

Peter Beater

---

# Grundkurs der Steuerungstechnik mit CODESYS

Grundlagen und Einsatz Speicherprogrammierbarer  
Steuerungen

Prof. Dr.-Ing. Peter Beater  
Fachhochschule Südwestfalen  
Fachbereich Maschinenbau-Automatisierungstechnik  
Lübecker Ring 2  
59494 Soest

1. Auflage  
Manuskript Version vom 1. 3. 2021

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils geltenden Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.

© Dr. Peter Beater, 2021

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften. Text und Abbildungen wurden mit größter Sorgfalt erarbeitet. Verlag und Autor können jedoch für eventuell verbliebene fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

ISBN 9783752661194  
Gedruckt von  
Books on Demand GmbH  
Gutenbergring 53  
D-22848 Norderstedt

# Was bringt Ihnen dieses Buch?

Es gibt heute fast keine Anlage oder Maschine mehr, in der nicht elektronische Steuerungen eingesetzt werden. Für Ingenieure und Studenten des Maschinenbaus ist es wichtig, dass sie den Entwicklern dieser Steuerungen eindeutig vorgeben können, welches Verhalten von der Maschine gefordert wird und mit der Steuerung realisiert werden muss. Ein geschriebener Text ist dabei selten zufriedenstellend für eine präzise Beschreibung, insbesondere wenn die Steuerungsvorgänge kompliziert sind und eine Auswahl von mehreren möglichen Abläufen und gleichzeitigen Aktionen bieten.

Ziel dieses Buchs ist es, Ingenieure und Studenten des Maschinenbaus in die Lage zu versetzen, eine eindeutige Beschreibung der Steuerungsaufgabe zu verfassen. Dazu ist ein Minimum an theoretischen Grundlagen und an Informationen über Aufbau und Funktionsweise heutiger Steuerungen erforderlich. Dieses Material wird im Folgenden präsentiert, soweit es zum Beschreiben von Ablaufsteuerungen erforderlich ist.

Die Steuerungstechnik ist keine rein theoretische Disziplin. Die Lösung einer Aufgabe setzt sich immer aus einem geeigneten Steuerungskonzept, der Auswahl der passenden Gerätetechnik und fehlerfreier Programmierung zusammen. In den ersten Kapiteln steht das Steuerungskonzept im Vordergrund. Im Kapitel 7 werden heutige Speicherprogrammierbare Steuerungen kurz beschrieben und in den folgenden Kapiteln dann weitere Programmiersprachen und Softwarekonzepte der europäischen Norm DIN EN 61131-3 vorgestellt.

Leser, die sich für die *Grundzüge der Steuerungstechnik*, aber nicht für die geräte-technische Realisierung interessieren, finden entsprechendes Material in den Kapiteln 1 bis 6. Sie können dabei die Unterkapitel überspringen, die sich mit Programmierdetails befassen.

Leser, die *CODESYS* näher kennenlernen wollen, finden entsprechendes Material z. B. in den Unterkapiteln 2.3, 3.7, 6.4 und 10.3.1 oder Aufgabe 5 – 1.

Leser, die einen kurzen *Überblick* über Aufbau und Arbeitsweise von *Speicherprogrammierbaren Steuerungen* erhalten wollen, finden entsprechendes Material in Kapitel 7.

Leser, die mit den Grundzügen der Steuerungstechnik vertraut sind und sich speziell für die *Programmiersprachen* und *Konzepte* der *DIN EN 61131-3* interessieren, können mit Kapitel 8 beginnen.

Dieses Buch basiert auf Erfahrungen am Fachbereich Maschinenbau-Automatisierungstechnik in Soest. Zu der Pflichtveranstaltung Steuerungstechnik gehört neben der zweistündigen Vorlesung eine einstündige Übung und ein einstündiges Laborpraktikum. In der Übung werden zusätzlich zu den abgedruckten Aufgaben weitere Aufgaben bearbeitet. Im Laborpraktikum arbeiten die Studenten dann mit typischen Kompaktsteuerungen.

Dieses Buch gibt im Wesentlichen den Inhalt der Vorlesung wieder. Die Umsetzung einer steuerungstechnischen Aufgabe in ein SPS-Programm wird in einer Reihe von Unterkapiteln dargestellt, die auch in den Gebrauch des Programmiersystems CODESYS einführen. Dieses System ist seit einigen Jahren im Labor in Soest erfolgreich im Einsatz. Da CODESYS einen eingebauten Simulator und eine Datenaufzeichnung zur Verfügung stellt, können die Beispiele und Aufgaben auch ohne SPS sinnvoll nachvollzogen werden.

