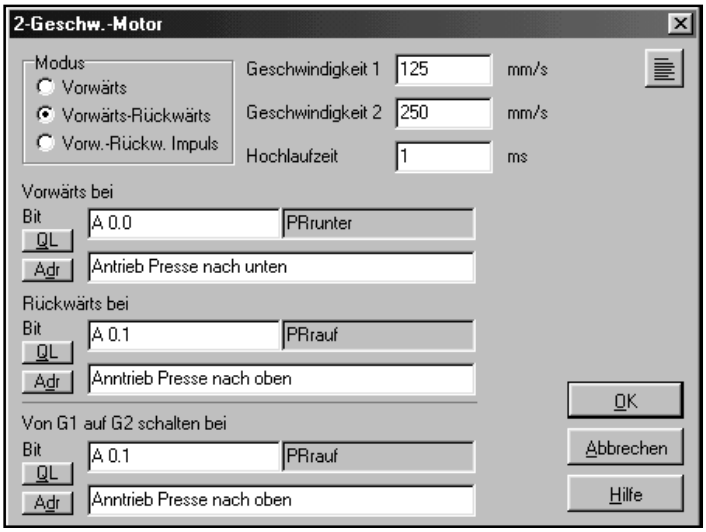
Es könnte auch sein, dass der untere Gittertaster <ETGunten> oder dessen Leitung eine Unterbrechung hat. Auch dieser Zustand kann als Fehlermeldung angezeigt werden, damit der Störungsdienst zielgerichtet den Fehler beseitigen kann, ohne lange zu suchen, warum die Anlage "schon wieder nicht läuft". Diese Fehlerfunktion können Sie mit <rM> auf <ETGunten> und Änderung im Editierfenster von <Typ> in <Schließer> kontrollieren.
"ETG_uTST" ist als LED ausgeführt.

(Wenn Sie nicht wissen, welcher der beiden Endtaster der richtige ist, klicken Sie auf den entsprechenden Text. Das zugehörige Symbol wird dann auch gekennzeichnet).

Beim Testen des Programms wird Ihnen sicherlich auffallen, dass der Pressenstempel langsamer runter- als rauffährt. Wie kommt das? Hierfür bietet TrySim Einstellmöglichkeiten. Das entsprechende Editierfenster für den Linearantrieb "LinPresse" öffnen Sie bei ausgeschalteter Anlage mit <rM> auf den mittleren, senkrechten Strich im Pressenstößel.



Sie sehen, dass ein 2-Geschwindigkeiten-Motor ausgewählt wurde. Mit <LM> auf <Antrieb> öffnet sich ein weiteres Fenster:



Sie können hier 2 Geschwindigkeiten einstellen. Außer den beiden Operanden für "vorwärts" und "rückwärts" kann ein Operand gewählt werden, mit dem die Geschwindigkeit 1 auf Geschwindigkeit 2 umgeschaltet wird. Da "A 0.1" (rückwärts) gewählt wurde, ist das gleichzeitig der Befehl für die erhöhte Rückwärtsgeschwindigkeit.

Vgl. TrySim-Projekt <Presse> im Verzeichnis <Lösungen>.

Sie haben gelernt:

- Öffner dienen zur unkritischen Überwachung von Grenzzuständen.
- Ein Drahtbruch kommt einem Abschalten gleich.
- Weiterschaltbedingungen müssen mit (separaten) Schließern erfüllt werden.
- Kritische Zustände sind zur Sicherheit mit Doppeltastern zu überwachen.
- Bei Not-Aus muss ggf. die ganze Anlage sofort anhalten.
- Nachdem Not-Aus entriegelt wurde, darf kein Teil der Anlage selbständig anlaufen.
- Kontroll- und Diagnosenetzwerke verhindern Fehlschaltungen, bzw. minimieren die Fehlersuche.

Projekt 37: Ampelsteuerung 1

Schwerpunkte: Programmierung nach Tabellen, Ablaufsteuerung, Timer

Aufgabe: Das Programm für eine Fußgängerampel ist zu erstellen. Für die zwingende Abfolge der Lampenschaltung bietet sich die Ablaufsteuerung mit S- und R-Funktionen an. Außerdem soll der Programmablauf, d.h. das Zusammenspiel der Ampelleuchten, als Bytemuster entworfen werden.

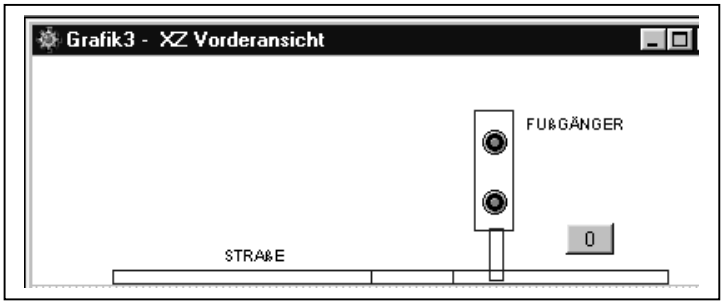
Im Verzeichnis <Vorlagen> ist das Projekt unter <Ampel_Tab_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

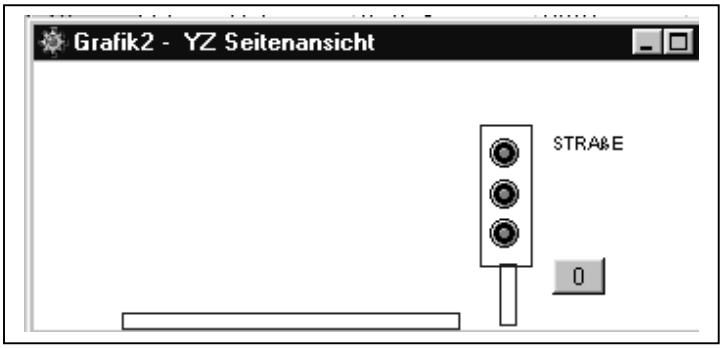
Die Symboltabelle und die Anlage sind bereits vorbereitet.

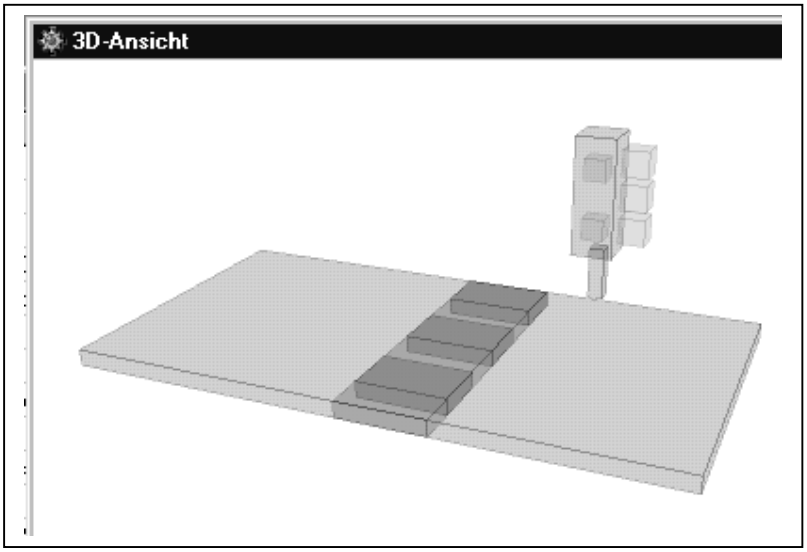
Symbolische Adresse	Operand	Datentyp	Kommentar
FA_rot	A 0.0	BOOL	Fußgängerampel ist rot
FA_grun	A 0.2	BOOL	Fußgängerampel ist grün
AA_rot	A 0.3	BOOL	AutoAmpel ist rot
AA_gelb	A 0.4	BOOL	AutoAmpel ist gelb
AA_grun	A 0.5	BOOL	AutoAmpel ist grün
START	E 0.0	BOOL	Fußgänger will rüber
Zyklus	M 10.1	BOOL	Merker Ampel-Zyklus läuft
STARTVER	M 10.0	BOOL	nur 1x eintasten
MAA_gelb	M 10.2	BOOL	Merker AA gelb
MAA_rot	M 10.3	BOOL	Merker AA rot
MFAgrun	M 10.4	BOOL	Merker Fußg.Ampel grün
MFA_rot	M 10.5	BOOL	Merker Fußg.Ampel rot
MAArtglb	M 10.6	BOOL	Merker AA rot + gelb

Erläuterungen:
„FA...“ steht für Fussgänger-Ampel
„AA...“ steht für Auto-Ampel
Diese Ausgänge werden nicht einzeln programmiert, sondern gemeinsam als Bitmuster – aber erst ganz zum Schluss des Programms. Vorher werden zum Ablauf der Schritte nur Merker („MFA...“ und „MAA...“) verwendet. Die Weiterschaltung von Schritt zu Schritt - und damit die Zeiten der Ampelphasen - wird von Zeitgliedern bestimmt.
Prinzip einer linearen Ablaufsteuerung ist, dass immer nur ein Schritt (Schrittmerker) aktiv ist. Es werden nur die Weiterschaltbedingungen der jeweiligen Schritte programmiert, aber noch nicht die Ausgänge selbst. Der jeweils aktive Schritt schaltet den vorher aktiven Schritt aus. Somit wird durch jedes Zeitglied ein Merker gesetzt und der vorherige Merker zurückgesetzt. Dieses Prinzip wird konsequent angewendet, ohne dass man sich Gedanken machen muss, wodurch die richtigen Ampellampen aktiviert werden. Erst zum Schluss des Programms werden die Schrittmerker abgefragt und dort werden die Ausgänge zugewiesen.

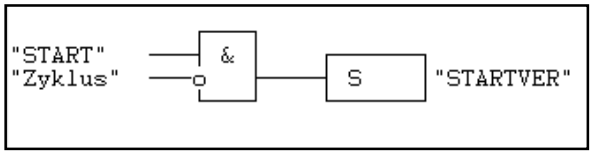
Ampelanlage:





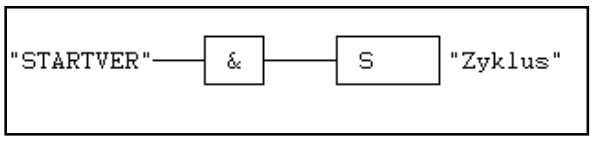


OB 1, NW 1: Startverriegelung, kein Nachtriggern möglich



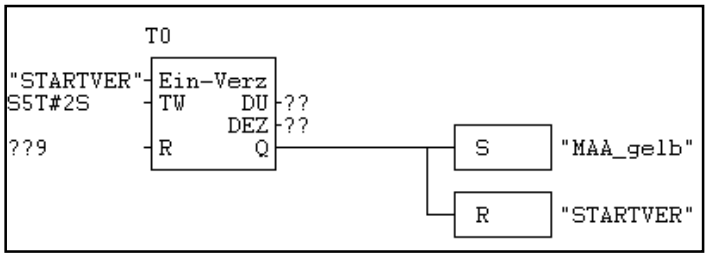
Bei Beginn des Ablaufs ist das folgende Netzwerk noch auf „0“ gesetzt, da „Zyklus“ erst durch „STARTVER“ gesetzt wir. Somit kann „STARTVER“ nur einmal mit <START> gesetzt werden. Danach liefert „Zyklus“ eine „1“. „Zyklus>“ wird erst am Ende des Programmdurchlaufs zurückgesetzt, so dass zwischendurch kein erneuter Startbefehl wirksam werden kann.

OB 1, NW 2: Zyklus läuft



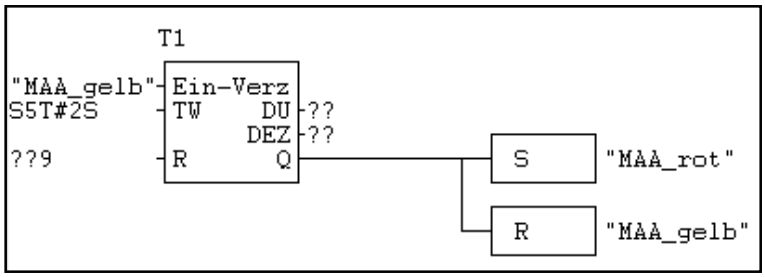
Dieses Netzwerk dient nur zur Verriegelung und ist kein Bestandteil der Schrittkette.

OB 1, NW 3: Merker Auto-Ampel gelb



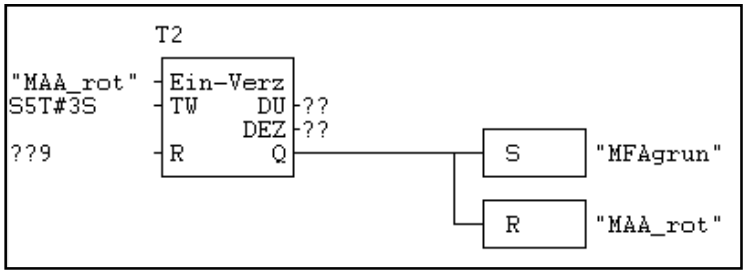
„STARTVER“ liegt als Dauer-„1“ an, bis T0 einschaltverzögert eine „1“ liefert. Dann wird der erste Schrittmerker „MAA_gelb“ gesetzt und „STARTVER“ zurückgesetzt. Dadurch wird gleichzeitig T0 abgeschaltet, weil am Eingang „0“ anliegt.

OB 1, NW 4: Merker Auto-Ampel rot



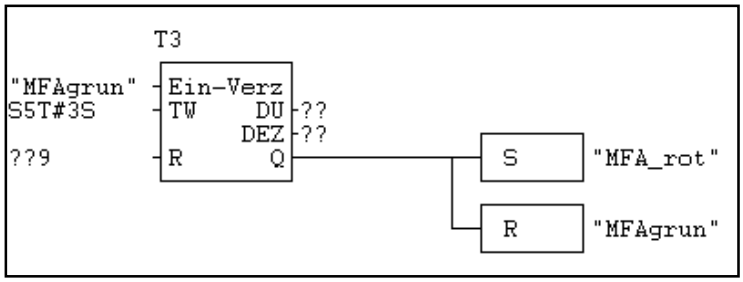
„MAA_gelb“ liegt als Dauer-„1“ an, bis T1 einschaltverzögert eine „1“ liefert. Dann wird der zweite Schrittmerker „MAA_rot“ gesetzt und „MAA_gelb“ zurückgesetzt. Dadurch wird gleichzeitig T1 abgeschaltet, weil am Eingang „0“ anliegt.

OB 1, NW 5: Merker FG-Ampel grün

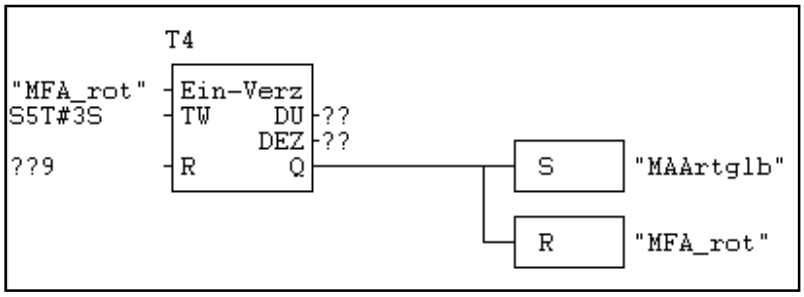


Das Prinzip des Netzwerkaufbaus ist identisch mit den vorausgegangenen Netzwerken.

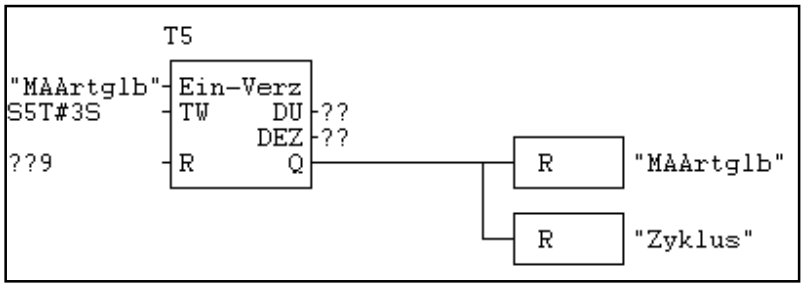
OB 1, NW 6: -Merker FG-Ampel wieder rot



OB 1, NW 7: Merker AutoAmpel rot / gelb



OB 1, NW 8: Merker AutoAmpel grün / Zyklus Ende

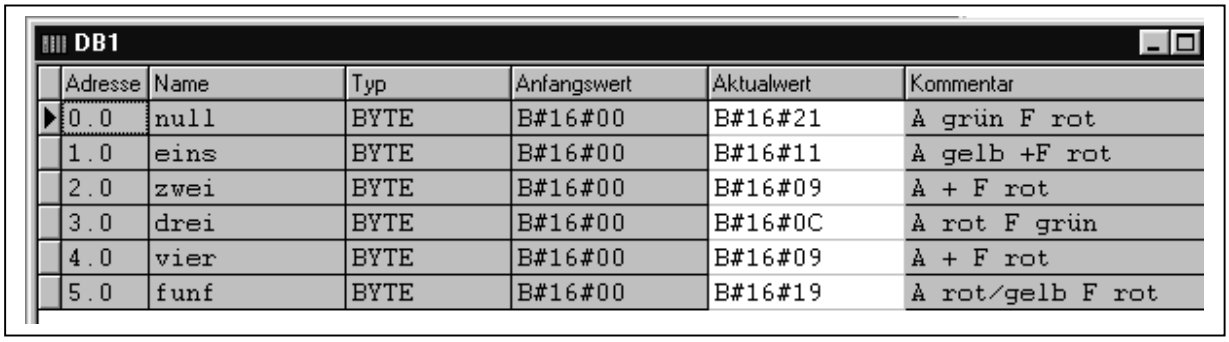


Jetzt sind alle Schritte abgearbeitet worden. Sie haben bestimmt schnell die immer wiederkehrende Struktur nachvollziehen können. Übrig bleibt noch die Zuweisung der Ausgänge. Man könnte natürlich jedem Schrittmerker alle jeweils aktiven Ausgänge zuordnen. In diesem Beispiel soll jedoch – wie schon eingangs erwähnt – eine andere Methode vorgestellt werden.
In der Symboltabelle wurden die Ampelleuchten bereits den Ausgangsoperanden zugeordnet. Für Übungszwecke – oder als gedankliche Anregung – wurde die „Klemme“ A 0.1 nicht belegt. In der folgenden Tabelle wurden die Funktionen der Lampen den „Klemmen“ zugeordnet. Sie wissen bereits, dass die Operanden A 0.0 bis A 0.7 auch gemeinsam als AB 0 adressiert werden können. Hier setzen wir bei der Wertzuordnung an. In der Grundstellung zeigt die Fußgängerampel rot und die Autoampel grün. Folglich muss „1“ an A 0.0 und A 0.5 liegen. (Alle anderen Ausgänge führen „0“-Signal).
Gemäß der Dualzahlen-Wertigkeit entspricht diese Kombination der Zahl „33“. Hexadezimal codiert entspricht das der „21“ hex. Dieser Wert wird in eine Zeile des DB 1 mit „B#16#21“ geschrieben. Diese Zeile wird vom Programm aufgerufen und der Wert wird an AB 0 transferiert, wenn die Fußgänger-ampel „rot“ und die Autoampel „grün“ zeigen soll.

Funktion	A_grün	A_gelb	A_rot	F_grün	xxx	F_rot	DEZ	HEX
Klemme	A 0.5	A 0.4	A 0.3	A 0.2	A 0.1	A 0.0		
Wert	32	16	8	4	2	1		
Merker								
NM 10.0	X					X	33	21
M 10.2		X				X	17	11
M 10.3			X			X	9	9
M 10.4			X	X			12	C
M 10.5			X			X	9	9
M 10.6		X	X			X	25	19

NM 10.0 bedeutet, der Merker ist nicht gesetzt. Wenn der Zyklus nicht läuft, d.h., die Ampelanlage in Normalstellung steht, müssen die entsprechenden Lampen leuchten.

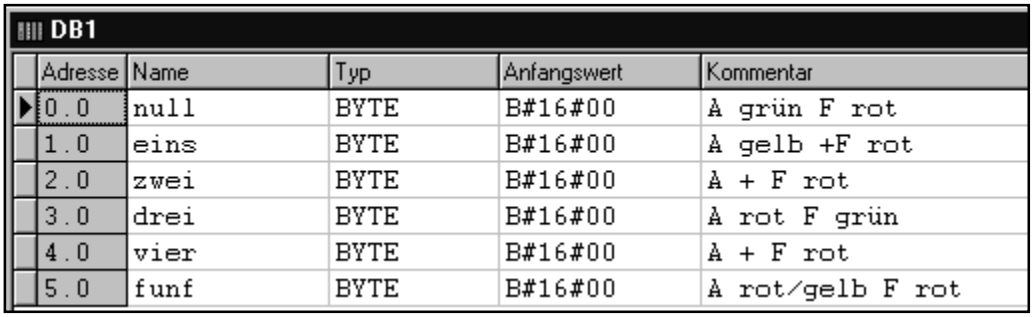
Sie sehen den vollständigen DB 1 – Eintrag in der Datenansicht. In dieser Ansicht können Sie die Aktualwerte eintragen bzw. ändern. Zur Erinnerung: <DB 1> aktivieren, <Ansicht>|<Datenansicht> wählen.



DB1

Adresse	Name	Typ	Anfangswert	Aktualwert	Kommentar
0.0	null	BYTE	B#16#00	B#16#21	A grün F rot
1.0	eins	BYTE	B#16#00	B#16#11	A gelb +F rot
2.0	zwei	BYTE	B#16#00	B#16#09	A + F rot
3.0	drei	BYTE	B#16#00	B#16#0C	A rot F grün
4.0	vier	BYTE	B#16#00	B#16#09	A + F rot
5.0	funf	BYTE	B#16#00	B#16#19	A rot/gelb F rot

Vorher sind in der <Deklarationsansicht> Name, Typ (Byte!) und der Kommentar einzutragen.



DB1

Adresse	Name	Typ	Anfangswert	Kommentar
0.0	null	BYTE	B#16#00	A grün F rot
1.0	eins	BYTE	B#16#00	A gelb +F rot
2.0	zwei	BYTE	B#16#00	A + F rot
3.0	drei	BYTE	B#16#00	A rot F grün
4.0	vier	BYTE	B#16#00	A + F rot
5.0	funf	BYTE	B#16#00	A rot/gelb F rot

Die Anfangswerte müssen Sie nicht eintragen.