Der Text dieses Buchs enthält eine Reihe von *Beispielen* mit vollständigem Lösungsweg, um die Theorie zu veranschaulichen. Die *Übungen* fordern Sie auf, über den Stoff nachzudenken und selbst tätig zu werden. Als Ergänzung ist eine Aufgabensammlung mit ausführlichen Lösungen ebenfalls bei BoD erschienen.

# Inhaltsverzeichnis

<b>1 Einführung Steuerungstechnik.....</b>	<b>1</b>
1.0 Inhalt dieses Kapitels.....	1
1.1 Abgrenzung Steuerung und Regelung .....	1
<b>2 Bausteine binärer Steuerungen .....</b>	<b>5</b>
2.0 Inhalt dieses Kapitels.....	5
2.1 Schaltglieder .....	5
2.2 Hinweis zu den Benennungen.....	13
2.3 CODESYS Beispiel zu Kapitel 2 .....	15
2.4 Aufgabe zu Kapitel 2.....	25
<b>3 Logische Verknüpfungen .....</b>	<b>27</b>
3.0 Inhalt dieses Kapitels.....	27
3.1 UND-Verknüpfung.....	27
3.2 ODER-Verknüpfung.....	29
3.3 Negation.....	30
3.4 NOR-Verknüpfung .....	30
3.5 NAND-Verknüpfung .....	31
3.6 Exklusiv-ODER-Verknüpfung.....	32
3.7 CODESYS Beispiel zu Kapitel 3 .....	34
3.8 Aufgaben zu Kapitel 3 .....	41
<b>4 Rechenregeln für Verknüpfungssteuerungen .....</b>	<b>47</b>
4.0 Inhalt dieses Kapitels.....	47
4.1 Verknüpfungssteuerungen .....	47
4.2 Aufgaben zu Kapitel 4 .....	54

<b>5 Speicher- und Zeitglieder .....</b>	<b>61</b>
5.0 Inhalt dieses Kapitels .....	61
5.1 Speicherglieder als Funktionsbausteine.....	61
5.2 Speicherglieder im Kontaktplan .....	64
5.3 Zeitglieder.....	68
5.4 Verwendung der Funktionsbausteine RS, SR und TON .....	70
5.5 Aufgaben zu Kapitel 5 .....	71
<b>6 Beschreiben von Ablaufsteuerungen.....</b>	<b>81</b>
6.0 Inhalt dieses Kapitels.....	81
6.1 Ablaufsteuerungen.....	81
6.2 Funktionsplan .....	82
6.3 Funktionsdiagramme.....	95
6.4 Erstellen von CODESYS-Programmen in Ablaufsprache (AS) .....	99
6.5 Aufgaben zu Kapitel 6 .....	104
<b>7 Speicherprogrammierbare Steuerungen (SPS) .....</b>	<b>117</b>
7.0 Inhalt dieses Kapitels.....	117
7.1 Entwicklung und Aufbau Speicherprogrammierbarer Steuerungen....	117
7.2 Programmierung .....	118
7.3 Hardware.....	119
7.4 Permanenter zyklischer Betrieb einer SPS .....	121
7.5 Betriebsarten.....	123
7.6 Meldungen .....	124
7.7 Steuerungssicherheit .....	125
7.8 Aufgabe zu Kapitel 7 .....	128
<b>8 Programmiersprache AWL (Anweisungsliste) .....</b>	<b>129</b>
8.0 Inhalt dieses Kapitels.....	129
8.1 Aufbau der Programmiersprache AWL.....	129
8.2 Boolesche Verknüpfungen.....	130
8.3 Bedingtes Setzen und Rücksetzen .....	132
8.4 Sprünge und Schleifen .....	134
8.5 Arithmetische Berechnungen.....	135
8.6 Aufruf von Funktionen .....	137
8.7 Aufruf von Funktionsbausteinen .....	138
8.8 Operatoren der Sprache AWL .....	140
8.9 Aufgaben zu Kapitel 8 .....	143