OB 1, NW 9: Wahl der Ausgangswörter / Zuordnung der Ausgänge

```
      UN "Zyklus"          //Wenn der Zyklus noch nicht gestartet hat, also in der Grundstellung
      SPBN  stp1           //Wenn doch, springe zur Sprungmarke <stp1:>
      L   DB  1.null       //Sonst lade das Byte <null> im DB 1... (symbolische Adressierung)
      T   AB  0            //... und transformiere es zum AB 0.
      SPA   end            //Danach springe absolut zur Marke <end:>
stp1: U  "MAA_gelb"        //Wenn dieser Merker (Step 1 = Schritt 1) gesetzt ist ...
      SPBN  stp2           //(hier wiederholt sich alles sinngemäß)
      L   DB  1.eins
      T   AB  0
      SPA   end
stp2: U  "MAA_rot"
      SPBN  stp3
      L   DB  1.zwei
      T   AB  0
      SPA   end
stp3: U  "MFAgrun"
      SPBN  stp4
      L   DB  1.drei
      T   AB  0
      SPA   end
stp4: U  "MFA_rot"
      SPBN  stp5
      L   DB  1.vier
      T   AB  0
      SPA   end
```

```
stp5: U  "MAArtglb"
      SPBN  end
      L   DB  1.funf
      T   AB  0
end:  NOP 0                    //Das ist lediglich das Sprungziel, um andere Programmdetails zu
                               //überspringen.
```

Hinweis: Im DB 1 wurde als Typ „Byte“ gewählt. Deshalb entspricht jede Zeile auch einem Byte. Wenn Sie eine Zeile ohne symbolische Adressierung aufrufen wollen, lautet der Befehl z.B. „L DB1.DBBx“, wobei x für das jeweilige Byte steht. Sie können das ganz einfach verfolgen, indem Sie bei aktivem OB1-Fenster unter <Ansicht> die <symbolische Adressierung> setzen bzw. entfernen.

Vgl. TrySim-Projekt <AMPEL_TAB_> im Verzeichnis <Lösungen>.

Sie können jetzt:

- Ein Projekt als Ablaufsteuerung aufbauen.
- Eine Wiedereinschaltsperre programmieren.
- Einen DB deklarieren
- DB-Werte an ein AB transferieren
- Für bestimmte Bitmuster eines Bytes die HEX-Werte bestimmen.
- Die Ampelanlage – nur für Übungszwecke! – so umprogrammieren (im DB!), dass in der Grundstellung alle Lampen gleichzeitig leuchten.

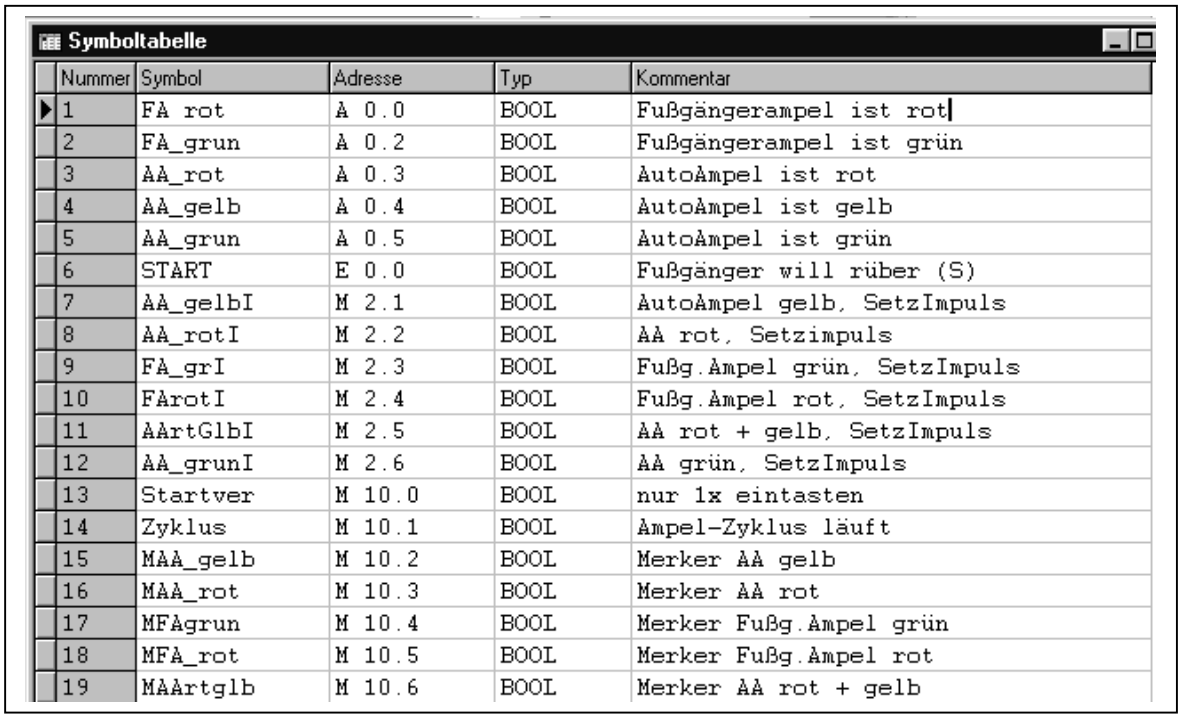
Natürlich kann die Ampelanlage auch als klassische Ablaufsteuerung aufgebaut werden. Testen Sie selbst, welche Version Ihnen besser gefällt, oder welche flexibler ist.
Auf den folgenden Seiten ist die „normale“ Steuerung dargestellt.

Projekt 38: Ampelsteuerung 2

Schwerpunkte: Ablaufsteuerung, zeitgesteuert.

Aufgabe: Das Programm für eine Fußgängerampel ist zu erstellen. Für die zwingende Abfolge der Lampenschaltung bietet sich die Ablaufsteuerung mit S- und R-Funktionen an.

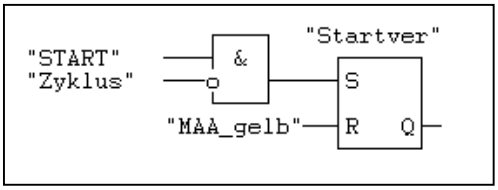
Hierfür kann die gleiche Vorlage des vorherigen Projektes verwendet werden.
Im Verzeichnis <Vorlagen> ist das Projekt unter <Ampel_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.
Die Symboltabelle kann noch etwas angepasst / erweitert werden.



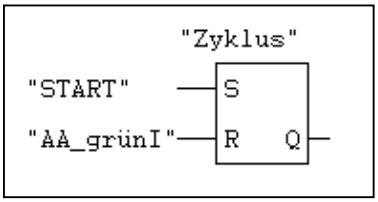
Symboltabelle

Nummer	Symbol	Adresse	Typ	Kommentar
1	FA rot	A 0.0	BOOL	Fußgängerampel ist rot
2	FA_grun	A 0.2	BOOL	Fußgängerampel ist grün
3	AA_rot	A 0.3	BOOL	AutoAmpel ist rot
4	AA_gelb	A 0.4	BOOL	AutoAmpel ist gelb
5	AA_grun	A 0.5	BOOL	AutoAmpel ist grün
6	START	E 0.0	BOOL	Fußgänger will rüber (S)
7	AA_gelbI	M 2.1	BOOL	AutoAmpel gelb, SetzImpuls
8	AA_rotI	M 2.2	BOOL	AA rot, Setzimpuls
9	FA_grI	M 2.3	BOOL	Fußg.Ampel grün, SetzImpuls
10	FArotI	M 2.4	BOOL	Fußg.Ampel rot, SetzImpuls
11	AArtGlbI	M 2.5	BOOL	AA rot + gelb, SetzImpuls
12	AA_grunI	M 2.6	BOOL	AA grün, SetzImpuls
13	Startver	M 10.0	BOOL	nur 1x eintasten
14	Zyklus	M 10.1	BOOL	Ampel-Zyklus läuft
15	MAA_gelb	M 10.2	BOOL	Merker AA gelb
16	MAA_rot	M 10.3	BOOL	Merker AA rot
17	MFAgrun	M 10.4	BOOL	Merker Fußg.Ampel grün
18	MFA_rot	M 10.5	BOOL	Merker Fußg.Ampel rot
19	MAArtglb	M 10.6	BOOL	Merker AA rot + gelb

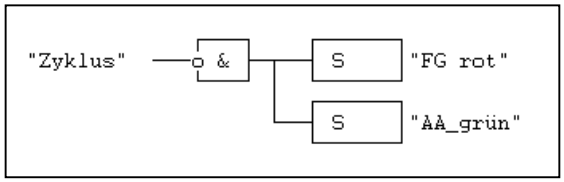
OB1, NW1: Startverriegelung, kein Nachtriggern möglich



OB1, NW2: Zyklus läuft

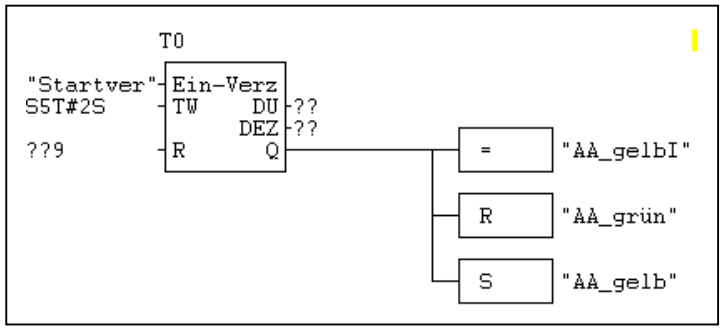


OB1, NW3: Grundstellung



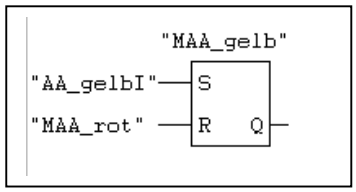
In der Grundstellung leuchtet die rote Fußgängerampel und die grüne Autoampel.

OB1, NW4: AA gelb wird vorbereitet



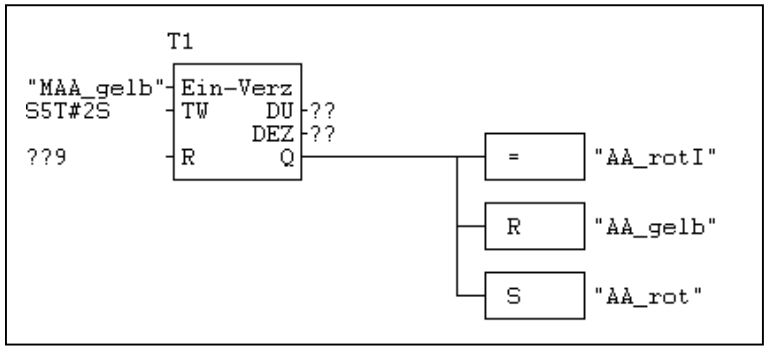
Nach dem Startsignal läuft die Zeitverzögerung, nach der die Autoampel von grün auf gelb umschaltet.
Der Merker „AA_gelbI“ speichert im nächsten Netzwerk die Weiterschaltbedingung. „Startver“ wird von dort zurückgesetzt.

OB1, NW5: Phase AA gelb wird gespeichert



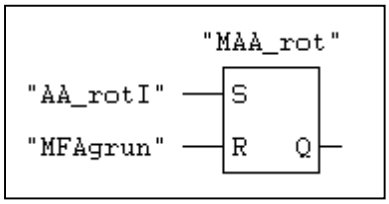
Wenn der Merker „MAA_rot“ gesetzt wird, wird gleichzeitig „MAA_gelb“ zurückgesetzt.

OB1, NW6: AA rot wird vorbereitet



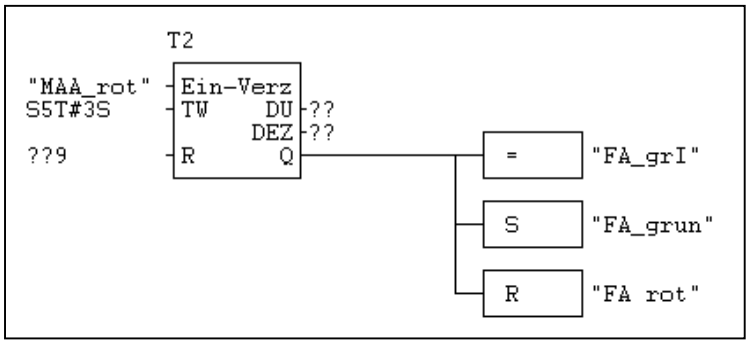
Nach Zeitablauf wird „AA_gelb“ aus- und „AA_rot“ eingeschaltet.
Das Eingangssignal „MAA_gelb“ wird vom nächsten Netzwerk zurückgesetzt.

OB1, NW7: Phase AA rot wird gespeichert



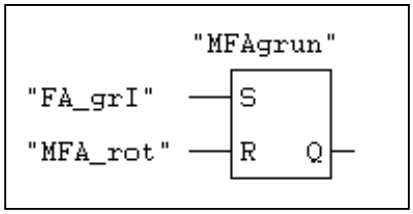
Nach Zeitablauf von T1 wird diese Weiterschaltbedingung gespeichert. Gleichzeitig wird das vorherige Speicherglied „MAA_gelb“ im NM 5 zurückgesetzt.

OB1, NW8: FA grün wird vorbereitet

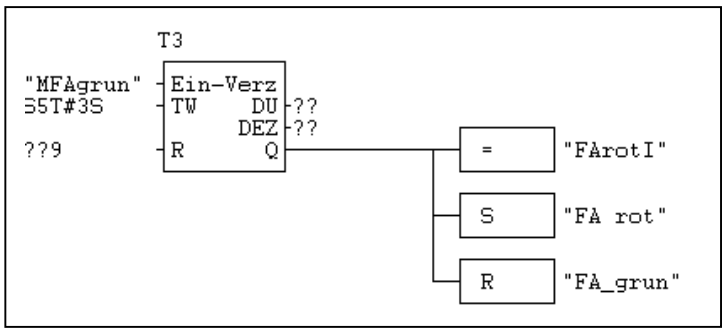


Der Wechsel von Timern und SR-FF wiederholt sich immer wieder.

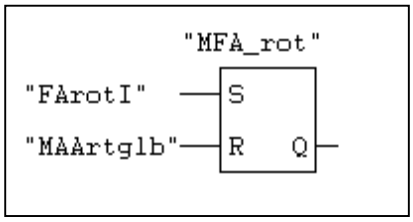
OB1, NW9: Phase FA grün wird gespeichert



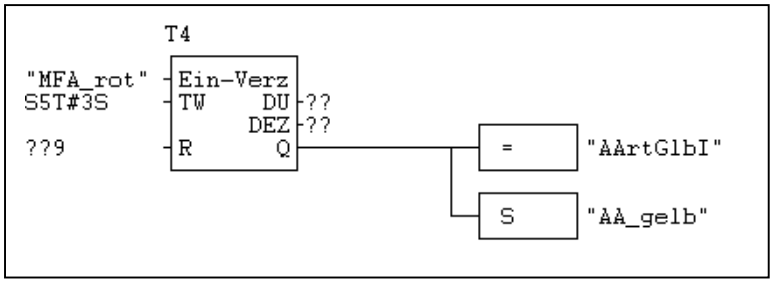
OB1, NW10: FA rot wird vorbereitet



OB1, NW11: Phase FA rot wird gespeichert

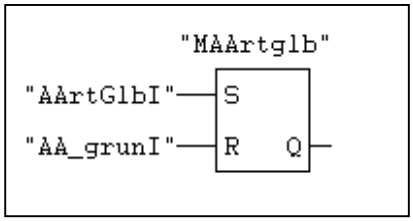


OB1, NW12: AA rot + gelb wird vorbereitet

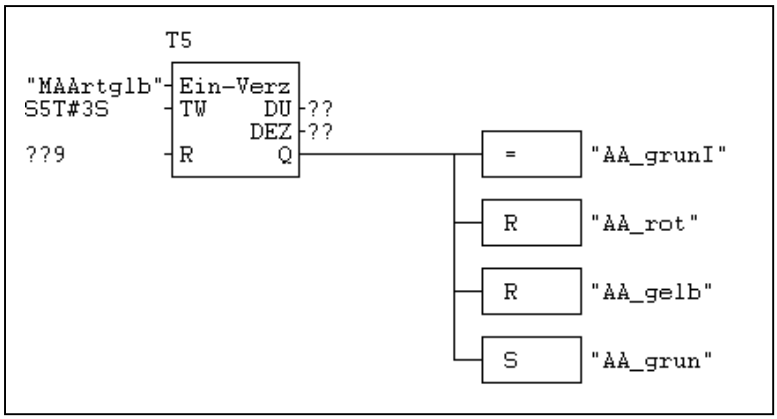


„AA_rot“ wurde im NW 6 gesetzt und ist seitdem aktiv. Dazu kommt jetzt noch „AA_gelb“.

OB1, NW13: Phase AA rot + gelb wird gespeichert



OB1, NW14: AA grün wird vorbereitet



„AA_grunI“ setzt im NW 2 „Zyklus“ zurück.
Außerdem wird dadurch im NW 13 „MAArtglb“ zurückgesetzt, so dass T5 ausschaltet.

Vgl. TrySim-Projekt <Ampel> im Verzeichnis <Lösungen>.

Projekt 39: Sortierstrecke 1

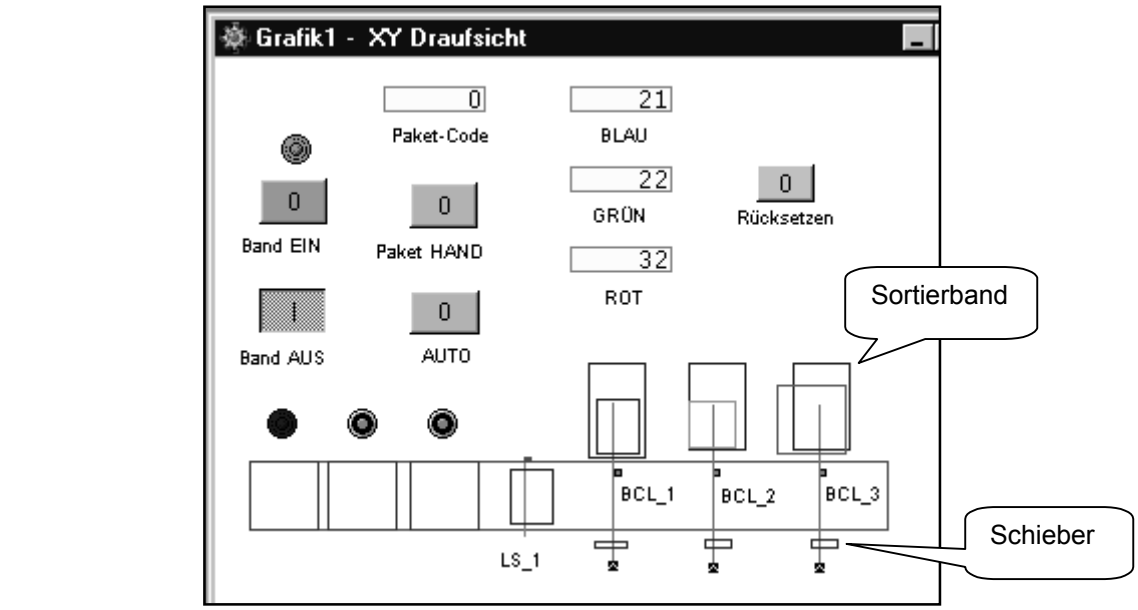
Schwerpunkte: WORD-Verarbeitung, Barcode; SR-FF, ==I.

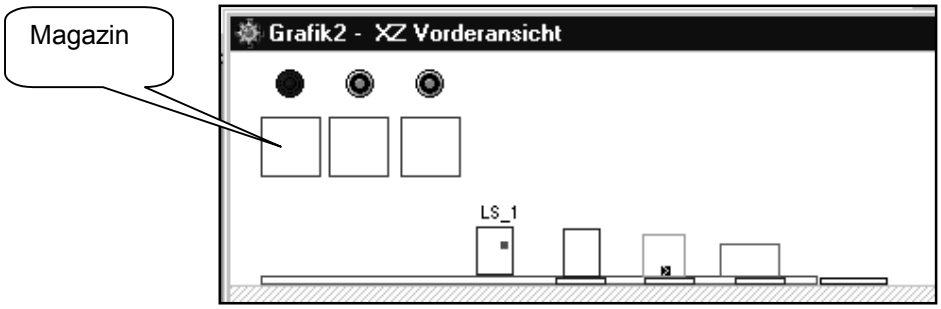
Aufgabe: Sortierstrecke
Auf einem Förderband sollen unterschiedliche Pakete (3Typen) sortiert werden. Die Pakete kommen in unterschiedlicher Reihenfolge an und werden nach dem Zufallsprinzip von TrySim erzeugt. Die Pakete werden 3 verschiedenen Magazinen entnommen.
Die Pakete unterscheiden sich optisch und werden mit einem Barcode gekennzeichnet. Für jeden Pakettyp gibt es eine Lagerstation, die mit einem Barcodeleser ausgestattet ist. Wenn der richtige Code erkannt wird, wird das entsprechende Paket zur Seite geschoben. Das Band muss dabei nicht anhalten.
Zuerst muss das Band eingeschaltet werden. Auch für Automatikbetrieb muss das erste Paket von Hand gestartet werden. Bei Automatikbetrieb gibt die Lichtschranke LS_1 das nächste Paket frei.

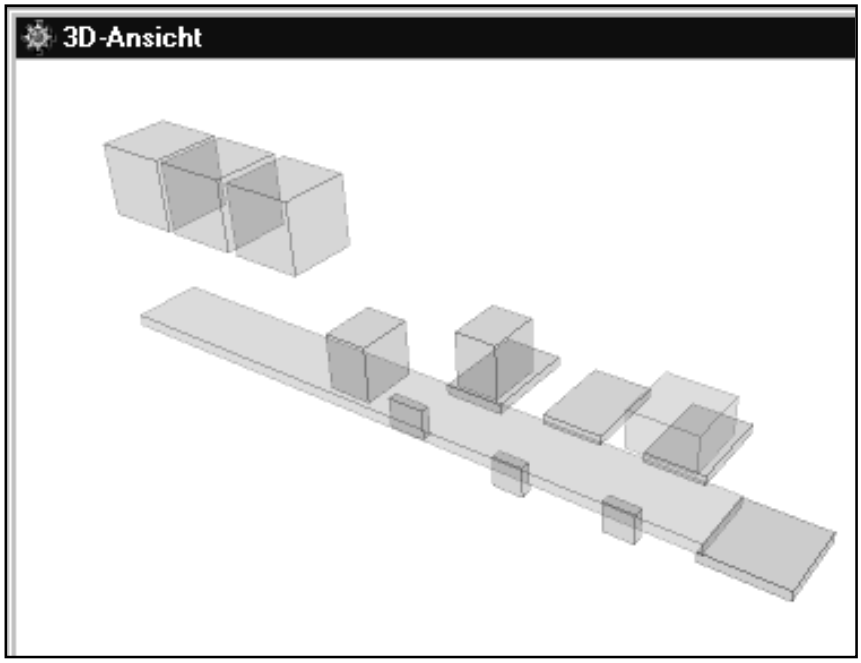
Beim Austasten muss das Band noch leerfahren. Die Anzahl der sortierten Pakete wird angezeigt und kann zentral auf „0“ zurückgesetzt werden.
Anmerkung: Die eigentliche Aufgabe ist nicht das Erzeugen, sondern das Sortieren der Pakete.