<b>9 Programmiersprache ST (Strukturierter Text).....</b>	<b>153</b>
9.0 Inhalt dieses Kapitels.....	153
9.1 Aufbau der Programmiersprache ST .....	153
9.2 Boolesche Verknüpfungen.....	154
9.3 Bedingtes Setzen und Rücksetzen .....	155
9.4 Schleifen und Sprünge .....	156
9.5 Arithmetische Berechnungen.....	161
9.6 Aufruf von Funktionen .....	162
9.7 Aufruf von Funktionsbausteinen .....	162
9.8 Operatoren der Sprache ST.....	164
9.9 Aufgaben zu Kapitel 9 .....	168
 <b>10 Programmstrukturierung mit DIN EN 61131-3 .....</b>	<b>173</b>
10.0 Inhalt dieses Kapitels.....	173
10.1 Programm-Organisationseinheiten (POE) .....	176
10.2 Datentypen und Variablen .....	186
10.3 Simultane Bearbeitung mehrerer Programme.....	189
10.4 Kommunikationsmodell der DIN EN 61131-3.....	193
10.5 Ausgewählte Funktionsbausteine .....	197
10.6 Was alles noch nicht gesagt wurde.....	204
10.7 Fragen zur Vorbereitung.....	205
 <b>Literatur .....</b>	<b>207</b>
 <b>Stichwortverzeichnis.....</b>	<b>209</b>





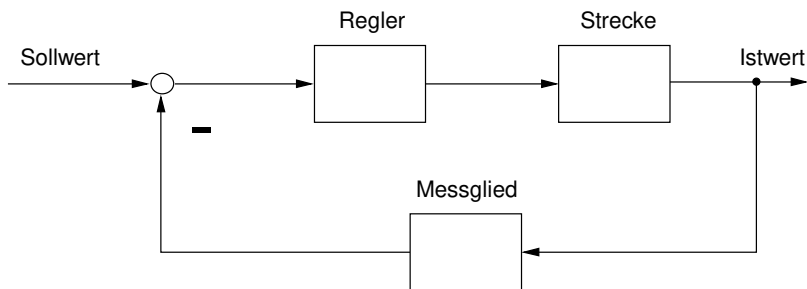
# 1 Einführung Steuerungstechnik

## 1.0 Inhalt dieses Kapitels

Dieses Kapitel gibt eine Übersicht über die in diesem Buch behandelten Steuerungen.

## 1.1 Abgrenzung Steuerung und Regelung

Häufig werden die Begriffe „Steuern“ und „Regeln“ in einem Atemzug genannt und fast als Synonyme gebraucht. Wir wollen aber folgende Unterscheidung machen: Bei einer Regelung geht es darum, die *Qualität* einer Größe sicherzustellen, z. B. eine Raumtemperatur von 20 °C. Der in Bild 1-1 gezeigte Regelkreis sorgt durch den Vergleich zwischen Sollwert und Istwert dafür, dass diese Vorgabe auch (möglichst gut) erfüllt wird. Die Analyse und Synthese von Regelungen erfordern spezifische Vorgehensweisen und werden daher in diesem Buch nicht behandelt<sup>1</sup>.



**Bild 1-1** Geschlossener Regelkreis

---

<sup>1</sup> Eine Einführung bietet z. B.: Lehr- und Übungsbuch zur Regelungstechnik: Eine beispielorientierte Einführung in die Theorie mit vielen gelösten Aufgaben, Peter Beater, BoD, 2019.

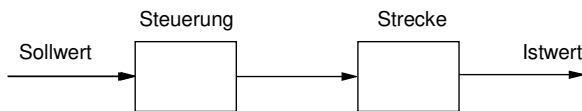
Kennzeichen dieses Regelkreises ist, dass an der *Vergleichsstelle* (Minuszeichen im Wirkungsplan) der Unterschied zwischen gewünschtem Wert (*Sollwert*) und tatsächlichem Wert (*Istwert*) gebildet wird und dieser Wert zur Ansteuerung der Strecke verwendet wird. Um einen kleinen *Regelfehler* zu erhalten, wird das Verhalten des Reglers auf die Strecke abgestimmt.

**Tabelle 1.1** Vor- und Nachteile einer Regelung

Vorteile einer Regelung	Nachteile einer Regelung
geringer Regelfehler erzielbar (statisch) z. B. konstante Raumtemperatur	hoher gerätetechnischer Aufwand z. B. Messglied, Regler
instabile oder schlecht gedämpfte Strecken können stabilisiert oder gut gedämpft werden z. B. ESP beim KFZ, verhindert Umfallen	Reglerentwurf teilweise aufwendig z. B. Arbeitspunktabhängigkeit bei nichtlinearen Strecken
Auswirkungen von Störungen werden reduziert	falsche Regler können das Verhalten verschlechtern; Instabilität der geregelten Strecke möglich

Bei vielen menschlichen Tätigkeiten handelt es sich um geregelte Vorgänge. So ist der senkrechte Gang nur möglich, weil im Ohr und Auge die Beschleunigung und Körperposition festgestellt werden und daraus vom Gehirn die entsprechenden Muskelbefehle bestimmt werden.