Im Verzeichnis <Vorlagen> ist das Projekt unter <SortierStrecke_2_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Symboltabelle und die Anlage sind bereits vorbereitet.



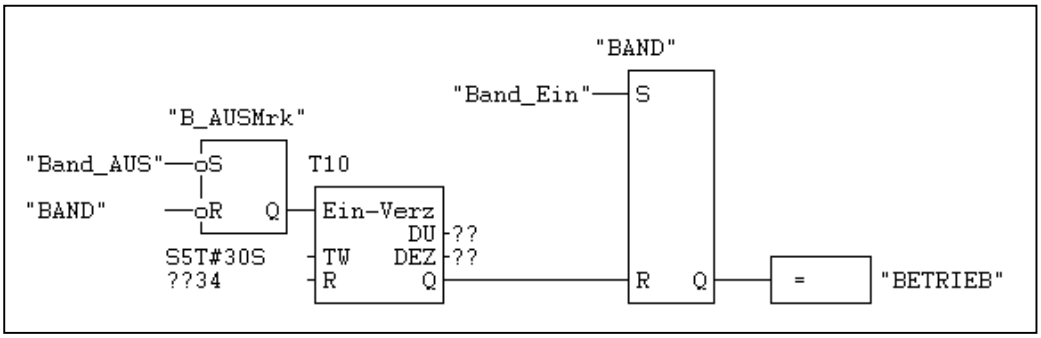




Symbolische Adresse	Operand	Datentyp	Kommentar
Schieb1	A 0.0	BOOL	Ventil Schieber Band blau
BAND	A 0.1	BOOL	Förderband-Antrieb
BETRIEB	A 0.3	BOOL	Betriebsanzeige
M_BLAU	A 1.0	BOOL	Magazin BLAU
M_GRUN	A 1.1	BOOL	Magazin GRÜN
M_ROT	A 1.2	BOOL	Magazin ROT
Schieb2	A 1.5	BOOL	Ventil Schieber Band grün
Schieb3	A 1.7	BOOL	Ventil Schieber Band rot
K_BLAU	AW 500	WORD	Code Blau
K_GRUN	AW 502	WORD	Code Grün
K_ROT	AW 504	WORD	Code Rot
P_CODE	AW 514	WORD	aktueller Paket-Code
Zahl_B	AW 516	WORD	Anzahl blaue Pakete
Zahl_G	AW 518	WORD	Anzahl grüne Pakete
Zahl_R	AW 520	WORD	Anzahl rote Pakete
Band_Ein	E 0.0	BOOL	Ein-Taster (S)
P_HAND	E 0.1	BOOL	Paket von Hand vorlegen
Band_AUS	E 0.2	BOOL	Aus-Taster (Ö)
LS	E 0.3	BOOL	Lichtschranke
AUTO	E 0.4	BOOL	Automatikbetrieb
ZRUCK	E 0.5	BOOL	Paketzähler rücksetzen
L_BLAU	EW 512	WORD	Barcodeleser Blau
L_GRUN	EW 514	WORD	Barcodeleser Grün
L_ROT	EW 516	WORD	Barcodeleser ROT
LEDBLAU	M 1.0	BOOL	Anzeige BLAU
LEDGRUN	M 1.1	BOOL	Anzeige GRÜN
LEDROT	M 1.2	BOOL	Anzeige ROT
IMP_BLAU	M 10.0	BOOL	Impuls Schieber BLAU
IMP_GRUN	M 10.21	BOOL	Impuls Schieber GRÜN
IMP_ROT	M 10.2	BOOL	Impuls Schieber ROT

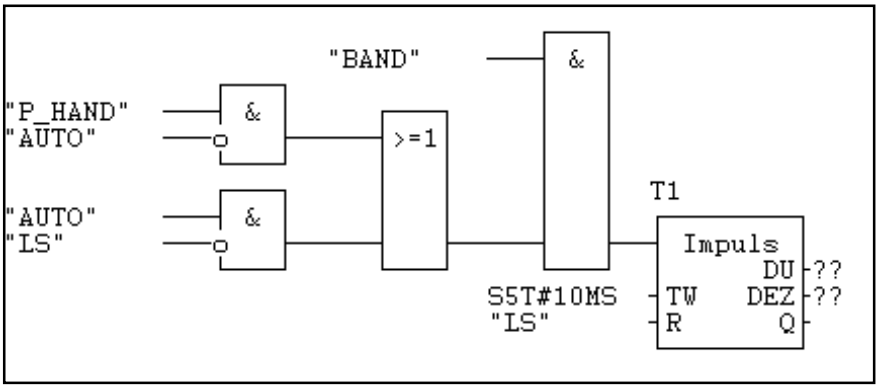
Zuerst ist das Förderband einzuschalten.

OB 1; Netzwerk 1: Förderbandbetrieb



Mit dem Taster <Band EIN> wird das Band eingeschaltet. Nach dem Tasterbefehl <Band AUS> („0"-Signal) liegt am Zeitglied T10 dauerhaft eine „1" an, mit der nach Ablauf einer Verzögerungszeit – wenn das Band leer ist – das Förderband ausgeschaltet wird. Aus der Taster-„1" wird über „B_AUSMrk" eine Dauer-„1". Diese wird zurückgesetzt, wenn das Band ausgeschaltet wurde.

OB 1; Netzwerk 2: Impulsbildung für PaketMagazine



Wenn das Band eingeschaltet ist und ein Paket von Hand angefordert wird, oder bei <AUTO> nach Durchlauf eines Paket die Lichtschranke eine „0" liefert, wird ein kurzer Impuls an T1 erzeugt. In dieser Impulszeit wird im nächsten Netzwerk der Zufallscode ermittelt.

Jetzt kommen wir zum „Geheimnis" der Zufallszahlen von TrySim. Vergleichen Sie hierzu auch unter TrySim <Hilfe> <Index> "FUNC". Diese Funktion erlaubt es, aus einer frei zu wählenden Anzahl einen Wert zufällig auszuwählen.

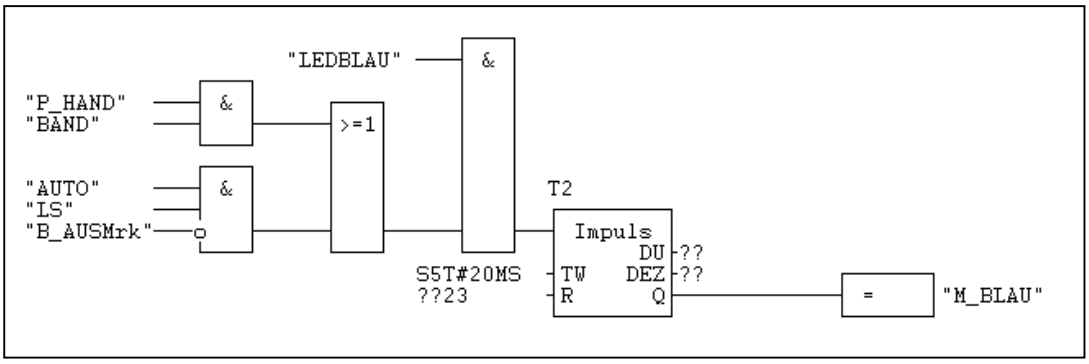
OB 1; Netzwerk 3: Zufallsgenerator für PaketMagazin

```
    UN T   1          //Wenn der Timer T1 nicht einen kurze Impuls bildet
    SPB   weit        //springe zur Sprungmarke <weit:>.
    L   3             //Sonst lade die Zahl 3 (3 Magazine)
    FUNC 100          //Aufruf der Zufallsfunktion
    T   "P_CODE"      //und transferiere den Wert an die Variable ...
                        (<P_CODE> wird also nur in der Impulszeit verändert).
weit: L   "P_CODE"    //Lade den Wert von <P_CODE>
    L   0             //Lade die Zahl 0
    ==I               //und vergleiche die beiden Werte
    =  "LEDBLAU"      //Wenn <P_CODE> also „0" ist, gebe eine „1" auf  <LEDBLAU">
    L   "P_CODE"      //(Wiederhole den Vorgang für den Vergleich mit 1)
    L   1
    ==I
    =  "LEDGRÜN"      //Wenn <P_CODE> also „1" ist, gebe eine „1" auf <LEDGRÜN">
```

```
L   "P_CODE"        //(Wiederhole den Vorgang für den Vergleich mit 2)
L   2
==I
=   "LEDROT"        //Wenn <P_CODE> also „2“ ist, gebe eine „1“ auf <LEDROT“>
```

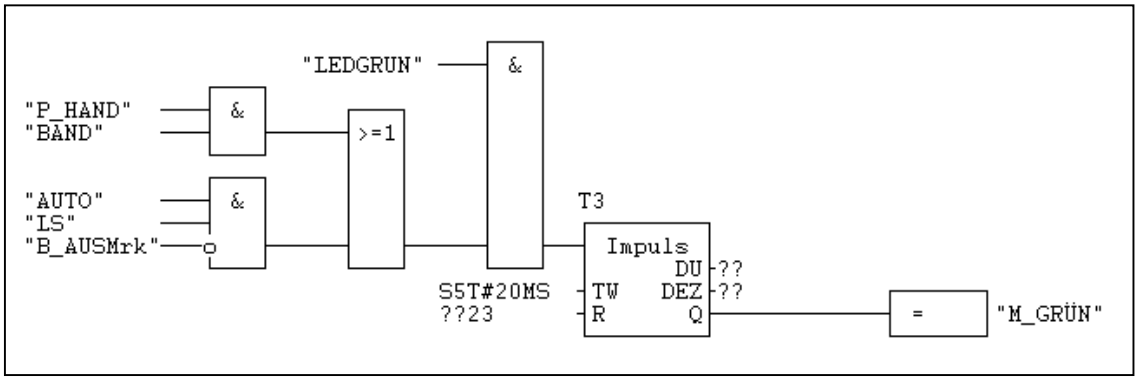
Das Signal steht bis zum nächsten Timer-Impuls dauerhaft an.
Sie können an den LEDs der Magazine schon im voraus erkennen, welchen Magazin geöffnet wird.

OB 1; Netzwerk 4: Paket aus Magazin BLAU



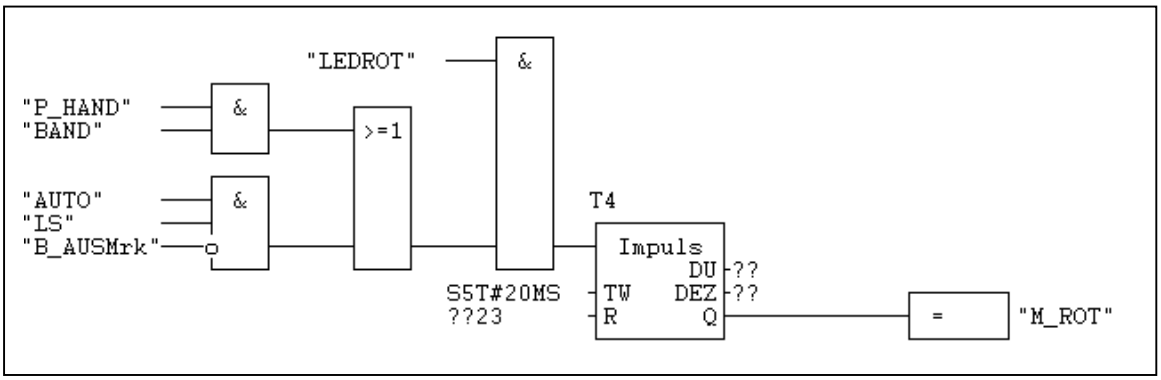
Wenn mit „LEDBLAU“ die Vorwahl für das blaue Magazin auf „1“ gesetzt wurde, das Band läuft und von Hand ein Paket angefordert wird, oder bei Automatikbetrieb die Lichtschranke vom letzten Paket betätigt wird und nicht <Band AUS> betätigt wurde, wird mit einem Impuls ein blaues Paket aus dem Magazin <M_BLAU> entnommen.

OB 1; Netzwerk 5: Paket aus Magazin GRÜN



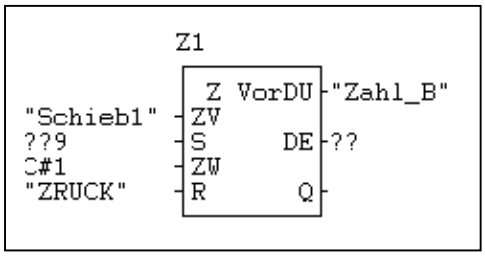
Wenn mit „LEDGRUN“ die Vorwahl für das grüne Magazin auf „1“ gesetzt wurde, das Band läuft und von Hand ein Paket angefordert wird, oder bei Automatikbetrieb die Lichtschranke vom letzten Paket betätigt wird und nicht <Band AUS> betätigt wurde, wird mit einem Impuls ein grünes Paket aus dem Magazin <M_GRUN> entnommen

OB 1; Netzwerk 6: Paket aus Magazin ROT



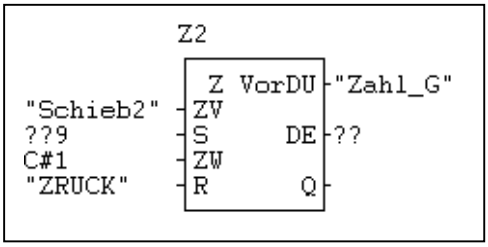
(Sie kennen das Programm jetzt sicherlich schon...)

OB 1; Netzwerk 7: Zählen der blauen Pakete

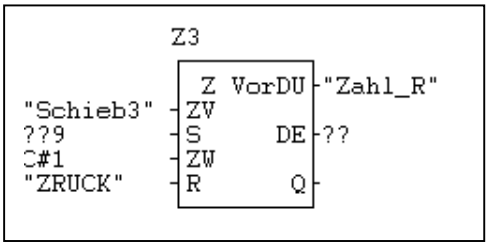


Durch den Impuls des Schiebers wird gleichzeitig ein Zähler für die blauen Pakete hochgezählt.
Mit dem Taster <Rücksetzen> werden alle Zähler auf „0“ zurückgesetzt.
Die Anzahl wird mit der Digitalanzeige als INT angezeigt. (Auf den richtigen Ausgang achten!).

OB 1; Netzwerk 8: Zählen der grünen Pakete



OB 1; Netzwerk 9:

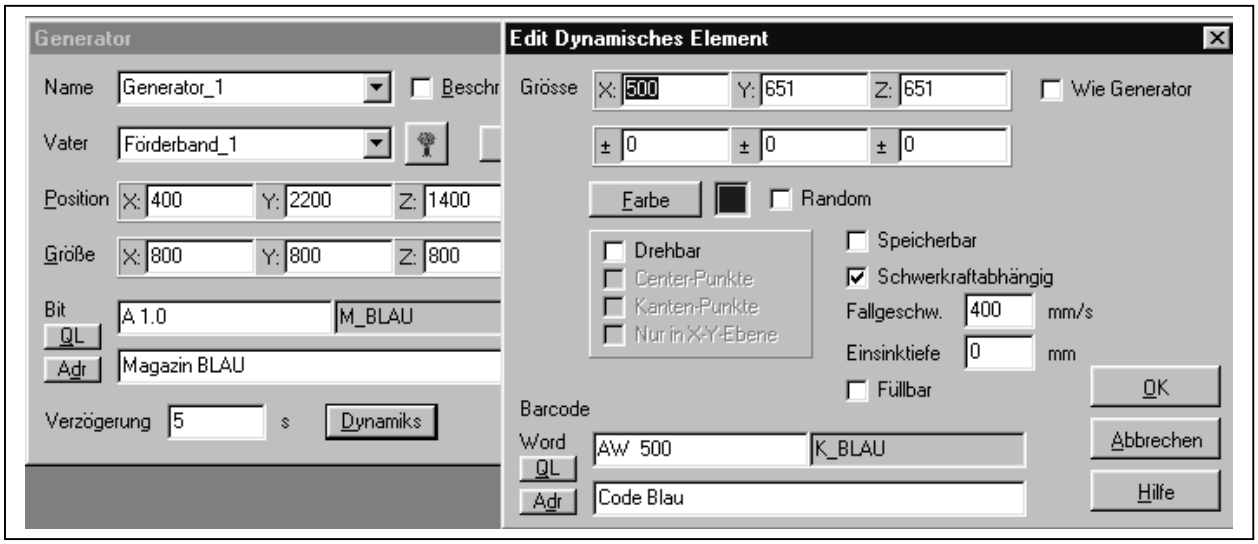


Im nächsten Netzwerk wird gezeigt, wie dem Paket ein Barcode zugeordnet wird.

OB 1; Netzwerk 10: Barcodeleser und Zuordner

```
UN "BAND"              //Wenn das Band nicht eingeschaltet wurde...
   SPB  end            //...springe zur Marke <end:>
   L  3                //Sonst lade die Zahl 3 ...
   T  "K_BLAU"         //...und transferiere sie an das AW 500.
```

Wenn Sie mit <rM> das linke Magazin (nicht die blauen Dynamiks) anklicken, so die Editiermaske öffnen und dann mit <LM> auf <Dynamiks> drücken, erkennen Sie das Wort AW 500, das der Barcode-Kennung zugeordnet ist. Aus der Symboltabelle wurde <K_BLAU> übernommen.



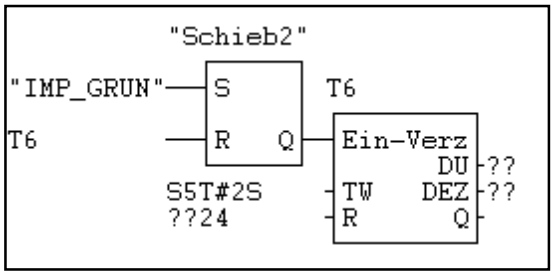
Wenn Sie im ausgeschalteten Zustand den Cursor auf den Dynamiks verweilen lassen, wird die hier vergebene Kennung angezeigt. Durch diesen Transfer wird dem Codewort der Wert zugeordnet. Für die anderen Magazine / Pakete / Dynamiks wird in gleicher Weise verfahren.

```
L  1
T  "K_GRUN"      // „K_GRUN“ bekommt die Kennung 1
L  2
T  "K_ROT"       // „K_ROT“ bekommt die Kennung 2
```

An jedem Barcode-Leser wird der gescannte Code, der wie bei einer Lichtschranke beim Überdecken bzw. Durchlaufen erfasst wird, mit den 3 Paketcodes verglichen. Wenn sie übereinstimmen, wird ein Impuls erzeugt, der den Schieber betätigt.

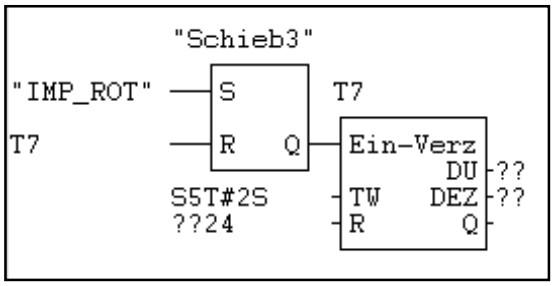
```
L  "L_BLAU"       // Lade das Eingangswort des Barcodelesers „BLAU“.
L  3              // Lade die Zahl 3
==I               // und vergleiche die beiden Werte.
=  „IMP_BLAU“     // Wenn sie identisch sind, betätige den Schieber “BLAU”.
SPB  end          // Danach springe zur Sprungmarke <end:>.
                  // Wenn diese Bedingung nicht erfüllt wurde...
L  "L_GRUN"       // …setze die Auswertung sinngemäß fort.
L  1
==I
=  „IMP_GRUN“
SPB  end

L  "L_ROT"
L  2
==I
=  „IMP_ROT“
```

OB 1; Netzwerk 11: Schieber GRÜN

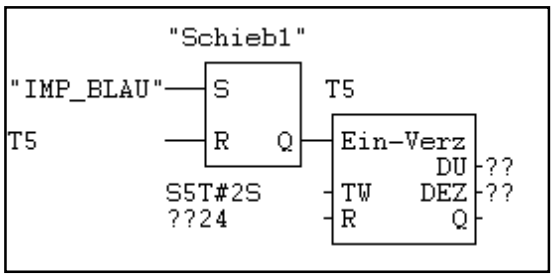


Durch den Impuls vom Vergleicher wird der Linearbeweger (Schieber) betätigt. Nach der eingestellten Zeit wird durch eine Art „Selbstmordschaltung“ der Ausgang auf „0“ gesetzt. Das 4/2-Wgeventil setzt den Antrieb (vgl. im Editierfenster des Antriebs mit <rM>) zurück.

OB 1; Netzwerk 12: Schieber ROT



OB 1; Netzwerk 13: Schieber BLAU



Vgl. TrySim-Projekt <SortierStrecke_2> im Verzeichnis <Lösungen>.

Projekt 40: Byteverarbeitung, Maskierung

Schwerpunkte: Abfrage vom Eingangsbyte, Maskierung, Ausgangsbyte.

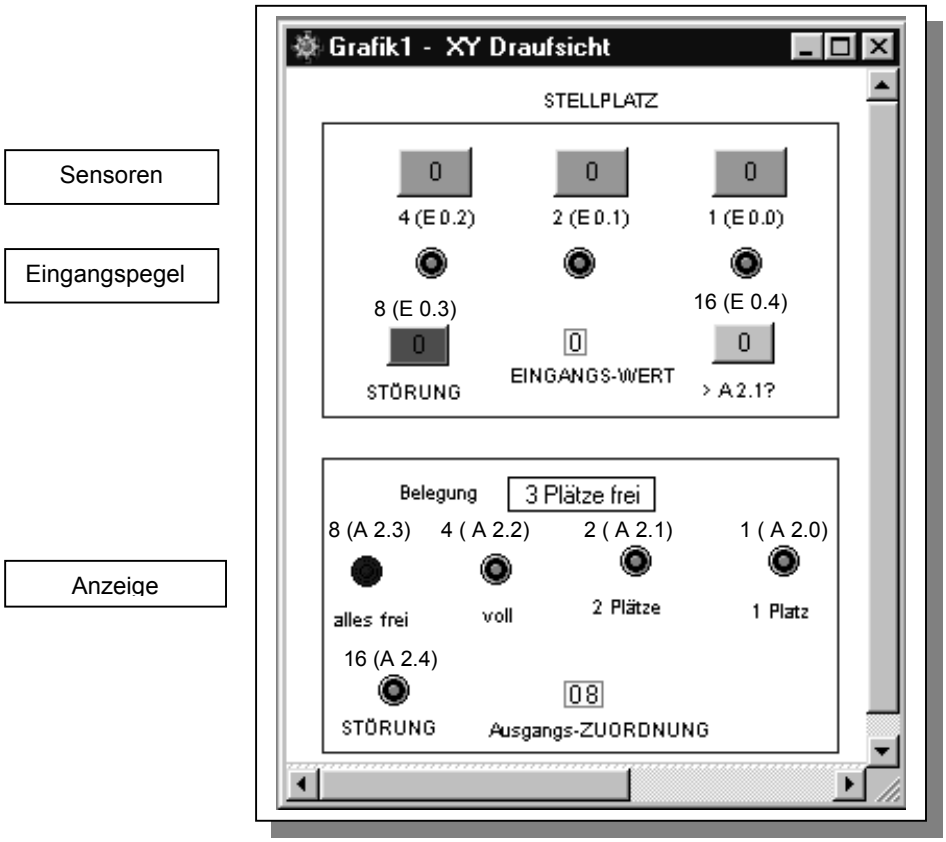
Aufgabe: In einer Miniparkgarage soll der Belegungszustand angezeigt werden. Wenn irgendein Platz belegt ist, soll die grüne Lampe leuchten (duale Wertigkeit 1), bei 2 belegten Plätzen die gelbe Lampe (Wert 2) und wenn alle 3 Plätze belegt sind, die rote Lampe (Wert 4). Sind alle Plätze frei, leuchtet die blaue Lampe (Wert 8).

Theoretische Erörterung:
Die Eingänge werden nicht einzeln erfasst und ausgewertet, sondern gemeinsam als Byte, d.h. 8 Eingänge (oder Ausgänge) bilden gemeinsam durch das Bitmuster eine einzige Information.

In diesem Beispiel wird eine logische Verknüpfung nicht durch digitale Bausteine sondern durch unterschiedliche Wertigkeit des Ausgangsbytes erreicht. Das Ergebnis wird als Dualzahl dargestellt und auf die entsprechenden Ausgänge (eines einzigen Ausgangsbytes) geleitet.

In der ersten Phase geht es um die minimale Lösung der Aufgabenstellung; später werden wir einige wichtige Randprobleme betrachten.

Zur Praxis mit TrySim:
Öffnen Sie bitte im Verzeichnis <Vorlagen> das Projekt <EB_AB> und speichern es in Ihrem Arbeitsverzeichnis unter einem sinnvollen Namen.
Das Projekt ist als Anlage und als Symboltabelle bereits vorbereitet.



Die Zahlen vor den Klemmen (Operanden) geben die Wertigkeit an, die diese Klemmen als Dualzahlabfrage haben.

Symbolische Adresse	Operand	Datentyp	Kommentar
LPlatz_1	A 0.0	BOOL	Platz 1 ist belegt
LPlatz_2	A 0.1	BOOL	Platz 2 ist belegt
LPlatz_3	A 0.2	BOOL	Platz 3 ist belegt
1_belegt	A 2.0	BOOL	1 Platz ist belegt
2_belegt	A 2.1	BOOL	2 Plätze sind belegt
voll	A 2.2	BOOL	Garage ist voll belegt
leer	A 2.3	BOOL	alle Plätze sind frei
LStörung	A 2.4	BOOL	Störung / Gas-Alarm
WERT_AB2	AB 256	BYTE	Zahlenwert AB 2
WERT_EB0	AW 512	WORD	Zahlenwert EW0
Platz_1	E 0.0	BOOL	Sensor Platz 1
Platz_2	E 0.1	BOOL	Sensor Platz 2
Platz_3	E 0.2	BOOL	Sensor Platz 3
Störung	E 0.3	BOOL	Sensor Störung / Gas-Alarm
Fehler	E 0.4	BOOL	Falschverdrahtung an A 2.1

Wie eingangs erwähnt, wird hier eine ganz andere Art der Programmierung vorgestellt. Die Spannungspegel an den Eingangsklemmen (Operanden) E 0.0 bis E 0.2 dienen als Abfrage, welche Parkboxen belegt sind. Bei 3 Plätzen (Eingängen) können $2^3 = 8$ verschiedene Belegungs-kombinationen vorkommen. Man kann jetzt entsprechend der Dualzahl-Logik den Schaltern eine unterschiedliche Wertigkeit zuordnen - von "ganz leer" = 0 bis "voll" = 7.
Da nicht alle Befehle durch FUP- oder KOP-Boxen darstellbar sind, werden sie hier in AWL programmiert. Mit dem umwerfend einfachen Befehl "L EB 0" wird die gesamte Information des **E**ingangs**B**ytes **0** abgefragt (geladen). Der Befehl (die Operation) beginnt also nicht mit "U"!

Netzwerk 1: Lesen des Eingangsbytes EB 0.
Die Einzelinformationen der Eingangsklemmen E0.0 bis (theoretisch) E0.7 werden gemeinsam als Eingangsbyte EB 0 abgefragt.

```
L  EB  0          //Lade das Eingangsbyte 0
T  "WERT_EB0"     //Transferiere den Wert an das Ausgangswort AW 512.
                  //hier: Anzeige
```

Der Zahlenwert des Eingangsbytes wird an der Digitalanzeige "Eingangs-Wert" dargestellt. Dazu wurde der Digitalanzeige in deren Editierfenster (im ausgeschalteten Zustand mit <rM> auf das Element klicken) das Ausgangswort "AW 512" = "Wert_EB0" zugewiesen.
Dieses Netzwerk ist jetzt schon zu testen. (Den Schalter "Störung" benutzen wir später).

Im nächsten Netzwerk werden einzelne Bits aus dem Byte „herausgefiltert“.

Netzwerk 2: Zuweisung von Eingangs- zu Ausgangspegeln.

```
U  E  0.0      //Lade den Signal-Pegel der Klemme E0.0
=  A  0.0      //Bei einer logischen 1 an E0.0 lasse LED A0.0 leuchten
U  E  0.1      //usw.
=  A  0.1
U  E  0.2      //Mehrere Zuweisungen können nur in AWL in einem Netzwerk erfolgen.
=  A  0.2
```

Sie sehen, die Byte-Abfrage schließt die Bit-Abfrage nicht aus. Dieses Netzwerk dient nur der separaten Anzeige der Eingangspegel. Es könnte auch durch folgende Anweisung ersetzt werden:

```
L EB 0
T AB 0
```

Netzwerk 3: Dies ist das "eigentliche" Programm.

Es sind keine digitalen Verknüpfungen programmiert, sondern es wird in Abhängigkeit vom Eingangswert ein Bitmuster geladen und an Ausgangsbytes transferiert.
Genau wie ein Wert eines Eingangsbytes geladen werden kann, kann auch eine Zahl (Konstante) vom Programm vorgegeben und geladen werden. Dazu ist es erforderlich, die Zahlenbasis anzugeben. Bei einer Dualzahl geschieht das durch die vorgestellte Kombination "2#".
Bei den 2#xxxxx - Konstanten entspricht jede Stelle einem Bit.
In der AWL-Darstellung werden führende Nullen unterdrückt, daher ist jedes Bitmuster der Klarheit wegen im Kommentar nochmals aufgeführt.

```
    L   EB  0          //Lade das Eingangsbyte 0, d.h. den Wert aller „Klemmen“ von E0.0 bis E0.7
    L   0              // Lade den Wert 0
    ==I                // und vergleiche die Werte.
    SPBN  __1          // Wenn sie nicht gleich sind, springe zur Marke <__1:>
    L   2#1000         // sonst (wenn Eingangsbyte 0 den Wert 0 hat), lade 2#1000 = 8
```

Wenn also kein Platz belegt ist, soll das als Ergebnis durch das Bit A 2.3 (LED „alles frei“) erkennbar sein. A 2.3 gehört zum Ausgangsbyte AB 2. Wenn im AB 2 der Wert „8“ dargestellt wird, leuchtet das Bit A 2.3. Deshalb wird in vorstehender Zeile die „8“ geladen.

```
    SPA   tran         // Springe absolut (auf jeden Fall) zum Transferieren
```

In diesem Programm wird Schritt für Schritt verglichen, ob der Wert des Eingangsbytes mit einer vorgegebenen Konstanten übereinstimmt. Wenn das der Fall ist, wird ein neuer (!) Wert in den Akku geladen und das Programm springt zur Sprungadresse <tran:>. Dieser Ausdruck ist frei gewählt. Er darf max. 4 Zeichen lang sein und steht hier als "Eselsbrücke" für "transferieren". Dort geht es weiter mit dem Befehl "T MB 2". Das bedeutet, der Wert im Akku wird in das **M**erker**B**yte **2** übertragen.
Wenn der Wert im EB 0 nicht den Wert "0" enthält, also wenn die Bedingung nicht erfüllt ist, springt das Programm zur Marke <__1:>". Daher der Befehl "SPBN". Hier ginge es wie folgt weiter:

```
__1:  L   EB  0        // Hier beginnt der gleiche Spaß, nur für den nächsten Vergleichswert.
    L   1              //Die Vergleichszahl hat den Wert "1"
    ==I                //hat EB 0 den Wert 2#0001 = 1?
    SPBN  __2          // ...sonst springe zum nächsten Vergleich.
    L   2#1            // falls ja, lade die Konstante "1" als Dualzahl, um weiter unten
                       // das niedrigste (rechte) Bit auf 2#0001 = 1 zu setzen.
```

Dieser Wert ist nicht zwangsläufig "1", er könnte auch anders gewählt werden, falls es die Aufgabe vorgäbe.

```
    SPA   tran         //Falls diese Bedingung erfüllt wurde, springe von hier aus zur Marke <tran:>.
__2:  L   EB  0
    L   2              //Wenn der mittlere Stellplatz besetzt ist, wird die Klemme E 0.1 auf "1"
                       //gesetzt.
                       //Die Klemme (der Operand) entspricht dem Wert 2^1 = 2.
    ==I                //hat EB 0 den Wert 2#0010 = 2?
    SPBN  __3
    L   2#1            // Dann soll ebenfalls später die rechte LED für eine besetzte Box leuchten.
    SPA   tran
__3:  L   EB  0
    L   3              //Der Wert "3" kann nur durch 2 besetzte Boxen zustande kommen.
    ==I
    SPBN  __4
    L   2#10           // deshalb wird 2#0010 zugewiesen. (Die LED für 2 besetzte Boxen).
    SPA   tran
```

```
__4: L   EB  0
     L   4              //Der Wert "4" kommt nur durch eine besetzte Box zustande.
     ==I
     SPBN  __5
     L   2#1            // deshalb wird 2#0001 zugewiesen. (die LED für 1 besetzte Box).
     SPA   tran
__5: L   EB  0
     L   5
     ==I
     SPBN  __6
     L   2#10           // 2#00 1 0
     SPA   tran
__6: L   EB  0
     L   6
     ==I
     SPBN  __7
     L   2#10           // 2#00 1 0
     SPA   tran
__7: L   EB  0
     L   7
     ==I
     SPBN  _8
     L   2#100          // 2#0 1 00; (die LED für 3 besetzte Boxen).
     SPA   tran
__8:  L 2#1 0000              //Wenn keine vorstehende Zahl stimmt, liegt ein Fehler vor.
```

Dieser Fehler kann erzeugt werden durch Druck auf "Störung" (E0.3). Egal, wieviel Boxen belegt sind, der Schalter allein erhöht den Wert von EB 0 schon um 16, wird also unter allen Umständen als Fehler erkannt. Mit der Anweisung "L 2#10000" wird also die 5. LED (A2.4) aktiviert - allerdings werden alle anderen Anzeigen auf "0" gesetzt!

```
tran: T   AB  2          // Transferiere den geladenen Wert an das Ausgangsbyte 2 (LEDs) und...
      T   AB 256         // ...transferiere den Wert zusätzlich an die Anzeige.
```

Jetzt können Sie das Programm testen.

Unbefriedigend ist sicherlich, dass im Störungsfall die Übersicht über die besetzten Boxen verloren ging. Deshalb soll Abhilfe geschaffen werden. Dazu ist ein kleiner Exkurs zu weiteren Befehlen erforderlich.

Für manche Aufgaben benötigt man - wie hier - nicht alle Klemmen eines Bytes. Die restlichen Klemmen sind aber zu kostbar, um unbenutzt verschwendet zu werden. Deshalb kann man die Bereiche voneinander trennen. Durch das Maskieren kann man bestimmte Bits zur Weiterverarbeitung auswählen. Dazu verknüpft man die gewünschten Bits einfach über ein UND mit einer "1" an der entsprechenden Stelle. Für das ganze **Wort**: gibt es einen erfreulich einfachen Befehl:

"UW" (= UND WORT)

Beispiel:
Es sollen nur die 3 niedrigsten Bits auswerten werden - egal, welche Bits noch gesetzt sind. Das ganze Byte soll zufällig folgende Bits gesetzt haben:

	L EB 4	0 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1 0	
die Maske:	L 2#111	0 0 0 0 0 0 0 0 0 0 0 0 0 **1 1 1**	wird addiert mit dem simplen Befehl...
	UW		und führt zum Ergebnis:
		0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0	

Nur die Bits, die auch in der Maske "1" führen, werden ausgewertet. Dieser Wert liegt dann automatisch im Akku.

```
     L   EB  0          //Lade das Eingangsbyte 0, d.h. alle Klemmen von E0.0 bis E0.7
     L   2#111          //Lade die Maske für die 3 niedrigsten Bits.
```

```
    UW              //"UND WORT"
                    //Die 3 niedrigsten Bits von EB 0 stehen zum Vergleich mit der nächsten Zeile
                    //an.
    L  0            // Lade den Wert 0.
    ==I             // und vergleiche die Werte.
    SPBN __1        // Wenn sie nicht gleich sind, springe zur Marke __1
    L  2#1000       // 2#1000 = 8, wird geladen, wenn Eingangsbyte 0 den Wert 0 hat
    SPA  tran       // Springe zum Transferieren

__1: L  EB 0        // Hier beginnt der gleiche Spaß, nur für den nächsten Vergleichswert.
    L  2#111        // Die Maske bleibt für jeden Vergleich dieselbe.
    UW
    L  1
    ==I
    SPBN __2
    L  2#1          // 2#0001
    SPA  tran
__2: L  EB 0
    L  2#111
    UW
    L  2
    ==I
    SPBN __3
    L  2#1          // 2#0001
    SPA  tran
__3: L  EB 0
    L  2#111
    UW
    L  3
    ==I
    SPBN __4
    L  2#10         // 2#0010
    SPA  tran
__4: L  EB 0
    L  2#111
    UW
    L  4
    ==I
    SPBN __5
    L  2#1          // 2#0001
    SPA  tran
__5: L  EB 0
    L  2#111
    UW
    L  5
    ==I
    SPBN __6
    L  2#10         // 2#0010
    SPA  tran
__6: L  EB 0
    L  2#111
    UW
    L  6
    ==I
    SPBN __7
    L  2#10         // 2#0010
    SPA  tran
__7: L  EB 0
    L  2#111
    UW
    L  7
    ==I
```

```
    SPBN  _8
    L   2#100           // 2#0100
    SPA   tran
_8:  L 2#0              //Wenn keine vorstehende Zahl stimmt, liegt ein Fehler vor

tran: T   MB  2         // Transferiere den geladenen Wert an das Merkerbyte 2
                        //(neu! Siehe Netzwerk 5)
```

Netzwerk 4: Störungsanzeige

```
        U  E   0.3      // Wenn der "Störungsschalter" betätigt wird,
        =  A   2.4      // ... soll die LED an A 2.4 (Teil des AB 2!) leuchten (Störungsanzeige).
        U  E   0.4      // Falls dieser Schalter betätigt wird, soll das eigentlich gar keine
                        // ... Auswirkungen haben
                        // Er wurde nur eingebaut, um zu zeigen, wie man Fehlschaltungen abfangen
                        // ... kann.
        =  A   2.1      // Aus Versehen wurde das VKE auf  Klemme A 2.1 gelegt, die aber nur für
                        // ... die Auswertung der Boxenbelegung verwendet werden darf.
```

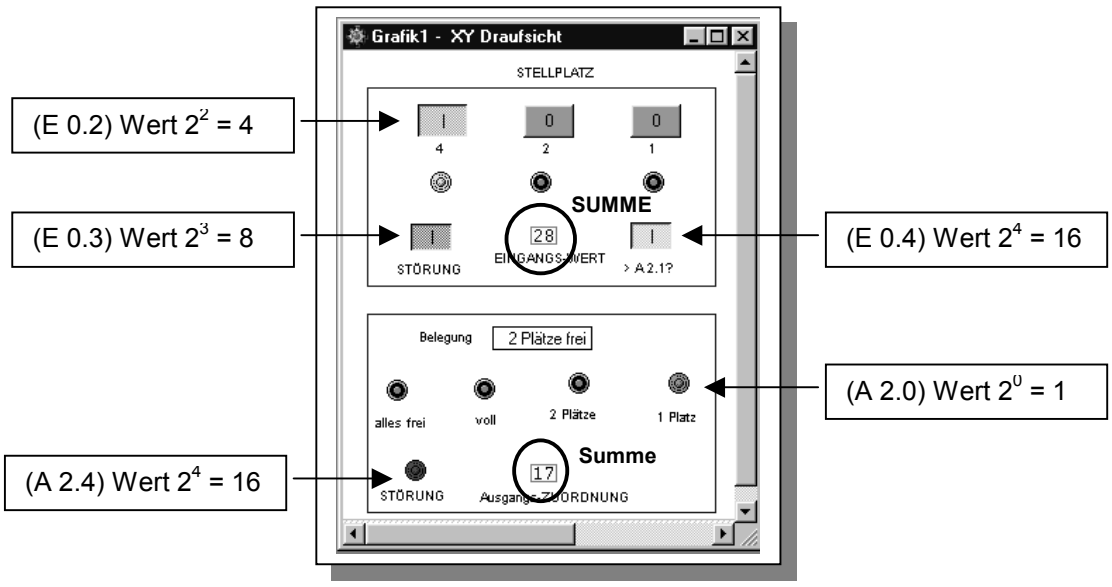
Im nächsten Netzwerk wird das Problem geklärt:

Netzwerk 5: Maskierung des Ausgangsbytes

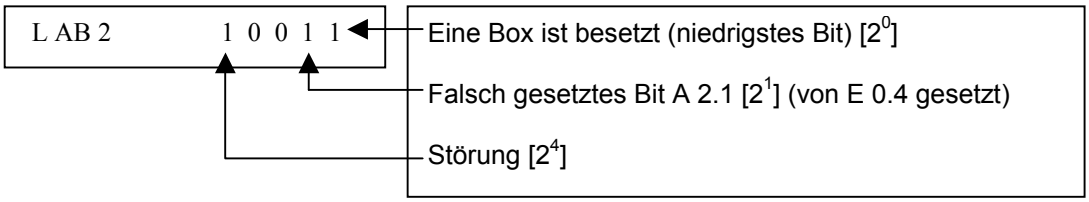
```
        L   AB  2       // Egal, was im AB 2 z.Z. steht, lade es.
        L   2#1111 0000         // Maskiere die oberen 4 Bits
        UW              // UND WORT
        T   AB  2       // Wirkung: Setze die 4 niedrigen Bits von AB 2 auf "0".
                        // Dadurch ist die Fehlschaltung durch E 0.4 an A 2.1 wirkungslos.
        L   MB  2       // Im MB 2 liegt die Information, welche Ausgangs-LED leuchten soll.
                        // Diese Information war auf die niedrigen 4 Bits beschränkt.
        OW              // neu: "ODER WORT", d.h. die oberen Bits aus AB 2 und die unteren aus
MB2
                        // werden neu "gepackt", wobei ihre jeweilige Wertigkeit erhalten bleibt.
        T   AB  2       // Dieses Gesamtergebnis wird wieder an AB 2 ...
        T   AB  256     // ... und über AB 256 an die Digitalanzeige gesandt.
```

Sie können die Zusammenhänge bzw. Auswirkungen sehr schön nachvollziehen, wenn sie das Netzwerk 5 auf "Beobachten" schalten.

Fallstudie: Einige Schalter / Sensoren sind betätigt (führen "1").



Zugehöriges Netzwerk 5, Zeile 1: Angezeigt wird im Akku 1 das Bitmuster am Byte AB 2, in das sich das falsche Bit von A 2.1 eingeschlichen hat (weil E 0.4 betätigt wurde). Dieses Bit und das "Störungs"-Bit wurden im Netzwerk 4 programmiert.

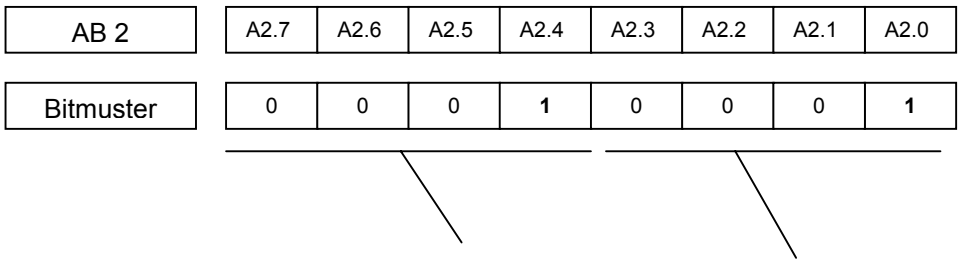


noch einmal diese Zeile:

L AB 2	1 0011	
L 2#1111 0000	1111 0000	Die Maske lässt nur die oberen Bits zu.
UW	1 0000	Nur Übernahme der Störung als A 2.4

In dieser letzten Zeile wurde die Darstellung des Wertes programmintern geändert. Darauf hat man keinen Einfluss. Man muss es nur wissen, damit man die Anzeige richtig deuten kann.
Im Akku kann ein Doppelwort abgelegt werden, d.h. er umfasst immer 2 x 16 Bits. Diese sind hier BCD-codiert, d.h. jeweils 4 Bits werden durch eine Stelle mit der Codierung von "0" bis "F" präsentiert. Deshalb sind in folgender Zeile die 4 Nullen des niedrigsten Bytes rechts auch als eine "0" dargestellt. Links daneben gefolgt von der "1" des nächsten Viererblocks (der vollständig so aussehen würde: 0001; die oberen Nullen werden nicht dargestellt).

T AB 2	0000 0010	Dieser Wert ist identisch mit der vorherigen Zeile, nur BCD-codiert.
L MB 2	0000 0001	Im MB 2 liegt die Information, dass nur eine LED leuchten soll.
OW	0000 0011	"OW" übernimmt alle "1"-Werte aus beiden Zeilen.
T AB 2	0000 0011	**Das sind nicht die Klemmen, sondern das ist der BCD-codierte Wert!!**



"11" bedeutet, dass der Wert der oberen vier Bits und auch der niederen vier Bits jeweils "1" beträgt. Dazu gehören im Byte AB 2: die Klemmen A 2.4 ("Störung) und A 2.0 (1 Platz belegt). Nur diese LEDs leuchten.