Bei vielen technischen Aufgaben sind die mit einer Regelung erzielbaren Vorteile so gering, dass man auf die Rückführschleife verzichtet und nur eine offene Steuerkette verwendet.



**Bild 1-2** Offene Steuerkette

Bei einer derartigen Struktur spricht man von einer *Steuerung* (im Gegensatz zu einer Regelung). Das Prinzip „Steuern“ kann ganz unterschiedlich aussehen. Eine klassische Fußgängerampel wird beispielsweise gesteuert. Nach Drücken der Taste wechselt die Lichtfolge für Fußgänger und Autofahrer nach einem festen Schema, ohne dass der Fußgänger darauf noch Einfluss nehmen könnte, z. B. durch langsames Überqueren der Fahrbahn.

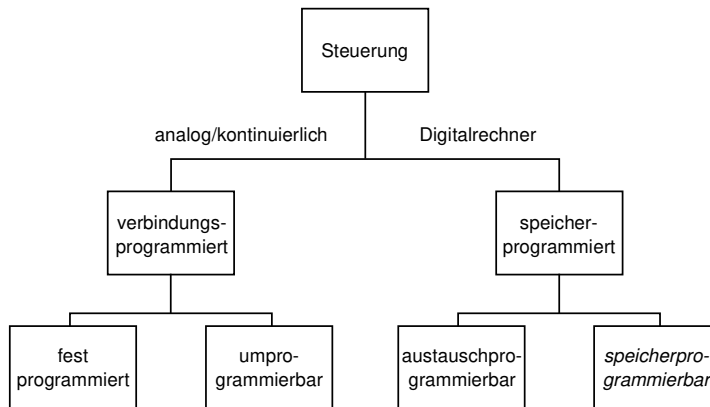
**Tabelle 1.2** Vor- und Nachteile einer Steuerung

Vorteile einer Steuerung	Nachteile einer Steuerung
geringer gerätetechnischer Aufwand	Störungen wirken sich unbeeinflusst aus
keine Stabilitätsprobleme	Verhalten instabiler oder schlecht gedämpfter Strecken kann nicht verbessert werden

Wichtig ist die Unterscheidung zwischen einer *kontinuierlichen Steuerung* (beliebig viele Zustände: Wasserhahn hat beliebig viele Zwischenschritte zwischen Auf und Zu) und einer *binären Steuerung* (genau zwei Zustände: Schalter hat genau zwei Stellungen, Ein und Aus). Im Folgenden werden binäre Steuerungen behandelt, die z. B. mit Schaltern, Schaltventilen oder auch elektronisch realisiert werden können.

Binäre Steuerungen werden in der Technik eingesetzt, um die *Abläufe* eines Prozesses sicherzustellen. Ein typisches Beispiel ist eine Waschmaschine: Es müssen nacheinander die Systemzustände Vorwäsche, Hauptwäsche, Spülen bzw. Schleudern verwirklicht werden.

Heute werden derartige Steuerungen elektronisch ausgeführt. Früher waren pneumatische Steuerungen weit verbreitet, die heute nur noch in Sonderfällen, z. B. in explosionsgefährdeten Bereichen, eingesetzt werden. Möglichkeiten der Programmverwirklichung bei einer elektrischen oder elektronischen Steuerung zeigt das folgende Bild.

**Bild 1-3** Systematik der Steuerungen

Bei *Verbindungsprogrammierten Steuerungen* ist das „Programm“ durch die Art der Komponenten und deren Verbindungen festgelegt. Bei einer konventionellen Pressen-Schützsteuerung steckt dieses Programm in der Verdrahtung der elektromechanischen Schütze. Man spricht dann von „fest programmiert“. Auch pneumatische oder hydraulische Steuerungen sind i. Allg. fest programmiert, da man die Schläuche üblicherweise nicht ändern kann. Bei einer altertümlichen Fernsprechvermittlung kann man die Verbindungen umstecken, so dass man von einer umprogrammierbaren Steuerung sprechen kann.

*Speicherprogrammierbare Steuerungen* werden i. Allg. mit Hilfe eines speziellen Digitalrechners realisiert. Das Programm ist als Folge von Rechenanweisungen im Speicher enthalten. Je nach der technischen Ausführung der Speicherbausteine kann es entweder überhaupt nicht (unveränderbar) oder recht einfach (frei programmierbar) geändert werden. Da wir uns in diesem Buch mit speicherprogrammierbaren Steuerungen befassen, ist dieses Adjektiv in Bild 1-3 hervorgehoben.