Ergebnis:
Die Anzeige der Boxenbelegung bleibt unmanipulierbar erhalten.
Die nicht für die Belegungserfassung benötigten Klemmen können eingangs- und ausgangsseitig weiter verwendet werden.

---

vgl. TrySim-Projekt <EB_AB> im Verzeichnis <Lösungen>.

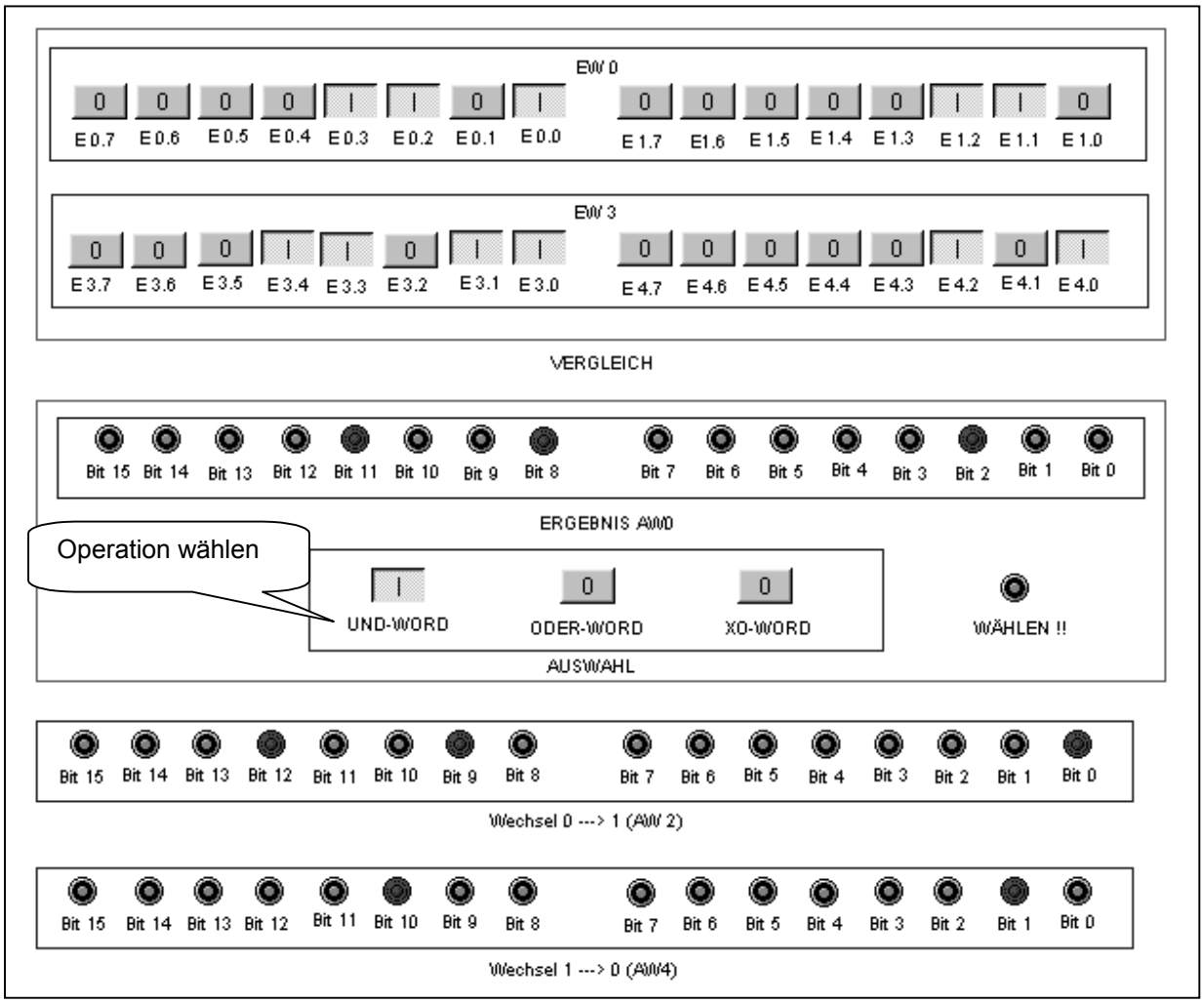
Projekt 41: UW_OW_XOW

Schwerpunkte: Maskieren von Binärstellen; Bitmuster-Ergänzung; Signalwechsel von Binärstellen erkennen.

Aufgaben:

1) Aus einem WORD-Muster sollen bestimmte Bits herausgefiltert werden, bzw. nur ein Teil davon soll verarbeitet werden.
2) Ein beliebiges WORD-Muster soll an bestimmten Bit-Stellen ergänzt werden.
3) Es ist zu überprüfen, ob sich beim Vergleich zweier Wörter einzelne Bits ändern. Dabei ist gleichzeitig festzustellen, ob die Änderung durch "0" / "1" oder einen "1" / "0"-Übergang entstand.

Hinweis: Dieses Projekt soll nur die Auswirkungen der Operationen „UW“, „OW“ und „XOW“ anschaulich darstellen.



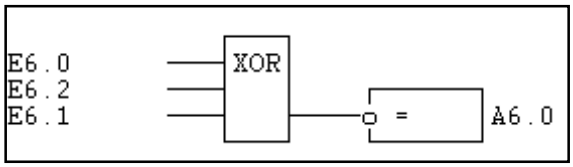
Symboltabelle

	Nummer	Symbol	Adresse	Typ	Kommentar
▶	1	ALT	EW 0	WORD	ursprüngliches Wort
	2	NEU	EW 3	WORD	neues Wort

Das Projekt ist vorbereitet unter <Vorlagen>|<UW_OW_XOW_V>. Sie können es nach dem Öffnen in Ihr Arbeitsverzeichnis kopieren und dort bearbeiten.

In der "Anlage" sind 3 Schalter als Auswahl vorbereitet. Diese sind als Stufenschalter konzipiert. (vgl. TrySim <Hilfe> <Index> "Stufenschalter").
Nach dem Laden des Projektes sind alle Schalter auf "0" gesetzt. Zu Beginn ist ein beliebiger Schalter zu betätigen. Danach ist immer nur der Schalter "1", der gerade betätigt wurde. Die anderen führen "0". Dadurch kann zwischen verschiedenen Programmen mit einem Klick gewechselt werden. Zur Erinnerung an das erste notwendige Einschalten leuchtet zu Beginn - wenn die Anlage eingeschaltet wird - eine Blink-LED. Dieses Programm steht im OB 2. (vgl. TrySim <Hilfe> <Index> "OB").

OB 2; Netzwerk 1: Verknüpfung wählen



Nur wenn 1 Eingang „1“ führt, ist A 6.0 auf „0“ gesetzt. Wenn also noch keine Operation vorgewählt wurde, blinkt die LED.

Zu Aufgabe 1:
"Aus einem WORD-Muster sollen bestimmte Bits herausgefiltert werden, bzw. nur ein Teil davon soll verarbeitet werden".
Vergleichen Sie hierzu auch das Projekt <EB_AB_2>. Es gibt einige Befehle, die es erlauben, Bitmuster von Wörtern zu vergleichen. Dadurch ist es möglich, nur die gewünschten Bits eines Wortes zu verwenden.
Zum Ausblenden von nicht benötigten Binärstellen bildet man eine "Maske", bei der die gewünschten Binärstellen auf "1" und die anderen, auszublendenden Bits auf "0" gesetzt werden.
Diesen Vorgang nennt man "Maskieren". In diesem Beispiel dient das EW 0 als Datenwort und EW 3 als Maske. Die Maske wird als bitweise mit der Wortvorlage UND-verknüpft.

OB 1; Netzwerk 1: Maskieren von Binärstellen

```
     U  E   6.0          // Wenn der Schalter <UND-WORD> gedrückt ist...
     SPB   und           //... springe zur Sprungmarke <und:>.
     U  E   6.1          // Wenn der Schalter <ODER-WORD> gedrückt ist...
     SPB   oder          //... springe zur Sprungmarke <oder:>.
     U  E   6.2          // Wenn der Schalter <XO-WORD> gedrückt ist...
     SPB   xow           //... springe zur Sprungmarke <xow:>.
     SPA   end           // Wenn bisher nichts betätigt wurde, springe zu <end:>.
und: L   "ALT"           // Lade das "alte" Wort EW 0
     L   "NEU"           // Lade das zweite Wort EW 3 (die Maske)
     UW                  // Jedes Bit wird UND-verknüpft
     SPA   end           // springe ohne weitere Bedingungen zur Marke "end"
oder: L   "ALT"
     L   "NEU"
     OW                  // Jedes Bit wird ODER-verknüpft
```

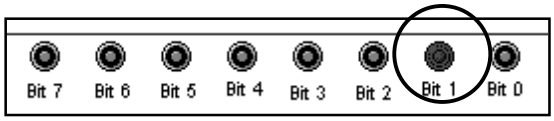
```
    SPA   end
xow:  L   "ALT"
    L   "NEU"
    XOW                 // Jedes Bit wird XOR-verknüpft ("1", wenn ungleich)
end:  T   AW  0
```

Es folgt ein Beispiel für die UND-Verknüpfung

Eingaben:

Wortvorlage: Wert = "E" = 14 dez

Maske: Wert = 3



Ergebnis: Nur Bit 1 ist jeweils "1" = 2

Sie können bei betätigtem <Auge>-Icon die jeweiligen Werte als Dualzahlen (HEX) ablesen:

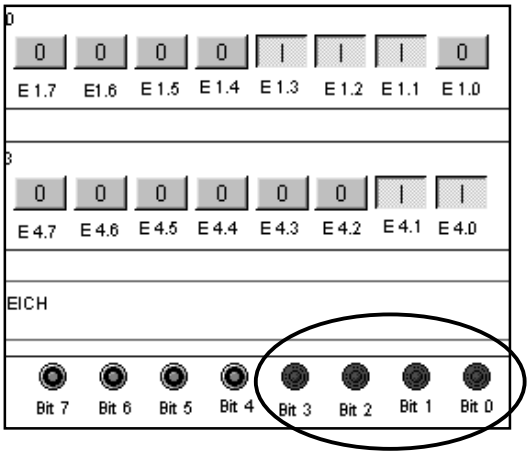
Ausschnitt aus dem aktivem Editierfenster:

```
und:  L   "ALT"                 1 1       00000E00
      L   "NEU"                 1 1       00000300
      UW                        1 1       00000200
      SPA   end
```

Zu Aufgabe 2:
"Ein beliebiges WORD-Muster soll an bestimmten Bit-Stellen ergänzt werden".
Hierzu ist nur der Schalter <ODER-WORD> zu drücken, so dass im Netzwerk 1 zu "oder" gesprungen wird.

```
oder: L   "ALT"                 1 1       00000E00
      L   "NEU"                 1 1       00000300
      OW                        1 1       00000F00
      SPA   end
```

F (HEX) = 15 (dez) als Ergebnis bedeutet, dass Bit 0 bis Bit 3 auf "1" gesetzt sind.



Zu Aufgabe 3:
"Es ist zu überprüfen, ob sich beim Vergleich zweier Wörter einzelne Bits ändern. Dabei ist gleichzeitig festzustellen, ob die Änderung durch "0" / "1" oder einen "1" / "0"-Übergang entstand".

Hierbei dient die zweite Eingabe (EW 3) nicht als Maske, sondern als zweiter Wert, der mit dem ersten verglichen werden soll. Jeder Wechsel soll angezeigt werden. XOW erfasst alle Unterschiede.

OB 1; Netzwerk 2: Übergang (Wechsel) von 0 auf 1.

```
L   "ALT"    0110    // Datenwort 1 ("ALT")
L   "NEU"    1010    // Datenwort 2 ("NEU")
XOW          1100    // Erfassung aller Unterschiede im AKKU
```

"XOW" erkennt den Wechsel im 3. und 2. Bit.

```
L   "NEU"    1010    // Erneutes Laden von "NEU" zum Vergleich mit AKKU (Ergebnis XOW)
UW           1000    // bitweise UND-Verknüpfung
T   AW  2    1000    // Ergebnistransfer an AW 2
```

Durch "UW" wird geklärt, dass nur im 3. Bit ein Wechsel von "0" auf "1" erfolgte.

OB 1; Netzwerk 3: Übergang (Wechsel) von 1 auf 0

```
L   "ALT"    0110    // Datenwort 1 ("ALT")
L   "NEU"    1010    // Datenwort 2 ("NEU")
XOW          1100    // Erfassung aller Unterschiede im AKKU
```

"XOW" erkennt den Wechsel im 3. und 2. Bit.

```
L   "ALT"    0110    // Erneutes Laden von "ALT" zum Vergleich mit AKKU
UW           0100    // bitweise UND-Verknüpfung
T   AW  4    0100    // Ergebnistransfer an AW 4
```

Durch "UW" wird geklärt, dass nur im 2. Bit ein Wechsel von "1" auf "0" erfolgte.

Vgl. TrySim-Projekt <UW_OW_XOW> im Verzeichnis <Lösungen>.

Projekt 42: Bahnsteuerung 1

Schwerpunkte: DWORD-Verarbeitung, NEGD, >D, Absolutwertbildung

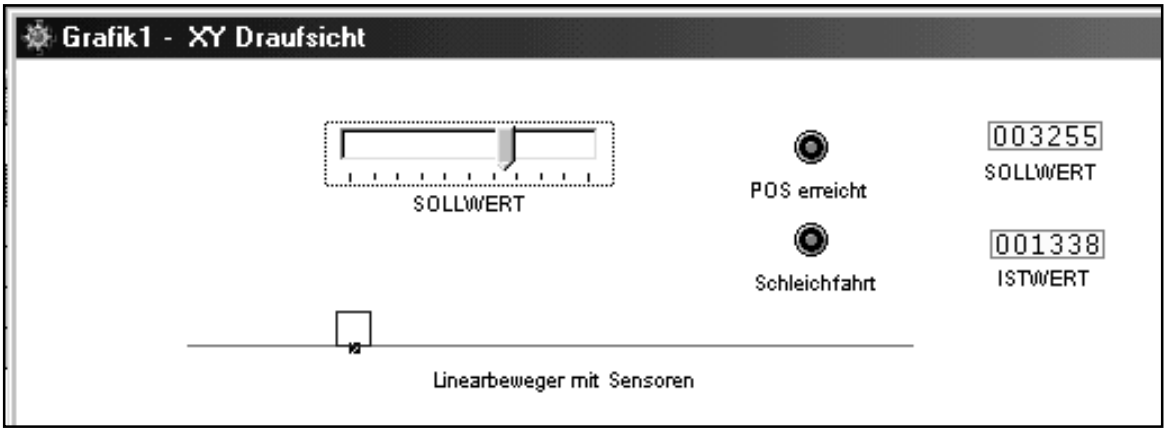
Aufgabe:
Auf einem Förderband soll ein Paket zu jeder beliebigen Position transportiert werden können. Die gewünschte Position soll mit einem linear (digital) arbeitenden Sollwertgeber variabel vorgewählt werden.

Analyse der Aufgabe:
Diese Aufgabe kann natürlich nicht mit einer Unmenge von Endtastern gelöst werden. Hierzu ist ein Soll-/Istwert-Vergleich von hochauflösenden Wegkoordinaten erforderlich. Als Förderbandantrieb kann bei TrySim ein „Linearbeweger mit Sensoren“ gewählt werden, bei dem mit der Antriebswelle ein digitaler Wegaufnehmer gekoppelt ist. Proportional zum Abstand vom Nullpunkt ändert sich der Wert des Doppelwortes (DW...). Die Auflösung ist in der Editiermaske des Linearantriebes einzustellen. Als Sollwerteinsteller dient ein „Regler“, der als Geber oder auch Empfänger von Daten dienen kann (vgl. TrySim <Hilfe> <Index> „Regler“). Beim Bewegen des Schiebers mit <LM> verändern Sie z.B. den Sollwert. Damit sind die Voraussetzungen für einen Soll-/Istwert-Vergleich gegeben.

Durchführung des Projektes mit TrySim:

Im Verzeichnis <Vorlagen> ist das Projekt unter <Bahnsteuerung_1_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Anlage und die Symboltabelle sind bereits vorbereitet.



Symboltabelle

Nummer	Symbol	Adresse	Typ	Kommentar
1	Mrechts	A 0.0	BOOL	Antrieb nach rechts
2	Mlinks	A 0.1	BOOL	Antrieb nach links
3	LEDPOSOK	A 0.2	BOOL	Sollpos. erreicht
4	Schleich	A 0.4	BOOL	Schleichfahrt
5	XPOS	AD 1024	DWORD	Position
6	SOLLWERT	AD 1028	DWORD	SOLLWERT
7	NAHDRAN	M 10.0	BOOL	nahe am Ziel
8	Abstand	MW 20	WORD	Abstand von Sollposition

Das „eigentliche“ Programm zum Soll-/Istwert-Vergleich soll in eine FC geschrieben werden. Diese muss zuerst angelegt und deklariert werden, bevor sie vom DB 1 aufgerufen werden kann.

FC 1: Netzwerk 1:

Deklaration

FC 1 Netzwerk 1

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
▶		in				
	0.0	out	rechts	BOOL		Antrieb nach rechts
	0.1	out	links	BOOL		Antrieb nach links
		in_out				

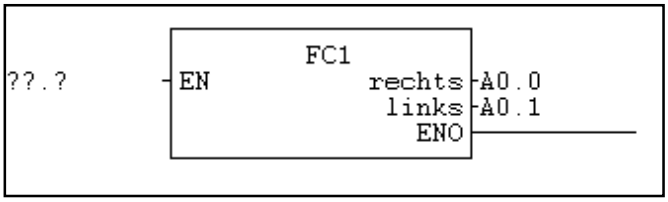
Programm:

In diesem Programm wird lediglich entschieden, ob der Sollwert größer oder kleiner als der Istwert ist. Je nachdem wird die entsprechende Antriebsrichtung eingeschaltet. Bei „Gleichstand“ passiert nichts weiter.

```
    SET             //Befehl, das VKE ohne Bedingungen auf „1“ zu setzten.
                    //Nur durch diese „Vorbereitung“ ist gewährleistet, dass
                    //die folgen „R“-Anweisungen auch sicher ausgeführt werden.
    R  #rechts      //Der Formalpararameter <#rechts> wird ohne Bedingung zurückgesetzt
    R  #links       //Der Formalpararameter <#links> wird ohne Bedingung zurückgesetzt
    L  "SOLLWERT"   //Lade den Sollwert als Doppelwort.
    L  "XPOS"       //Lade den Istwert – die augenblickliche Position des Paketes.
    >D              //prüfe, ob Wert 1 > Wert 2 (Doppelwortvergleich)
    SPB  Xre        //Wenn die Bedingung erfüllt ist, springe zur Marke <Xre:>
    L  "SOLLWERT"   //sonst: Lade den Sollwert als Doppelwort.
    L  "XPOS"       //Lade den Istwert – die augenblickliche Position des Paketes.
    <D              //prüfe, ob Wert 1 < Wert 2
    SPB  Xli        //Wenn die Bedingung erfüllt ist, springe zur Marke <Xli:>
SPA  endx           //sonst springe ohne Bedingung zur Schlussmarke
Xre: SET            //Sprungmarke für den Rechtsantrieb. Zu „SET“ siehe oben.
    R  #links       //Linkslauf wird auf jeden Fall beendet.
    S  #rechts      //Rechtslauf wird über den Formalparameter gesetzt.
    SPA  endx       //Springe absolut zum Bausteinende

Xli: SET            //Sprungmarke für den Linksantrieb
    R  #rechts      //Rechtslauf wird auf jeden Fall beendet.
    S  #links       //Linkslauf wird über den Formalparameter gesetzt.

endx: NOP 0
```

Nun ist das Wahlprogramm der Drehrichtung erst einmal beendet. Der Aufruf erfolgt vom OB 1.

OB 1; Netzwerk 1: Aufruf der Funktion FC 1



Jetzt ist es Zeit für einen Probelauf. Aktivieren Sie dazu das <Auge>-Icon und ziehen den Schieber in eine beliebige Position. Es sieht dann wahrscheinlich so aus, als würde das Paket an der gewünschten Soll-Position liegen bleiben. Verlangsamen Sie aber bitte einmal die Simulationsgeschwindigkeit, indem Sie mehrmals auf die Schildkröte drücken. Sie werden feststellen, dass die Istwert-Anzeige etwas hektisch um den gewünschten Sollwert pendelt, ihn aber nicht sicher erreicht. Wie ist das zu erklären?
In der Simulation und auch in der realen Anlage werden je nach Zyklus- und Rechnerzeit nur zu bestimmten Zeiten – in Intervallen – die Eingangswerte gelesen (und die Ausgangswerte gesetzt). Wenn jetzt der Antrieb auf sein Ziel „zueilt“, ist es sehr unwahrscheinlich, dass genau in dem Augenblick der Messwertaufnahme der Istwert dem Sollwert entspricht. Sollte ein Messwert noch sehr dicht vor der Sollposition liegen, „rast“ der Antrieb dennoch in der alten Richtung weiter. Bei der nächsten Messwerterfassung wurde vielleicht die Sollposition schon überschritten – der Antrieb würde umschalten und mit gleicher Geschwindigkeit zurückrasen. Das Spielchen würde sich wiederholen. Wie ist Abhilfe zu schaffen? Z.B. dadurch, die Geschwindigkeit zu reduzieren. Dadurch würden aber die Prozesse bei langen Bahnen zu langsam werden. Eine mögliche Lösung wäre ein Umschalten der Geschwindigkeiten in der Nähe der Sollposition. Dieser Weg wird jetzt aufgezeigt.

TrySim bietet die Möglichkeit, bei einem wählbaren Operanden die Geschwindigkeit umzuschalten. In der Editormaske des Linearantriebs können Sie unter <Typ> <2-Geschw.-Motor> auswählen und unter <Antrieb> die Geschwindigkeiten und den Umschaltoperanden wählen.
Es ist eine Routine zu entwickeln, mit der die Nähe zum Sollwert erkannt wird und bei dem die Geschwindigkeit des Motors reduziert wird. Dabei ist unerheblich, ob der Antrieb links oder rechts vom Sollwert steht. Der Abstand muss also als Absolutwert (ohne Vorzeichen) ermittelt werden. Vorher wird obiges FC 1-Programm etwas modifiziert, denn die Sollwertposition soll exakt angesteuert werden. Die Änderungen sind fett gedruckt.

```
    SET             //Befehl, das VKE ohne Bedingungen auf „1“ zu setzten.
                    //Nur durch diese „Vorbereitung“ ist gewährleistet, dass
                    //die folgen „R“-Anweisungen auch sicher ausgeführt werden.
    R  #rechts      //Der Formalpararameter <#rechts> wird ohne Bedingung zurückgesetzt
    R  #links
    R  "LEDPOSOK"   //Setze den Ausgang „LEDPOSOK“ auf jeden Fall zurück
    R  "NAHDRAN"    //bei jedem Zyklus sind alle Operanden zuerst auf „0“ gesetzt

    L   "SOLLWERT"  //Lade den Sollwert als Doppelwort.
                    //(Es ist egal, ob es als ED... oder AD...vorliegt)
    L   "XPOS"      //Lade den Istwert – die augenblickliche Position des Paketes.
    >D              //prüfe, ob Wert 1 > Wert 2
    SPB   Xre       //Wenn die Bedingung erfüllt ist, springe zur Marke <Xre:>
    L   "SOLLWERT"  //Wenn vorstehende Bedingung nicht erfüllt ist, lade den Sollwert
    L   "XPOS"      //Lade den Istwert – die augenblickliche Position des Paketes
    <D              //prüfe, ob Wert 1 < Wert 2
    SPB   Xli       //Wenn die Bedingung erfüllt ist, springe zur Marke <Xli:>
    ==D             //prüfe, ob Wert 1 = Wert 2
    S  "LEDPOSOK"   //Setze dann den Ausgang „LEDPOSOK“
    SPA   endx      //dann ist hier praktisch Schluss (es geht auch: „BEA“)

Xre:  SET           //Sprungmarke für den Rechtsantrieb. Zu „SET“ siehe oben.
    R  #links       //Linkslauf wird auf jeden Fall beendet.
    S  #rechts      //Rechtslauf wird über den Formalparameter gesetzt.
    SPA   endx      //Springe absolut zum Bausteinende

Xli:  SET           //Sprungmarke für den Linksantrieb
    R  #rechts      //Rechtslauf wird auf jeden Fall beendet.
    S  #links       //Linkslauf wird über den Formalparameter gesetzt.

endx: NOP 0
```

Die restlichen Änderungen erfolgen im OB 1:

OB 1; Netzwerk 2: Schleichfahrt

```
    L   "SOLLWERT"      //Sollwert
    L   "XPOS"          //Istwert
    -D                  //bilde die Differenz (mit Vorzeichen)
    T   "Abstand"       //transferiere den Wert
    L   0               // Lade die Zahl 0
    >D                  //prüfe, ob „Abstand“ > 0 ist
    SPB   ok            //wenn ja, springe zur Marke <ok:>. (sonst ist „Abstand“ negativ)
    L   "Abstand"       //sonst lade „Abstand“
    NEGD                //und multipliziere den Wert mit –1
    T   "Abstand"       //dadurch ist „Abstand“ jetzt positiv, bzw. eine absolute Zahl
ok: L   "Abstand"       //Sprungmarke. 1) Abfrage des absoluten Abstandes vom Sollwert.
    L   30              //2) Lade Zahl 30 (soll als Abstandswert für die Schleichfahrt dienen)
    -D                  //3) bilde die Differenz: 1) – 2)
    L   30              // Lade die Zahl 30
    <D                  //Wenn die Differenz [3)] kleiner ist als 30
    S  "NAHDRAN"        //dann setze den Merker
    U  "NAHDRAN"        //Wenn der Merker gesetzt wurde...
    UN "LEDPOSOK"       //... und die Zielposition noch nicht erreicht wurde...
    =  "Schleich"       //... soll „Schleichfahrt“ gelten.
```

Vgl. TrySim-Projekt <Bahnsteuerung_1> im Verzeichnis <Lösungen>.

Projekt 43: Bahnsteuerung 2; (Variante von Bahnsteuerung 1)

Schwerpunkte: DWORD-Verarbeitung, NEGD, Absolutwertbildung, Frequenz-Umrichter

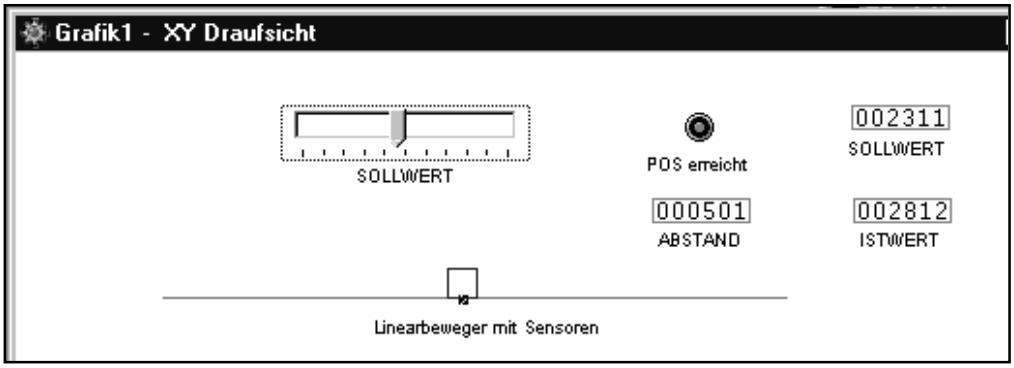
Aufgabe:
Auf einem Förderband soll ein Paket zu jeder beliebigen Position transportiert werden können. Die gewünschte Position soll mit einem linear (digital) arbeitenden Sollwertgeber vorgewählt werden.

Analyse der Aufgabe:
Diese Aufgabe kann natürlich nicht mit einer Unmenge von Endtastern gelöst werden. Hierzu ist ein Soll-/Istwert-Vergleich von hochauflösenden Wegkoordinaten erforderlich. Als Förderbandantrieb kann bei TrySim ein „Linearbeweger mit Sensoren“ gewählt werden, bei dem mit der Antriebswelle ein digitaler Wegaufnehmer gekoppelt ist. Proportional zum Abstand vom Nullpunkt ändert sich der Wert des Doppelwortes (DW...). Die Auflösung ist in der Editiermaske des Linearantriebes einzustellen.
Als Sollwerteinsteller dient ein „Regler“, der als Geber oder auch Empfänger von Daten dienen kann (vgl. TrySim <Hilfe> <Index> „Regler“). Beim Bewegen des Schiebers mit <LM> verändern Sie z.B. den Sollwert. Damit sind die Voraussetzungen für einen Soll-/Istwert-Vergleich gegeben.
Im Unterschied zu <Bahnantrieb_2> soll hier nicht in „Schleichfahrt“ umgeschaltet werden, um die Soll-Position eindeutig anzufahren, sondern die Geschwindigkeit soll abstandsabhängig geändert werden. Dazu bietet sich beim Linearantrieb der WORD-Antrieb oder der Frequenz-Umrichter an.

Durchführung des Projektes mit TrySim:

Im Verzeichnis <Vorlagen> ist das Projekt unter <Bahnsteuerung_2_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Anlage und die Symboltabelle sind bereits vorbereitet.



Symboltabelle

Nummer	Symbol	Adresse	Typ	Kommentar
1	LEDPOSOK	A 0.2	BOOL	Sollpos. erreicht
2	XPOS	AD 1024	DWORD	Position
3	SOLLWERT	AD 1028	DWORD	SOLLWERT
4	Antrieb	AW 512	WORD	WORD-Antrieb
5	ABSTAND	AW 514	WORD	absoluter Abstand
6	DIFF	MD 10	DWORD	+ / - Differenz

Für den Antrieb (Linearbeweger) gibt es nur eine Wortadresse. Dieses Wort beinhaltet schon die Angabe, ob der Motor links oder rechts laufen soll.
Für die Anzeige des Abstandes von der Soll-Position muss aus der Differenz, die vorzeichenbehaftet ist, noch der Absolutwert (ohne Vorzeichen) gebildet werden.

Das „eigentliche“ Programm zum Soll-/Istwert-Vergleich soll in eine FC geschrieben werden. Diese muss zuerst angelegt und deklariert werden, bevor sie vom DB 1 aufgerufen werden kann.

FC 1: Netzwerk 1: Antrieb: Geschwindigkeit und Richtung

Deklaration

FC 1 Netzwerk 1 Antrieb: Geschwindigkeit und Richtung

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
		in				
	0.0	out	MOTOR	WORD		WORD-Antrieb
		in_out				
		temp				

Programm:

```
SET
R  "LEDPOSOK"        //Bedingungslosen Rücksetzen der Anzeige <POS erreicht>.
L   "SOLLWERT"       //Lade den Sollwert
L   "XPOS"           //Lade den Istwert
-D                   // bilde die Differenz
T   #MOTOR           //transferiere die Differenz als Formalparameter
L   "SOLLWERT"       //Lade den Sollwert
L   "XPOS"           //Lade den Istwert
==D                  //Wenn Gleichheit besteht....
S  "LEDPOSOK"        //... setze den LED-Ausgang
```

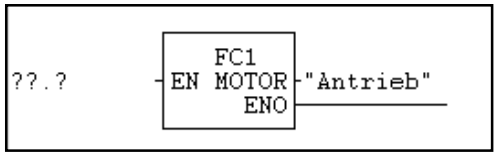
Das ist schon das gesamte Programm. Der Motor läuft abstandsabhängig. Für die Anzeige des Abstandes von der Sollwert-Position muss der Differenzwert noch vorzeichenlos umgewandelt werden. Die Methode entspricht dem Beispiel aus <Bahnsteuerung_2>.

FC 1; Netzwerk 2: Absolutwertbildung

```
    L   0            //Lade die Zahl 0
    L   #MOTOR       //Lade die Motordaten
    <D               //Ist 0 < #MOTOR?
    SPB   ok         //Wenn ja, springe zur Marke <ok:>.
    NEGD             // Sonst multipliziere den Wert mit –1.
    T   "ABSTAND"    //transferiere den (positiven) Wert an „ABSTAND“.
    BEA              //absolutes Baustein-Ende
ok: T   "ABSTAND"    //Sprungmarke. Wen oben schon ein pos. Wert anlag, wird dieser hier der
                     // Variablen „ABSTAND“ zugewiesen.
```

Der Aufruf der FC 1erfolgt vom OB 1.

OB 1; Netzwerk 1: Aufruf der Funktion FC 1



Vgl. TrySim-Projekt <Bahnsteuerung_2> im Verzeichnis <Lösungen>.

Projekt 44: Position-Steuerung (Erweiterung des Projektes 43)

Schwerpunkte: Einsatz von Wegaufnehmern, WORD-Verarbeitung, FC, NEGD, >D, SPB, SET

Aufgabe: Bahnsteuerung / Beschickungsautomat

Die Erfahrungen der <Bahnsteuerung_2> soll jetzt 2-dimensional angewendet werden.
Auf einem Montagetisch soll über einen 2-Achsen-Antrieb jede beliebige Position angesteuert werden. Für diese Demonstration werden über 2 Zufallsgeneratoren beliebige Positionen mit X- und Y-Koordinaten vorgegeben. Die Istwerte werden über Wegaufnehmer erfasst und mit den Sollwerten verglichen. Daraus werden die Antriebsrichtungen abgeleitet.
Zum ersten Start ist der <EIN>-Schalter zu betätigen. Durch Tipp auf "Neue Position" wird eine neue Koordinate erzeugt. Die X- und Y-Positionen werden separat angesteuert. Zur genauen Zielansteuerung wird die Bahngeschwindigkeit kurz vor dem Ziel gedrosselt, die "Schleichfahrt" beginnt. Die momentane Abweichung zur Sollpossition kann abgelesen werden. Wenn die Sollposition erreicht wird, leuchtet die entsprechende LED. Erst wenn beide Sollwerte erreicht worden sind, kann eine neue Koordinate gewählt werden.
In der Praxis würden die Beschickungskoordinaten aus einer Datei (einem Datenbaustein) ausgelesen werden.

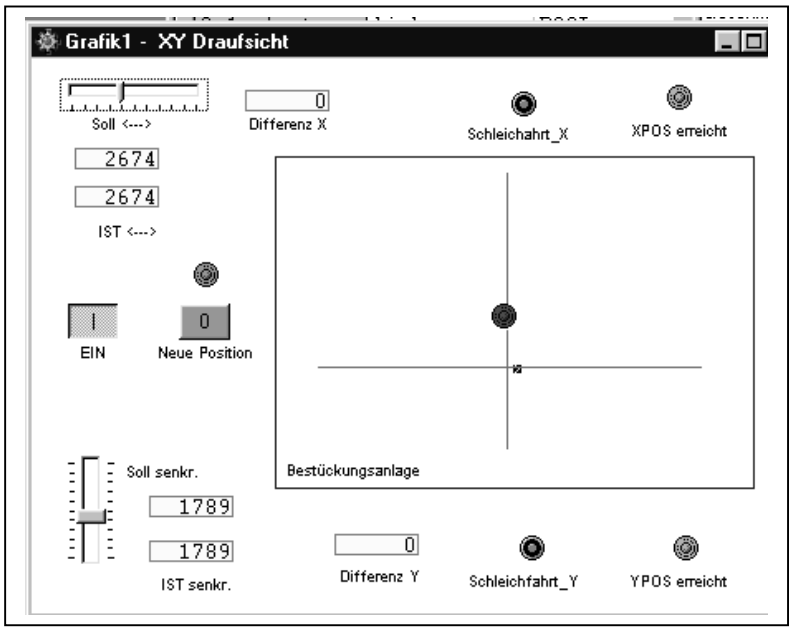
Theorie:

Vergleichen Sie bitte die Anmerkungen zum Projekt 38.

Auch wenn heutzutage derartige Aufgaben schon wieder anders (durch Servoantriebe) gelöst werden, ist dieses Beispiel doch zum Erlernen dieser logischen Verknüpfungen geeignet.

Zur Praxis mit TrySim:
Im Verzeichnis <Vorlagen> ist das Projekt unter <POSITION_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

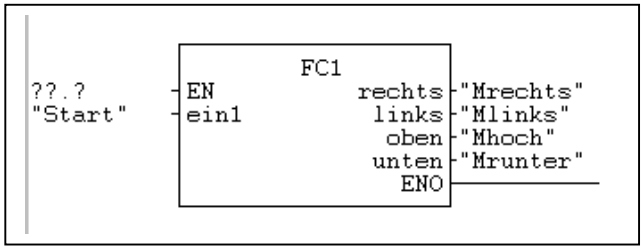
Anlage:



Symboltabelle:

Symbolische Adresse	Operand	Datentyp	Kommentar
Mrechts	A 0.0	BOOL	Antrieb Rechtslauf
Mlinks	A 0.1	BOOL	Antrieb Linkslauf
Blink	A 0.2	BOOL	Blinker
Mhoch	A 0.3	BOOL	Antrieb hoch
Mrunter	A 0.4	BOOL	Antreib runter
LEDSchlX	A 0.5	BOOL	LED Schleichfahrt X-Richtung
AnzXok	A 0.6	BOOL	X-Position erreicht
LEDSchlY	A 0.7	BOOL	LED Schleichfahrt Y-Richtung
AnzYok	A 1.0	BOOL	LED SollPos Y erreicht
NeuPosLD	A 1.1	BOOL	LED Neue Position wählen
Start	E 0.2	BOOL	Startschalter
NeuPos	E 0.5	BOOL	Neue Position wählen (S)
POSX	ED 1024	DWORD	Wegaufnehmer waagerecht
POSY	ED 1028	DWORD	Wegaufnehmer senkrecht
SollY	EW 510	WORD	Sollwert senkrecht
SollX	EW 514	WORD	Sollwert waagerecht
Schlf_Y	M 1.0	BOOL	Schleichfahrt Y-Richtung
Schlf_X	M 2.0	BOOL	Schleichfahrt X-Richtung
PosXok	M 9.2	BOOL	waager. SollPos X erreicht
PosYok	M 9.6	BOOL	senkr. SollPos Y erreicht
AnzXIst	MW 10	WORD	Anzeige Istwert waagerecht
AnzYIst	MW 15	WORD	Anzeige Istwert senkrecht
Diff_X	MW 23	WORD	Differenz Soll/Ist X
Diff_Y	MW 33	WORD	Differenz Soll/Ist Y

OB 1, Netzwerk 1: Aufruf der Funktion zum Ansteuern der Richtungen.



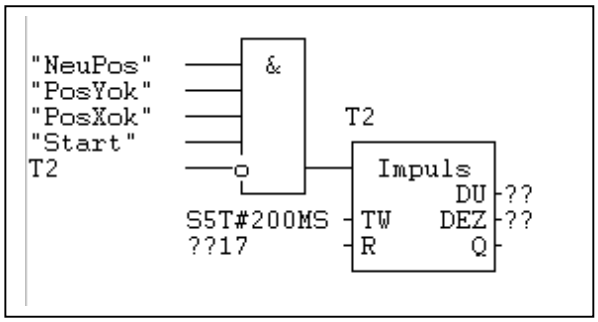
Das eigentliche Programm zum Ansteuern der Sollposition steht in der Funktion FC 1. Bevor Sie diese Funktion über das Icon <FUP-Elemente> und <FC> aufrufen, muss es vorher angelegt und abgespeichert worden sein.
Die Ein- und Ausgänge der FC werden mit den entsprechenden Klemmen (Operanden) der SPS belegt.

OB 1, Netzwerk 2: Ausschalten der Antriebe

```
U  "PosXok"    //Wenn der PosX-Merker "1" führt,...(wenn die Position erreicht wurde)
R  "Mrechts"   //...wird der Ausgang für RECHTS abgeschaltet
R  "Mlinks"    //...wird der Ausgang für LINKS abgeschaltet
U  "PosYok"    //sinngemäß wird der andere Achsenantrieb...
R  "Mhoch"     //          ...unabhängig von der FC ...
R  "Mrunter"   //                     ...abgeschaltet.
```

Durch dieses Rücksetzen außerhalb der FC wird garantiert, dass auf jeden Fall (auch wenn durch "schlechte" Programmierung eine FC-Anweisung durch einen Sprung nicht mehr gelöscht würde) die Antriebe ausgeschaltet werden. Vielleicht wundern Sie sich gelegentlich bei Programmierübungen, dass irgend ein Ausgang, den Sie gar nicht mehr verwenden, auf "1" gesetzt ist. Die Ursache wird sein, dass Sie vorher einmal diesen Ausgang auf "1" gelegt haben und ihn dann im Programm gelöscht haben. Solange Sie das Projekt nicht beenden, bleibt diese "Zelle" aktiv.

OB 1, Netzwerk 3: Impulsbildung, wenn Sollposition erreicht wurde.
Impulszeit, in der im nächsten Netzwerk die neuen Koordinaten berechnet werden.



Wenn die X- und Y-Positionen erreicht wurden (s. FC 1), der <EIN>-Schalter und der <Neue Position>-Taster betätigt wurden, gibt T2 ganz kurz einen Impuls, da durch diese "Selbstmordschaltung" die "1" am Eingang durch die negierte Abfrage von T2 verhindert wird. Die Zeitangabe an <TW> ist belanglos. Der Impuls entspricht nur einer Zykluszeit, reicht aber aus, um <POSYok> und <POSXok> im selben Zyklus in der FC 1 umzuschalten, so dass die Setzbedingung für T2 nicht mehr gegeben ist.

OB 1, Netzwerk 4: Zufallskoordinaten
Berechnung der neuen Koordinaten aus Zufallszahlen der Funktion FUNC 100.

```
      U T  2          //Wenn T2 eingeschaltet ist...
      SPBN ende       //...wird FUNC 100 aktiv; falls nicht, springe zur Marke <ende>.
      L  5200         //Aus dem max. Wert 5200...(entspricht der Weglänge in X-Richtung)
      FUNC 100        //bildet diese Sonderfunktion einen Zufallswert
      T  "SollX"      // und weist ihn dem Wort EW 514 zu.
```

Dieser Wert würde in der richtigen Anlage von außen (deshalb: EW) oder aus einem Datenbaustein abgefragt werden. Der "REGLER" in der Anlage kann als Sollwertgeber bedient werden oder - wie hier - als Anzeige. (vgl. TrySim-<Hilfe> <Index> "Regler").
Der eine Regler hat dieselbe Adresse wie die Digitalanzeige für den Sollwert in X-Richtung (EW 514), der andere Regler liegt parallel zur Digitalanzeige für den Sollwert in Y-Richtung (EW 510).

```
      L  3750         // (wie vorher, nur für die Y-Koordinate)
      FUNC 100
      T  "SollY"      // EW 510
ende: NOP 0           //Sprungmarke, falls T2 gerade nicht eingeschaltet ist.
```

Also nur in dem klitzekleinen Augenblick der Impulszeit werden die neuen Koordinaten bestimmt.

"FUNC 100" ist eine TrySim-interne Sonderfunktion und ist nicht auf eine SPS übertragbar. TrySim dient in erstere Linie dazu, Industrieanlagen vor dem Ernstfall zu simulieren. Aus diesem Grunde werden verschiedene Varianten, die in der Praxis durch andere Bedingungen auftreten können, hier als Zufallsparameter erzeugt (vgl. TrySim-<Hilfe> <Index> "FUNC").

OB 1, Netzwerk 5: Rücksetzen der alten Zykluswerte

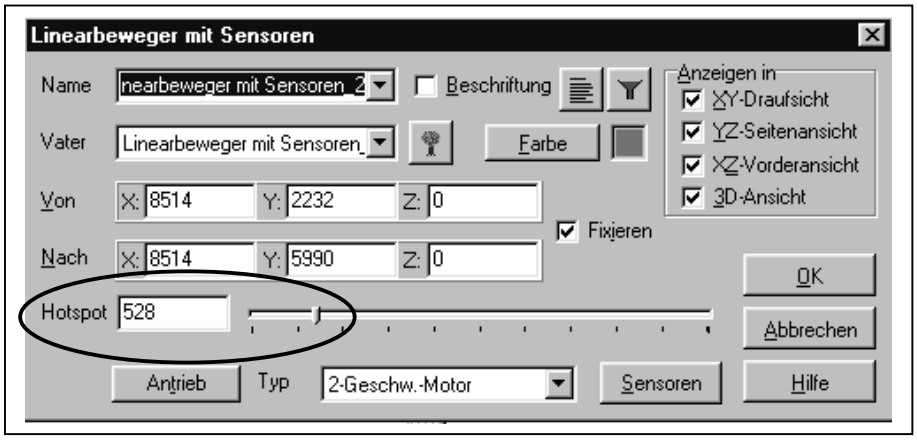
```
U T  2          //T2 wird nur gesetzt, wenn der Istwert gleich dem Sollwert ist. (s. OB 1,NW 3)
R "PosXok"      //Rücksetzen des Merkers
R "PosYok"
```

Die Merker werden aus der FC 1 beim Erreichen der Position gesetzt und dienen in OB 1, NW 3 der Freigabe des Timers T2, der wiederum in OB 1, NW 4 die neuen Koordinaten bestimmt. Durch das Rücksetzen wird das erneute Einschalten von T2 verhindert.