*Anmerkung:* Im täglichen Sprachgebrauch der Technik wird die oben definierte Unterscheidung zwischen Regelung (mit Rückführung) und Steuerung (ohne Rückführung) nur selten so strikt durchgeführt. Häufig spricht man von der Steuerung und meint „den Kasten“, der die Bedienereingaben, z. B. die Handhebel eines Baggers oder die Pedale eines Staplers, in Signale an die Verbraucher, z. B. Zylinder oder Motoren, umsetzt. Bei diesen Steuerungen handelt es sich dann häufig um umfangreiche Signalverarbeitung, die sowohl aus Regelkreisen als auch aus Steuerketten besteht. Sie enthält auch die Ansteuerung nachfolgender Leistungsteile, z. B. Ventile oder Motoren.

## 2 Bausteine binärer Steuerungen

### 2.0 Inhalt dieses Kapitels

Jede Disziplin hat ihre eigene Weltsicht und ihre eigene graphische Fachsprache. So werden für die Fertigung eines Bauteils technische Zeichnungen verwendet, die genauen Regeln unterliegen. In der Elektrotechnik wird der Stromlaufplan verwendet, der z. B. einfache Kontakte oder Verbraucher enthält. Dieses Kapitel beginnt mit diesen bekannten Komponenten. Im Vergleich dazu wird der Kontaktplan als eine allgemeinere Darstellungsform für Aufgaben der Steuerungstechnik vorgestellt.

### 2.1 Schaltglieder

Binäre Steuerungen bestehen aus mehreren *Schaltgliedern*, die zusammengeschaltet logische Verknüpfungen realisieren. Dabei können als Zustände nur jeweils „logisch 1“ oder „logisch 0“, „Ein“ oder „Aus“, „TRUE“ oder „FALSE“, „Spannung vorhanden“ oder „Spannung nicht vorhanden“ auftreten.

Das einfachste Schaltglied der Elektrotechnik ist ein Kontakt, der z. B. als Taster bei Haustürklingeln verwendet wird. Im *Stromlaufplan* wird er wie folgt dargestellt (DIN EN 60617-7):



**Bild 2-1** Klingeltaster und Schließersymbol im Stromlaufplan

Es handelt sich um einen *Schließer*, der im unbetätigten Fall offen ist und durch Betätigung geschlossen wird und dann Strom von einem Anschluss zum anderen leitet.

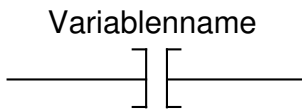
Der Stromlaufplan enthält viele Angaben zur gerätetechnischen Ausführung einer Steuerung, die für die Betrachtung ihrer Arbeitsweise nicht erforderlich sind. In der Steuerungstechnik hat sich daher der *Kontaktplan* (KOP) durchgesetzt, der nur die *logische Struktur* zeigt. Diese Darstellungsform hat ihren Ursprung in der Relais-technik, woher auch im Wesentlichen die Symbole stammen. Der Kontaktplan ist als *Programmiersprache* für Speicherprogrammierbare Steuerungen in DIN EN 61131-3 genormt.

*Im Kontaktplan verknüpft ein Kontakt den logischen Zustand der linken horizontalen Verbindung mit dem logischen Zustand der zugeordneten Variablen.*

Der Zustand der Variablen wird dabei nicht modifiziert. Die Variable beschreibt z. B. die elektrische Spannung an einer Eingangsklemme einer SPS. Sie ist im logischen 1-Zustand, wenn eine Spannung von 24 V anliegt und im logischen 0-Zustand, wenn eine Spannung von 0 V anliegt.

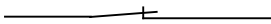
Wenn bei dem Kontakt in Bild 2-2 an der linken horizontalen Verbindung ein logischer 1-Zustand vorhanden ist und auch die Variable sich im logischen 1-Zustand befindet, liegt dieser logische 1-Zustand auch an der rechten horizontalen Verbindung an. Sonst ist die rechte horizontale Verbindung im logischen 0-Zustand.

In Analogie zum Stromlaufplan kann man sagen, dass der Pfad geschlossen wird, wenn die zugeordnete Variable im logischen 1-Zustand ist.



**Bild 2-2** Kontakt als Symbol im Kontaktplan

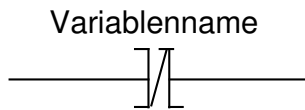
Neben Schließern gibt es im Stromlaufplan auch *Öffner*, die im unbetätigten Fall geschlossen sind und durch Betätigung geöffnet werden.



**Bild 2-3** Öffnersymbol im Stromlaufplan

Diese Negation können wir auch im Kontaktplan darstellen. Das entsprechende Kontaktplansymbol ist in Bild 2-4 dargestellt. Wenn bei dem Kontakt in Bild 2-4 an der linken horizontalen Verbindung ein logischer 1-Zustand vorhanden ist und die Variable sich im logischen 0-Zustand befindet, liegt an der rechten horizontalen Verbindung der logische 1-Zustand an. Sonst ist die rechte horizontale Verbindung im logi-

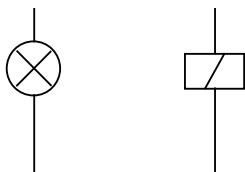
schen 0-Zustand. In Analogie zum Stromlaufplan kann man sagen, dass der Pfad geschlossen wird, wenn die zugeordnete Variable im 0-Zustand ist.



**Bild 2-4** Negierter Kontakt als Symbol im Kontaktplan

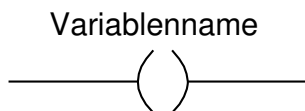
Bei den Signalgebern im Stromlaufplan kann es sich neben Tastern und Schaltern auch um magnetisch betätigte Kontakte (z. B. Reed-Schalter am Pneumatikzylinder), induktive oder kapazitive Näherungsschalter, Lichtschranken o. Ä. handeln.

Neben den Signalgebern müssen im Stromlaufplan auch die Verbraucher dargestellt werden. Typische Verbraucher sind die Signallampe, der Motor oder das Relais. Beim Relais wird durch Anlegen einer elektrischen Spannung die Relaisspule erregt, deren Magnetfeld den Anker anzieht, so dass die Kontakte geschlossen werden. Wird der Spulenstromkreis unterbrochen, bricht das Magnetfeld der Erregerspule zusammen und die Kontakte federn in die Ruhelage zurück. Oft werden mehrere Kontaktpaare gleichzeitig von einem Anker bedient. Ein Relais können wir als Leistungsverstärker betrachten. Bei Speicherprogrammierbaren Steuerungen dient das Relais zur galvanischen Entkopplung der Steuerung vom Prozess.



**Bild 2-5** Signallampe und Relais im Stromlaufplan

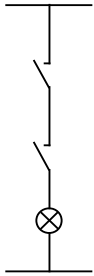
Das Gegenstück zu Verbrauchern im Stromlaufplan sind Spulen im Kontaktplan: *Mit einer Spule weisen wir den Zustand auf der linken Spulenseite der zugeordneten Variablen zu.* Diese Variable kann die Bezeichnung einer Ausgangsklemme einer SPS sein. Wenn wir ihr einen logischen 1-Zustand zuweisen, liegen an der Ausgangsklemme 24 V an. Bei dem logischen 0-Zustand sind es 0 V.



**Bild 2-6** Spule als Verbraucher im Kontaktplan

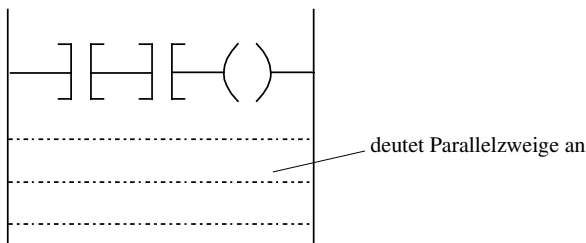
Der rechte Anschluss jeder Spule besitzt den gleichen logischen Zustand wie der linke Anschluss. In Analogie zum negierten Kontakt gibt es auch negierte Spulen: Mit einer negierten Spule weisen wir den negierten Zustand auf der linken Spulenseite der zugeordneten Variablen zu. Das Symbol zeigt Tabelle 2.2.

Die Regeln für das Zeichnen eines Stromlaufplans und eines Kontaktplans sind unterschiedlich. Beim Stromlaufplan zeichnet man die Elemente untereinander, wobei die Schalter über den Verbrauchern angeordnet werden. Beim Stromlaufplan muss in jedem Zweig immer ein Verbraucher sein, da sonst ein Kurzschluss auftreten kann.



**Bild 2-7** Beispiel für einen Stromlaufplan

Beim Kontaktplan zeichnen wir die Elemente von links nach rechts. Wir beginnen mit einer Verbindung von der immer links angeordneten *Stromschiene*, die im logischen 1-Zustand ist. Anschließend zeichnen wir in der Regel zuerst die Kontakte und dann die Spule. Die Schiene am rechten Rand des Kontaktplans dürfen wir auch weglassen. In einem Kontaktplan müssen mindestens die linke Stromschiene und eine Spule vorhanden sein, damit das SPS-System ihn auswerten kann.