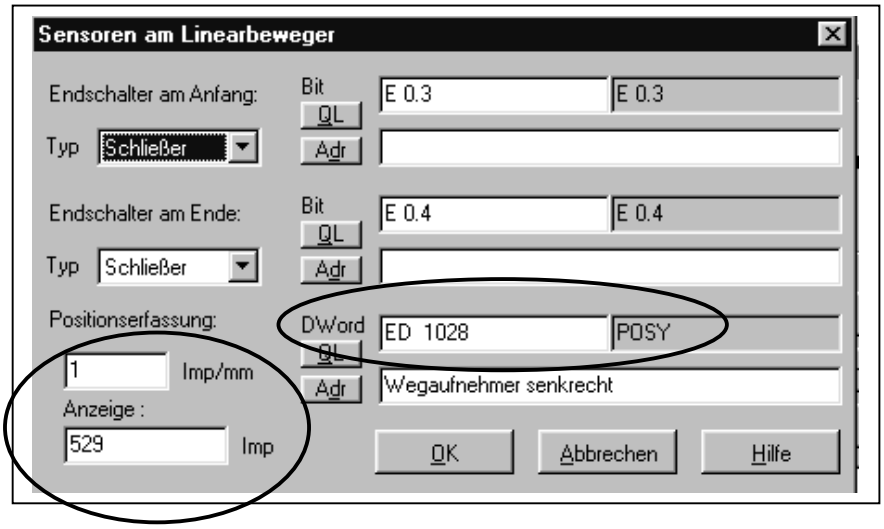
OB 1, Netzwerk 6: Zuweisung der IST-Werte zur Anzeige und Programm für die Schleichfahrt_X

```
L  "POSX"       //Laden des X-ISTwertes (ergibt sich aus der Stellung des Linearantriebes)
```

Klicken Sie bitte bei ausgeschalteter Anlage mit <rM> auf den Linearantrieb (den Strich). Es öffnet sich das Editierfenster des Antriebes.

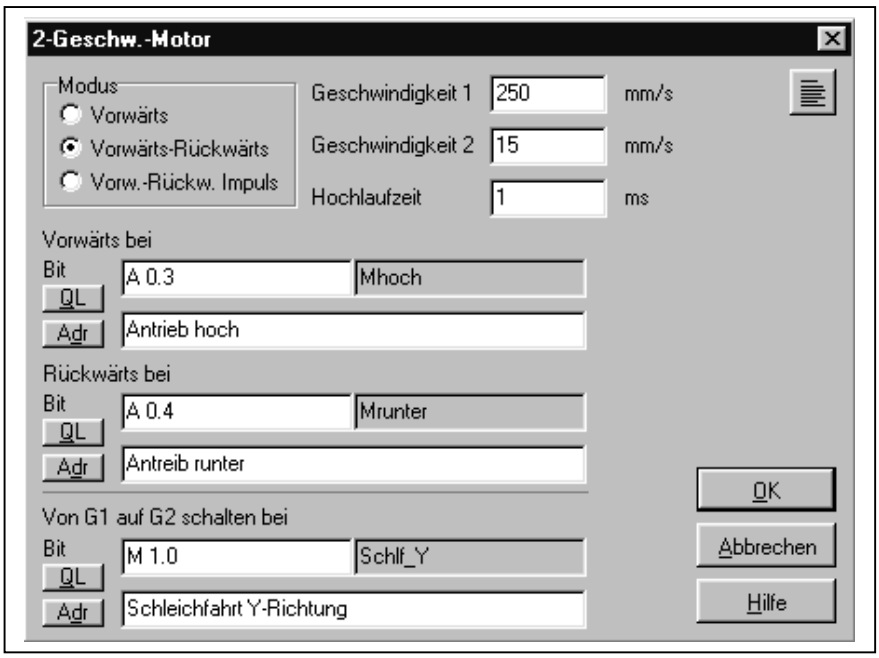


Sie erkennen im Hotspot den augenblicklichen Istwert, welcher der Schieberstellung entspricht. Auch hier können Sie den Schieber zum Verstellen benutzen. Sie erkennen ferner, dass als Antrieb der 2-Geschwindigkeiten-Motor ausgewählt wurde. Durch Drücken von <Sensoren> öffnet sich folgendes Fenster:



Mit <Imp/mm> können Sie den "Maßstab" einstellen. Die Impulszahl wird als DWORD erfasst (hier: ED 1028).

Wenn Sie oben in der Editiermaske auf <Antrieb> drücken, öffnet sich folgende Maske, die gleich eine Rolle spielt.



Sie können 2 Geschwindigkeiten einstellen und entscheiden, durch welches Ereignis von der ersten Geschwindigkeit auf die zweite umgeschaltet wird (hier: M 1.0).

Zurück zum Programm:

```
T   "AnzXIst"        //Zuweisen / Anzeigen des X-ISTwertes
L   "POSY"           //Laden des Y-ISTwertes
T   "AnzYIst"        //Zuweisen / Anzeigen des Y-ISTwertes
U  "Start"           //Wenn eingeschaltet wird,
R  "Schlf_X"         //...wird - egal, was vorher war - die Schleichfahrt beendet
R  "LEDSchlX"        //...und die Anzeige ausgeschaltet
L   "SollX"          //Lade die Soll-Position X
L   "POSX"           //Lade die Ist-Position X
-D                   //Differenzbildung zwischen Soll- u. Istwert X
T   "Diff_X"         //transferiere Differenz zum Merkerwort und zur Anzeige
```

Dies ist ein "Trick", um den (vorzeichenlosen) Absolutwert bei INT-Werten zu bilden.

```
L   0                //Lade den Wert "0"...
>D                   //...und prüfe, ob der erste Wert "Diff_X" größer ist als "0".
                     //Ist der Wert positiv, ...
SPB   okx            //...springe (bedingt) zur Marke "okx".
L   "Diff_X"         //Lade wieder den Differenzwert, der ja negativ ist (!)
NEGD                 //...und negiere den negativen Wert, d.h., er wird positiv.
T   "Diff_X"         //Trage den positiven Wert wieder als Operand "Diff_X" ein.
```

Dadurch wurde der Absolutwert gebildet. Egal, ob die Ist-Position "vor" oder "hinter" der Soll-Position liegt, wenn der Betrag kleiner als "30" ist, also die Ist-Position nahe am Sollwert liegt, soll die Schleichfahrt beginnen.

```
okx:  L   "Diff_X"   //Lade die jetzt absolute Differenz, den Abstand.
      L   30         //Lade den Wert "30"
      -D             //Vergleiche den Abstand mit der Zahl 30.
      L   30
      <D             //Ist der Istwert weniger als 30 vom Sollwert entfernt...
```

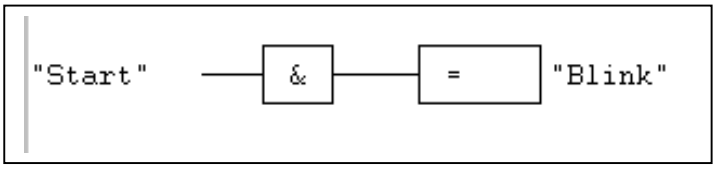
Beachten Sie bitte, dass der erste Wert nicht explizit geladen werden muss. Er liegt schon im AKKU aus der vorangegangenen Operation (-D). Wenn dieser erste Wert kleiner als "30" ist, ergibt "<D" als Ergebnis "1".
Damit wird die folgende Zeile verknüpft:

```
      UN "AnzXok"          //Wenn die X-Position noch nicht erreicht wurde, ...
                           //...(aber der Abstand klein ist)...
      S  "Schlf_X"         //...beginne die Schleichfahrt.
            // (Setze den Merker M 2.0, der als Umschalter für die Geschwindigkeit gilt).
      S  "LEDSchlX"        // Die Schleichfahrt wird angezeigt.
      U  "PosXok"          //Wenn in FC1 die SOLL-Position erreicht wurde...
      =  "AnzXok"          //...leuchtet die LED
```

OB 1, Netzwerk 7: Schleichfahrt_Y

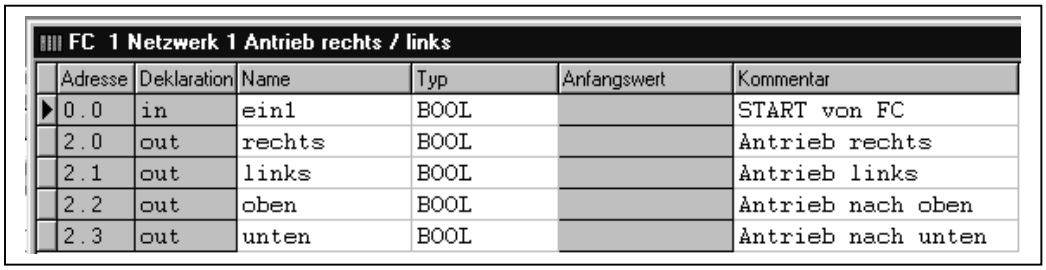
```
      U  "Start"           //Der Ablauf ist identisch mit der Schleichfahrt_X
      R  "Schlf_Y"         //Nur die Operanden sind entsprechend anders.
      R  "LEDSchlY"
      L   "SollY"
      L   "POSY"
      -D
      T   "Diff_Y"
      L   0
      >D
      SPB   oky
      L   "Diff_Y"
      NEGD
      T   "Diff_Y"
oky:  L   "Diff_Y"
      L   30
      -D
      L   30
      <D
      UN "AnzYok"
      S  "Schlf_Y"
      S  "LEDSchlY"
      U  "PosYok"
      =  "AnzYok"
      U  "PosXok"          //Wenn die X-Position...
      U  "PosYok"          // ...und die Y-Position in FC 1 erreicht sind...
      U  "Start"           // ...und der <EIN>-Schalter geschlossen ist, ...
      =  "NeuPosLD"        // ...leuchtet die LED für die Freigabe der neuen Koordinaten.
```

OB 1, Netzwerk 8: Der Blinker wird zugewiesen



Wenn der <EIN>-Schalter geschlossen ist, leuchtet die Blink-LED in der IST-Position.

Wechsel zur FC 1



FC 1 Netzwerk 1 Antrieb rechts / links

	Adresse	Deklaration	Name	Typ	Anfangswert	Kommentar
▶	0.0	in	ein1	BOOL		START von FC
	2.0	out	rechts	BOOL		Antrieb rechts
	2.1	out	links	BOOL		Antrieb links
	2.2	out	oben	BOOL		Antrieb nach oben
	2.3	out	unten	BOOL		Antrieb nach unten

Diese Kopfdaten sind vor dem Aufruf durch den OB 1 auszufüllen und zu speichern.

FC 1, Netzwerk 1: Antrieb rechts / links

```
SET                 //Ohne irgendeine Bedingung wird das VKE auf "1" gesetzt, so als
                    // wäre eine UND-Verknüpfung erfüllt. Erst unter dieser Bedingung
                    // kann das Programm formal richtig eine "="-Zuweisung (oder
                    // vergleichbare Operationen) durchführen. Sonst ist nicht sicher, dass
                    // die "R"-Operation der nächsten Zeile auch ausgeführt wird.
R  #rechts          //Setze den Antrieb für "Rechtslauf" zurück.
```

Sie müssen nur "r rechts" schreiben. TrySim erkennt die Operation "R" und ändert das Zeichen. Ferner wird erkannt, dass "rechts" als "out"-Parameter im Kopf der FC deklariert wurde. Um deutlich zu machen, dass dieser Parameter verwendet wird, setzt das Programm "#" automatisch davor. Im OB 1, NW 1 haben Sie schon diesen Parameter mit dem Operanden für den Rechtslauf verbunden.

```
R  #links           //Setze den Antrieb für "Linkslauf" zurück.
UN "Start"          //Wenn der <EIN>-Schalter nicht betätigt ist, ...
SPB endx            //...überspringe diesen Programmteil bis zur Marke <endx:>.
L   "SollX"         //Lade die Soll-Position X (1)
L   "POSX"          //Lade die IST-Position X  (2)
>D                  // Wenn (1) > (2), ist das VKE "1".
SPB   Xre           // Wenn diese Bedingung erfüllt ist, springe zur Marke <Xre:>.
L   "SollX"
L   "POSX"
<D                  // sonst prüfe, ob (1) < (2) (wenn ja, ist das VKE "1")
SPB   Xli           // Wenn diese Bedingung erfüllt ist, springe zur Marke <Xli:>.
==D                 // sonst prüfe, ob (1) == (2) (wenn ja, ist das VKE "1")
```

Wie Sie sehen, müssen die beiden Werte gar nicht "nachgeladen" werden; sie stehen noch im AKKU an.

```
S  "PosXok"         // Wenn diese Bedingung erfüllt ist, setze den Merker.
```

Da eine der vorstehenden Bedingungen erfüllt sein muss, ist spätestens hier Schluss (wenn die X-Position erreicht wurde). Deshalb...

```
    SPA   endx      //Springe ohne Bedingung zur Marke <endx>.

Xre:  SET           //muss nicht unbedingt sein, da von oben unter einer Bedingung
                    //gesprungen wurde.
    R  #links       //Falls vorher "Linkslauf" gesetzt war, wird er hier abgeschaltet...
    S  #rechts      //...und der "Rechtslauf" über den Parameter und Operand A 0.0
                    //gesetzt.
```

Da nach "Rechtslauf" nicht "Linkslauf" kommen kann, ist jetzt wieder Schluss.

```
    SPA   endx      //Springe ohne Bedingung zur Marke <endx>.
```

```
 Xli:  SET                    // (dasselbe Spiel für Linkslauf)
       R  #rechts
       S  #links
endx: NOP 0
```

FC 1, Netzwerk 2: Antrieb rauf / runter

```
       SET                    //(dasselbe Spiel für die senkrechte Achse - vgl. X/Y: FC 1, NW 1)
       R  #oben
       R  #unten
       UN "Start"
       SPB   endy
       L   "SollY"
       L   "POSY"
       >D
       SPB   Yo
       L   "SollY"
       L   "POSY"
       <D
       SPB   Yu
       ==D
       S  "PosYok"
       SPA   endy

 Yo:   SET
       R  #unten
       S  #oben
       SPA   endy
 Yu:   SET
       R  #oben
       S  #unten
endy: NOP 0
```

---

vgl. TrySim-Projekt <Position> im Verzeichnis <Lösungen>.

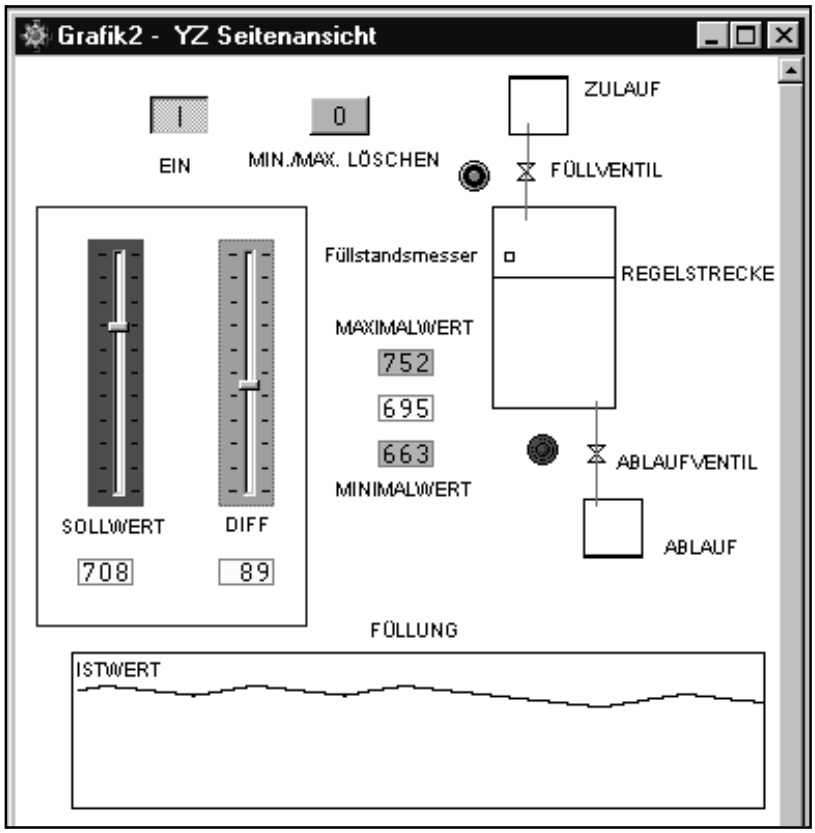
Projekt 45: Zweipunkt-Regler

Schwerpunkte: <I; >I, *I, +I, -I, /I, MOVE, L, T, R, S, SPBN, ENO/EN, MAX.-Wert-Anzeige

Aufgabe:
Ein Behälter soll über einen 2-Punkt-Regler bis zu einer beliebigen Höhe gefüllt werden.
Mit <EIN> wird der Ablauf konstant eingeschaltet. Der Zulauf wird programmabhängig geschaltet. Die Zulaufmenge ist konstant größer als die Abflussmenge. Bei eingeschaltetem Zulaufventil steigt also die Flüssigkeit gleichmäßig an. Die Schaltdifferenz ΔX- und damit der obere und untere Schaltpunkt – soll für Testzwecke a) absolut und b) prozentual eingestellt werden.
Es muss verhindert werden, dass [Sollwert + ΔX/2] > [Sollwert(max.)] wird. Ebenso darf [Sollwert - ΔX/2] nicht < [Sollwert(min.)] werden, damit die Umschaltpunkte nicht außerhalb der Regelstrecke liegen.
Die erreichten Minimal- und Maximalwerte sollen angezeigt werde. Mit Tasterdruck sind diese Werte zu aktualisieren..
Aus dem Projekt für die absolute Einstellung der Schaltdifferenz soll durch Ergänzungen das Projekt für die prozentuale Einstellung der Schaltdifferenz erstellt werden.

Durchführung des Projektes mit TrySim:
Im Verzeichnis <Vorlagen> ist das Projekt unter <Regel_2Pkt_V> vorbereitet. Laden und speichern Sie bitte die Vorlage in Ihrem Übungsordner unter einem sinnvollen Namen.

Die folgende Anlage und die Symboltabelle sind bereits vorbereitet.

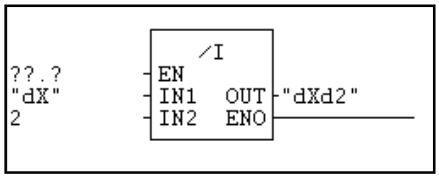


a) Absolute Einstellung der Schaltdifferenz

Symboltabelle

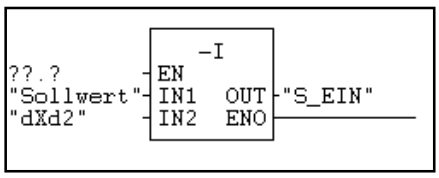
	Nummer	Symbol	Adresse	Typ	Kommentar
▶	1	FÜLLVENT	A 0.0	BOOL	Füllventil
	2	ABL_VENT	A 0.1	BOOL	Ablaufventil
	3	LED_ZUL	A 0.2	BOOL	LED Zulauf-Ventil
	4	LED_ABL	A 0.3	BOOL	LED Ablauf-Ventil
	5	EIN	E 0.0	BOOL	Ablaufventil EIN (S)
	6	MW_LOSCH	E 0.1	BOOL	MIN.-/MAX.-Wert löschen
	7	ISTWERT	EW 512	WORD	Istwert
	8	Sollwert	EW 514	WORD	Sollwert
	9	dX	EW 516	WORD	Schaltdifferenz
	10	S_EIN	MW 14	INT	Einschaltwert
	11	S_AUS	MW 16	INT	Auschaltwert
	12	dXd2	MW 20	INT	1/2 Schaltdifferenz
	13	MAXWERT	MW 500	INT	aktueller Maximalwert
	14	MINWERT	MW 512	INT	aktueller Minimalwert

OB 1; Netzwerk 1: Schaltdifferenz / 2



Die Schaltdifferenz wird durch 2 geteilt.
Die FUP-Elemente für die Division und Multiplikation finden Sie unter Icon <FUP-Elemente>.

OB 1; Netzwerk 2: Einschalt-Wert



"dX/2" ist für den Einschaltwert vom „Sollwert“ zu subtrahieren und für den oberen Ausschaltwert zum Sollwert zu addieren.

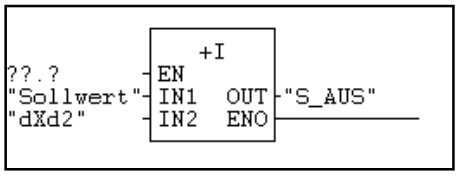
OB 1; Netzwerk 3: Minimaler Einschaltpunkt (Min.-Wert-Begrenzung)

```
L   50              //Lade die Zahl 50
L   "S_EIN"         //Lade den Wert, bei dem das Füllventil einschalten soll
>I                  //prüfe, ob 50 > „S_EIN“ ist
SPBN  end1          //wenn nicht, springe zur Marke <end1:>
L   50              //Sonst lade die Zahl 50 und
T   "S_EIN"         //transferiere die 50 nach „S_EIN“
```

(Dadurch wird die Berechnung so korrigiert, dass der minimale Einschaltwert innerhalb der Regelstrecke liegt).

end1: NOP 0

OB 1; Netzwerk 4: Ausschalt-Wert

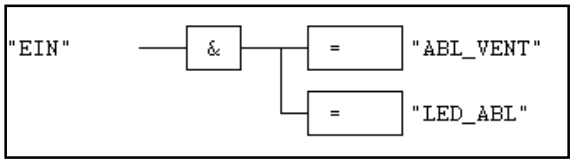


Der tatsächliche obere Ausschaltwert liegt nicht bei „Sollwert“, sondern liegt um „dXd2“ höher. Die Werte werden addiert zum Wert „S_AUS“.

OB 1; Netzwerk 5: -Maximaler Ausschaltpunkt (max._Wert-Begrenzung)

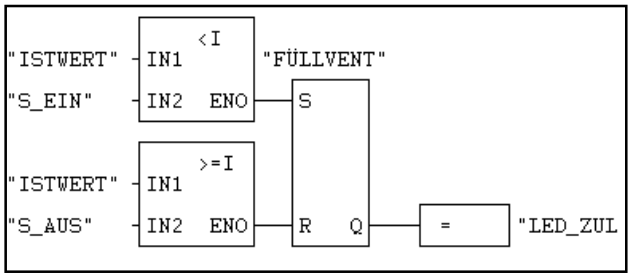
```
      L   "S_AUS"
      L   1000
      >I                    //prüfe, ob „S_AUS“ größer als 1000 ist.
      SPBN  end2
      L   1000              //wenn ja, lade 1000 und
      T   "S_AUS"           //transferiere den Wert nach „S_AUS“.
end2: NOP 0
```

OB 1; Netzwerk 6: Ablaufventil u. Anzeige



Mit „EIN“ wird das Ablaufventil und die LED eingeschaltet

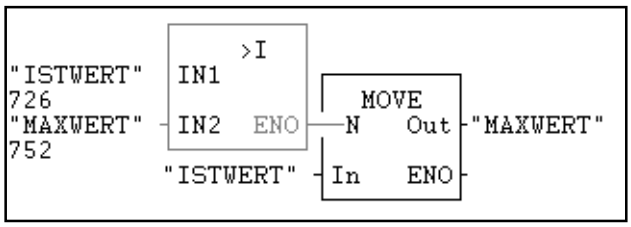
OB 1; Netzwerk 7: Füllventil



Wenn der Istwert kleiner als „S_EIN“ ist, wird das Füllventil geöffnet.

Wenn der Istwert größer als „S_AUS“ ist, wird das Füllventil geschlossen.

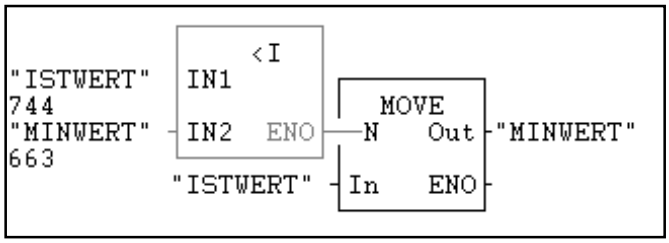
OB 1; Netzwerk 8: MAX.-Anzeige



Diese Kombination erlaubt eine Maximalwertanzeige.

Der Istwert wird mit dem gerade gültigen Maximalwert verglichen. Wenn der Istwert größer als der (alte) Maximalwert ist, wird der aktuelle Istwert in den Maximalwertspeicher geladen und stellt beim nächsten Zyklus den gerade gültigen Maximalwert dar.

OB 1; Netzwerk 9: MIN.-Anzeige



Sinngemäß ist auch eine Minimalwert-anzeige möglich.

OB 1; Netzwerk 10: Rücksetzen MIN. / MAX.-WERT

Bei veränderten Sollwerten oder aus sonstigen Gründen sollen die neuen Grenzwerte angezeigt werden. Dafür müssen die alten Werte gelöscht werden.

```
U  "MW_LOSCH"          //Wenn der Taster betätigt wird
   SPBN  nix           //(falls nicht, überspringe den Rest)
   L  0                //Lade den Wert 0 und
   T  "MAXWERT"        //ändere dementsprechend „MAXWERT“.
                       //(Die neuen – größeren - Istwerte überschreiben die 0).
   L  1000             // Lade die Zahl 1000 (höchster Füllstand) und
   T  "MINWERT"        //ändere dementsprechend „MINWERT“.
                       //(Die neuen – kleineren -Istwerte überschreiben die 1000).
nix:  NOP 0
```

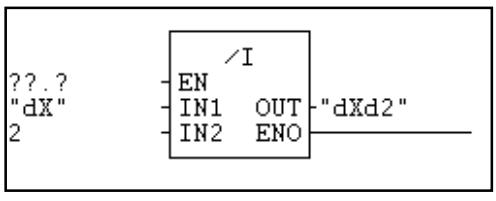
Die fertige Lösung finden Sie auch unter <Lösungen>|<Regel_2Pkt>

b) Prozentuale (relative) Einstellung der Schaltdifferenz

Nachdem das vorstehende Projekt funktioniert, legen Sie bitte davon eine Kopie an mit <Projekt> <Speichern unter...> und einem sinnvollen Projektnamen. Danach arbeiten Sie bereits in diesem neuen Projekt, dass vielleicht <2P_Regler_proz> heißt. Es soll genutzt werden, um einen „Umbau“ zu üben, bzw. vorzunehmen.
Das Prinzip der Aufgabenlösung wird im Fall b) nicht geändert. Nur die Wertermittlung der halben Schaltdifferenz wurde im neuen Netzwerk 2 geändert und zum Sollwert addiert oder subtrahiert.

Das bisherige Netzwerk 1 bleibt erhalten.

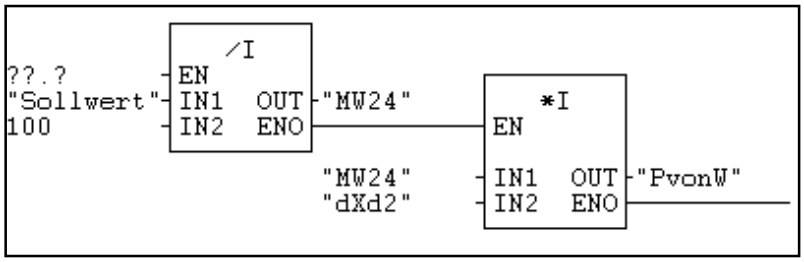
OB 1; Netzwerk 1: Schaltdifferenz / 2



Die Schaltdifferenz, die jetzt als prozentuale Abweichung vom Sollwert anzusehen ist, wird durch 2 geteilt. Technisch macht es nur einen Sinn, wenn dieser Wert kleiner als der Sollwert ist. Er wird in der Editiermaske für <Diff> auf “20“ (%) eingestellt.

„dX“ und „dXd2“ beziehen sich nach wie vor auf den Eingangsmesswert der zulässigen Schaltdifferenz, bekommt aber die Bedeutung der prozentualen Abweichung.
Es wird nur ein neues Netzwerk nach OB 1; Netzwerk 1 eingeschoben: Wählen Sie dazu das alte Netzwerk **2** an und wählen Sie <Bearbeiten>|<Neues Netzwerk>. Damit wird aus dem alten Netzwerk 2 jetzt das Netzwerk 3. Es wird also ein neues Netzwerk dazwischengeschoben und die folgenden Netzwerknummern passen sich automatisch an.

OB 1; Netzwerk 2 (neu): Prozentuale Abweichung



Das Netzwerk errechnet den prozentualen Anteil des Sollwertes nach der Beziehung:

„PvonW“ = („Sollwert“ / 100) * Prozentzahl „dXd2“

"Sollwert" wird als Eingangswort an <IN1> gelegt, "100" kann direkt an den Eingang <IN2> geschrieben werden.
„PvonW“ lässt sich natürlich nur an den Ausgang schreiben, wenn vorher in der Symboltabelle eine Zuordnung erfolge. Wie oder wo erfährt man, welcher Operand frei ist? Gefährlich wäre es, sich auf die Einträge in der Symboltabelle zu verlassen. Dort stehen nur „von Hand“ eingetragene Operanden! In die Adressentabelle werden zwar alle Ein- und Ausgangsoperanden automatisch angelegt, aber nur die Merkeroperanden, die in TrySim mit „Endgeräten“ verbunden sind. TrySim verzeiht diese – auch von mir verwendete – Schludrigkeit, praktisch müssten natürlich dafür EW- oder AW-Operanden benutzt werden. Ebenso werden keine Timer oder Zähler aufgelistet. Besser wäre es, alle Operanden und deklarierte Bausteine in die Symboltabelle aufzunehmen, damit andere Kolleginnen und Kollegen nicht irrtümlicherweise Operanden überschreiben. Eine vollständige Auflistung aller Operanden erhalten Sie unter <SPS>|<Querverweise>. Nachdem Sie sich dort überzeugt haben, dass MW 22 noch frei ist, können Sie es in der Symboltabelle verwenden. Sicherheitshalber tragen Sie dort auch MW 24 ein, obwohl es keinen Sinn macht, diese Bedeutung symbolisch zu erklären. Das Wort wird nur hier in diesem Netzwerk verwendet und erklärt sich selbst..
Durch die direkte Verbindung von <ENO> mit dem folgenden <EN> sind die Operationen miteinander verknüpft. Das MW 24 an <OUT> wird vom folgenden Element (im gleichen Netzwerk) weiter verarbeitet, wenn es dort an einem Eingang wieder aufgerufen wird. Ohne diese <ENO>-Verbindung könnte die Multiplikation auch in einem separaten Netzwerk folgen.

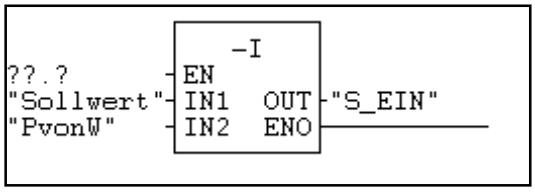
Die Symboltabelle sieht jetzt so aus:

neu

Symboltabelle

	Symbol	Adresse	Typ	Kommentar
	FÜLLVENT	A 0.0	BOOL	Füllventil
	ABL_VENT	A 0.1	BOOL	Ablaufventil
	LED_ZUL	A 0.2	BOOL	LED Zulauf-Ventil
	LED_ABL	A 0.3	BOOL	LED Ablauf-Ventil
	EIN	E 0.0	BOOL	Ablaufventil EIN (S)
	MW_LOSCH	E 0.1	BOOL	MIN.-/MAX.-Wert löschen
	ISTWERT	EW 512	WORD	Istwert
	Sollwert	EW 514	WORD	Sollwert
	dX	EW 516	WORD	Schaltdifferenz
	S_EIN	MW 14	INT	Einschaltwert
	S_AUS	MW 16	INT	Auschaltwert
	dXd2	MW 20	INT	1/2 Schaltdifferenz
	PvonW	MW 22	INT	Prozent vom Sollwert
	MW24	MW 24	INT	Verknüpfung
	MAXWERT	MW 500	INT	aktueller Maximalwert
	MINWERT	MW 512	INT	aktueller Minimalwert

OB 1; Netzwerk 3 (neu): Einschalt-Wert

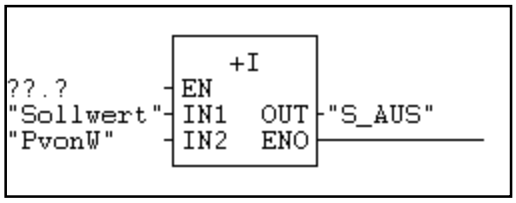


Hier wird nur an <IN2> der bisherige Wert „dXd2“ gegen „PvonW" ausgetauscht.

OB 1; Netzwerk 4 (neu):

Dieses Netzwerk wird einfach übernommen.

OB 1; Netzwerk 5 (neu): Ausschalt-Wert



Ebenfalls wird für den oberen Ausschalt-Wert an <IN2> der bisherige Operand "dXd2" gegen "PvonW" ausgetauscht.

Die anderen Netzwerke bleiben so erhalten, wie sie waren.
Die fertige Lösung finden Sie auch unter <Lösungen>|<Regel_2Pkt>

Vertiefung:

Unter bestimmten Bedingungen (überlegen bzw. kontrollieren Sie bitte selbst. Denken Sie dabei an geänderte Sollwerte.) kann der Minimal- oder der Maximalwert im neuen Zyklus falsch sein.

Zusatzaufgabe, um dieses Problem zu beseitigen:

Ohne den Rücksetztaster zu bedienen (er ist außer Funktion zu setzen) soll für jeden Füll-/Entleerzyklus der Maximal- u. Minimalwert angezeigt werden.

Dazu muss nur das NW 11 (neu) geändert werden:

OB 1; Netzwerk 11 (neu): Rücksetzen MIN. / MAX.-WERT

```
      U  "FÜLLVENT"        //Wenn das Füllventil eingeschaltet ist...
      FP M   30.0          //(der Flankenmerker sorgt für nur einmalige Ausführung)
      SPBN  weit           //(wenn die Bedingung nicht erfüllt ist, springe zur Marke <weit:>.
      L   0                //... lade (beim ersten Zyklus mit pos.Flanke) die Zahl 0
      T   "MAXWERT"        //und transferiere sie an die Variable.
```

(Hierdurch wird nur einmal – zu Beginn des Füllvorganges – *MAXWERT* auf 0 gesetzt. Anschließend wird *MAXWERT* von *ISTWERT* übernommen).

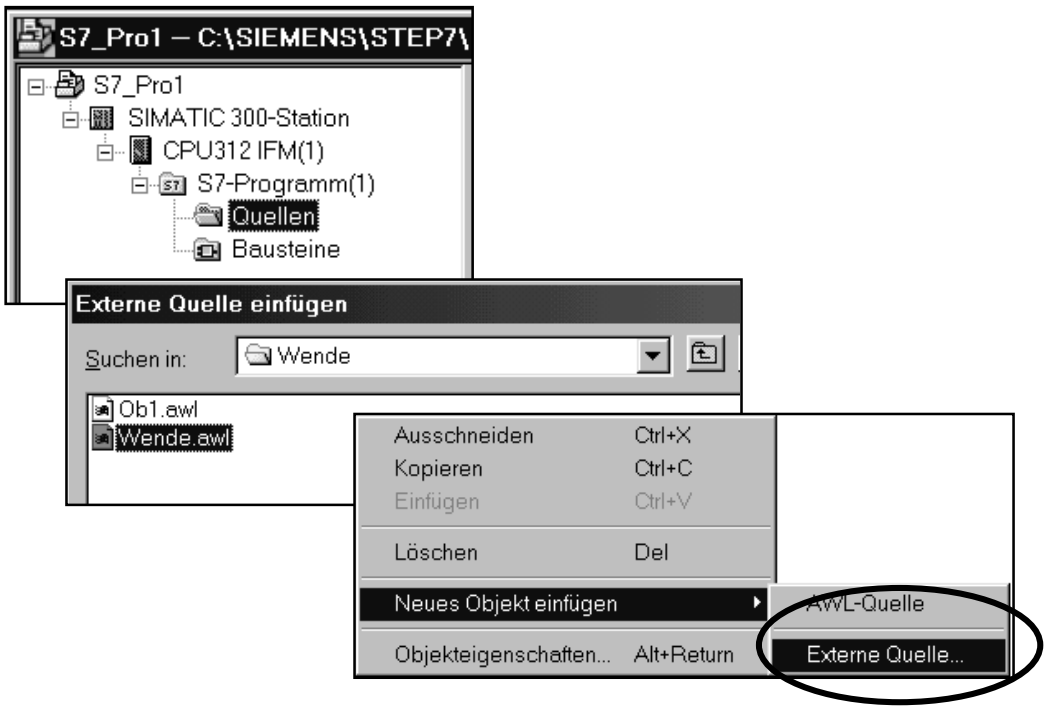
```
weit: UN "FÜLLVENT"        //Wenn das Füllventil ausgeschaltet wird...
      FP M   30.1          //(der Flankenmerker sorgt für nur einmalige Ausführung)
      SPBN  nix            //(wenn die Bedingung nicht erfüllt ist, springe zur Marke <nix:>.
      L   1000             //... lade (beim ersten Zyklus mit pos. Flanke) die Zahl 1000.
      T   "MINWERT"        //und transferiere sie an die Variable.
nix:  NOP 0                //Sprungmarke; Ende des Netzwerkes.
```

Die lauffähige Lösung finden Sie unter <Lösungen>|<Regel_2Pkt_auto>.

Projekt 46: Von TrySim® nach STEP®7

Es ist sehr einfach, das unter TrySim getestete Projekt in die Siemens-Steuerung zu laden – sofern Sie eine TrySim-Professional-Version erworben haben.
Hierzu wählen Sie bitte unter TrySim bei geöffnetem Projekt (hier als Beispiel: <Wende>) im Menü <Projekt>|<Exportieren>|<Bausteine>, wählen in dem sich öffnenden Fenster die Bausteine aus, die Sie exportieren möchten und drücken <OK>. Das war´s schon.
Der Rest spielt sich auf der Siemens-Seite ab. Nachdem Sie dort ein Projekt geöffnet haben, wählen Sie bitte die Quellen aus. Mit der rechten Maustaste öffnet sich das weitere Fenster.

Wählen Sie dort die externe Quelle...



...und wählen in dem TrySim-Ordner, in dem die Exportdatei angelegt wurde, die *.AWL, die so heißt, wie das unter TrySim exportierte Projekt (nicht OB1.awl !)

Dann klicken Sie bitte doppelt auf das (auf der rechten Seite) entstandene Symbol <Wende>.
Die folgende Übersetzung wird jetzt aus TrySim übernommen. Damit läuft das Programm allerdings noch nicht. (Keine Grund zur Sorge - wir sind ganz nahe dran, sofern die unter TrySim gewählten Operanden in Ihrer Siemensanlage tatsächlich vorhanden sind. Sonst müssten diese umbenannt werden).

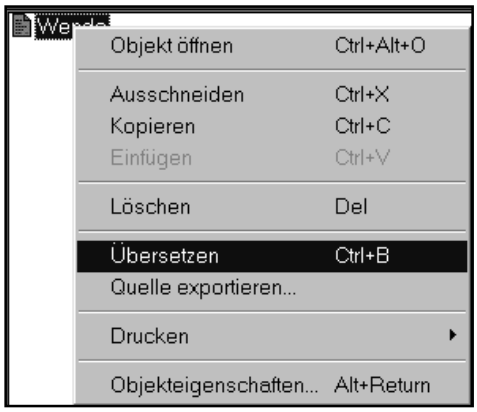
Um die Details der folgenden Darstellung müssen Sie sich gar nicht kümmern. Sie sind lediglich der Vollständigkeit halber aufgelistet.

```
ORGANIZATION_BLOCK OB 1
//

VAR_TEMP
  OB1_EV_CLASS : BYTE ;
  OB1_SCAN_1 : BYTE ;
  OB1_PRIORITY : BYTE ;
  OB1_OB_NUMBR : BYTE ;
  OB1_RESERVED_1 : BYTE ;
  OB1_RESERVED_2 : BYTE ;
  OB1_PREV_CYCLE : INT ;
```

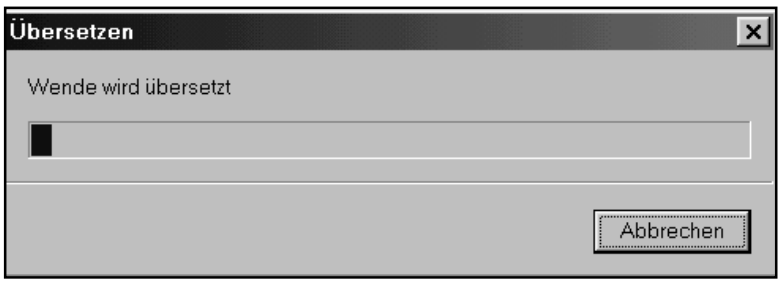
```
  OB1_MIN_CYCLE : INT ;
  OB1_MAX_CYCLE : INT ;
  OB1_DATE_TIME : DATE_AND_TIME ;
END_VAR
BEGIN
NETWORK
TITLE =Auswahl Hand
//Für das Programm <Handbetrieb> wird der Wahlschalter nicht gedrückt.
//Die LED zeigt den entsprechenden Zustand an.
      ON    E      0.6;
      =     A      0.5;
NETWORK
TITLE =Auswahl Automatik
//Für das Programm <Automatikbetrieb> wird der Wahlschalter  gedrückt.
//Die LED zeigt den entsprechenden Zustand an.
      O     E      0.6;
      =     A      0.6;
NETWORK
TITLE =Rechtslauf
//
      UN    A      0.1;
      U(    ;
      O     E      0.2;
      O     ;
      U     E      0.1;
      U     E      0.6;
      )     ;
      S     M      0.0;
      U(    ;
      ON    E      0.3;
      O     E      0.0;
      ON    E      0.5;
      )     ;
      R     M      0.0;
      U     M      0.0;
      =     A      0.2;
      =     A      0.0;
NETWORK
TITLE =Linkslauf
//
      U(    ;
      O     E      0.4;
      O     ;
      U     E      0.6;
      U     E      0.0;
      )     ;
      UN    A      0.0;
      S     M      0.1;
      U(    ;
      ON    E      0.3;
      O     E      0.1;
      ON    E      0.5;
      )     ;
      R     M      0.1;
      U     M      0.1;
      =     A      0.1;
      =     A      0.4;
NETWORK
TITLE =STOP-LED

      UN    A      0.0;
      UN    A      0.1;
      =     A      0.3;
```

```
END_ORGANIZATION_BLOCK
```

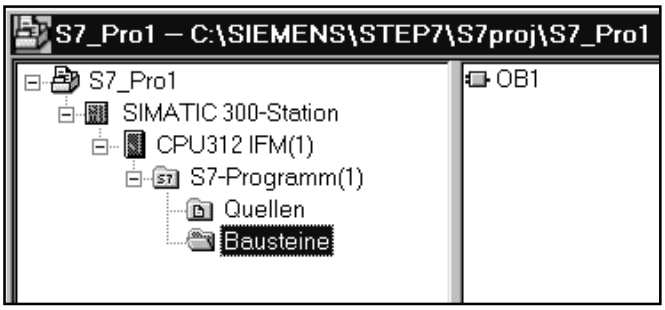
Wenn Sie nur 1-mal mit der rechten Maustaste klicken, öffnet sich folgendes Fenster. Wählen Sie <Übersetzen> aus.



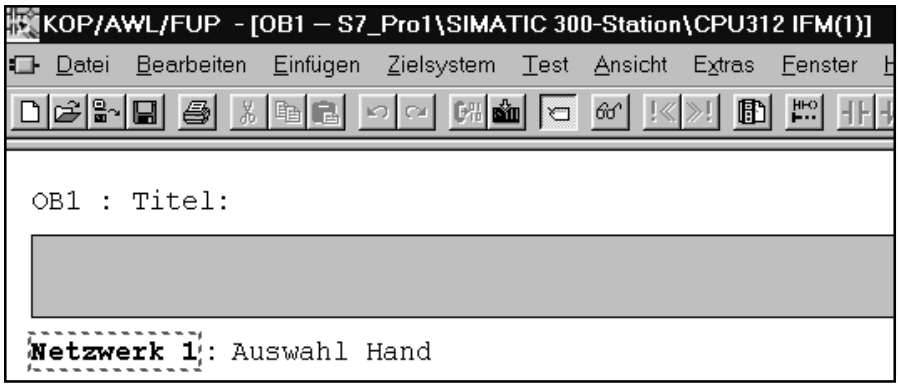
Kurzzeitig wird folgendes Fenster sichtbar:



Wenn Sie dann auf <Bausteine> klicken, sehen Sie das neue OB1-Symbol. Klicken Sie doch einmal doppelt darauf.



... und siehe da: die bekannte Struktur der AWL-Netzwerke ist angelegt. Sie sind in der Siemens-Welt angekommen.



Wenn Sie nicht schon gleich die entsprechenden Siemens-Operanden für die TrySim-Simulation verwendet haben, können Sie diese auch hinterher unter Step®7 anpassen.

Stichwortverzeichnis