**Bild 2-8** Beispiel für einen Kontaktplan







Die Zuordnung, welcher Gerätezustand welchem logischen Zustand zugeordnet ist, ist beliebig und wird von den zur Verfügung stehenden Komponenten und Betrachtungen des Fehlerfalls bestimmt. Dabei untersucht man, welche Auswirkungen ein Komponentenausfall oder ein Drahtbruch hat und entscheidet sich für die Zuordnung, die am sichersten ist. Die Booleschen Variablen werden häufig gemäß Tabelle 2.1 definiert.

**Tabelle 2.1** Variablendefinition

Boolesche Variable	Schalter	Grenzwert	Motor	Lampe	Pneumatik
1	geschlossen	erreicht	läuft	leuchtet	Druck > Atmosphärendruck
0	offen	nicht erreicht	steht	leuchtet nicht	Druck = Atmosphärendruck

**Tabelle 2.2** Definition der Zustände im Kontaktplan

	Spule	negierte Spule	Kontakt	negierter Kontakt
Symbol				
Funktion bei 1	<i>A14 ist bei 1-Signal am linken Anschluss logisch 1</i>	<i>A15 ist bei 1-Signal am linken Anschluss logisch 0</i>	<i>rechter Anschluss ist bei 1-Signal für A14 wie linker Anschluss</i>	<i>rechter Anschluss ist bei 1-Signal für A15 logisch 0</i>
Funktion bei 0	<i>A14 ist bei 0-Signal am linken Anschluss logisch 0</i>	<i>A15 ist bei 0-Signal am linken Anschluss logisch 1</i>	<i>rechter Anschluss ist bei 0-Signal für A14 logisch 0</i>	<i>rechter Anschluss ist bei 0-Signal für A15 wie linker Anschluss</i>

Bei den Spulen gibt es mit Setz- und Rücksetzspulen noch weitere Ausführungen. Diese werden im Kap. 5.2 behandelt.

Wir können den Kontaktplan als eine graphische Programmiersprache für Steuerungen betrachten. Die Eingangsgrößen sind die Spannungen an der Eingangsbaugruppe der SPS. Wenn die Spannung an einer Klemme 24 V beträgt, ist die zugeordnete Variable logisch 1; wenn die Spannung null ist, ist die Variable logisch 0. Durch die Anordnung der Kontakte und Zuweisung der Variablen bauen wir die logische Verknüpfung der Eingangssignale auf. Das berechnete Ausgangssignal weisen wir einer Klemme der Ausgangsbaugruppe der SPS zu. Zu einem vollständigen SPS-Programm gehört daher neben dem Kontaktplan auch immer eine Definition der verwendeten Variablen und ihre Zuordnung zu den Eingangs- bzw. Ausgangsklemmen der SPS.

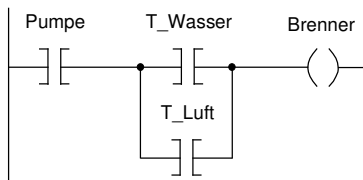
Der Kontaktplan ist eine Programmiersprache, um logische Verknüpfungen zu beschreiben. Der Stromlaufplan beschreibt Gerätetechnik. Jeder der beiden Pläne hat eine eigene Aufgabe und kann den anderen nicht ersetzen.

### Beispiel 2 – 1

Der Brenner einer Heizungsanlage soll eingeschaltet werden, wenn die Umwälzpumpe eingeschaltet ist und der Temperaturfühler für die Warmwasserversorgung oder der Raumtemperaturfühler anspricht.

Bereits dieses ganz einfache Beispiel zeigt uns, dass ein geschriebener Text selten zufriedenstellend ist für eine präzise Beschreibung von Steuerungsvorgängen. Denn es ist nicht eindeutig, ob es sich bei dem „oder“ um ein striktes „entweder ... oder“ handelt oder ob wenigstens eine der beiden Voraussetzungen erfüllt sein muss. Daher werden in Kap. 3 Schaltgleichungen als eine exakte mathematische Darstellungsform vorgestellt.

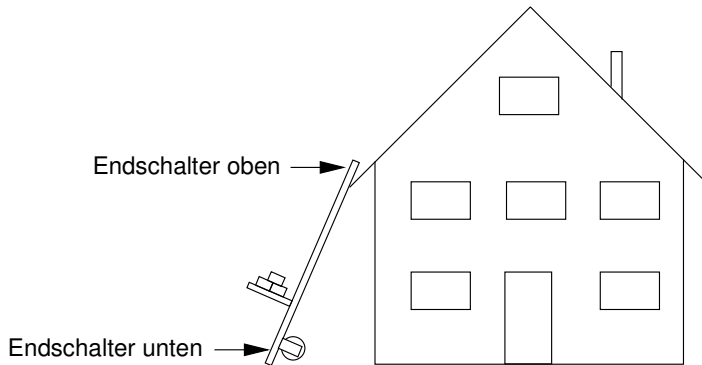
Bild 2-9 zeigt den Kontaktplan der Brennersteuerung. Als Variablennamen werden direkt die Komponentenbezeichnungen verwendet, wie wir es bei der Programmierung einer SPS auch tun werden. Die Variablen definieren wir gemäß Tabelle 2.1.



**Bild 2-9** Kontaktplan zum Beispiel Brennersteuerung

### Beispiel 2 – 2

In Bild 2-10 ist ein Bauaufzug schematisch dargestellt. Oben und unten gibt es jeweils zwei Taster, um den Aufzug nach oben bzw. unten zu fahren. Der Motor besitzt zwei Kontakte, wobei er entweder linksherum oder rechtsherum läuft. Zusätzlich befinden sich unten und oben Endschalter.



**Bild 2-10** Schematische Darstellung des Aufzugs

## Aufgabe

- Erstellen Sie eine Belegungsliste.
- Zeichnen Sie den Kontaktplan der Steuerung.
- Was passiert bei Fehlbedienung?

## Lösung

a) Die *Belegungsliste*<sup>2</sup> gibt an, welche Komponente wir an welche Klemme der SPS anschließen und wie wir sie im SPS-Programm benennen. Um Platz zu sparen, lassen wir meistens in diesem Buch die Spalte mit der SPS-Adresse, die sich direkt aus der belegten Klemme ergibt, weg.

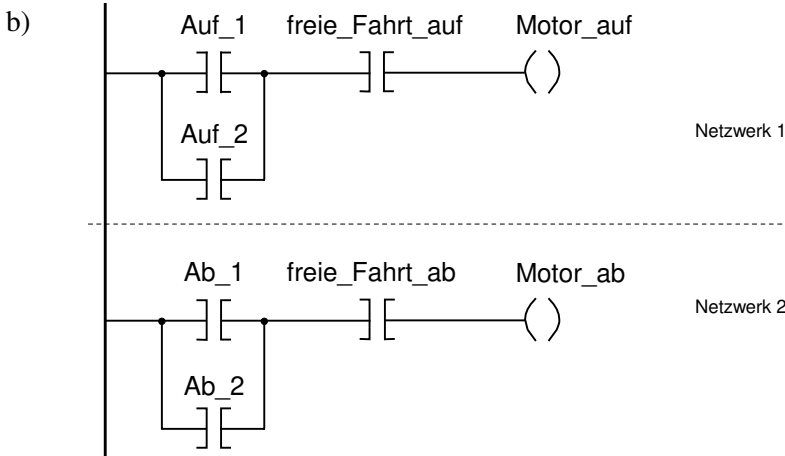
Ganz wichtig ist, dass wir in der Belegungsliste das logische Verhalten der Komponenten beschreiben. Also zum Beispiel, dass der Taster bei Betätigung eine Spannung an die SPS liefert, die dies als logisches 1-Signal umsetzt. Oder aber, dass der Motor den Aufzug nach oben zieht, wenn die Variable **Motor\_auf** logisch 1 ist, und dass er steht, wenn **Motor\_auf** und **Motor\_ab** beide logisch 0 sind.

<sup>2</sup> Gelegentlich wird auch der Begriff Zuordnungstabelle verwendet.

**Tabelle 2.3** Belegungsliste zum Beispiel 2 – 2

Komponente	Variable	Definition der logischen Zustände
Auf-Taster 1	<b>Auf_1</b>	Taster gedrückt => <b>Auf_1</b> = 1, sonst <b>Auf_1</b> = 0
Auf-Taster 2	<b>Auf_2</b>	Taster gedrückt => <b>Auf_2</b> = 1, sonst <b>Auf_2</b> = 0
Ab-Taster 1	<b>Ab_1</b>	Taster gedrückt => <b>Ab_1</b> = 1, sonst <b>Ab_1</b> = 0
Ab-Taster 2	<b>Ab_2</b>	Taster gedrückt => <b>Ab_2</b> = 1, sonst <b>Ab_2</b> = 0
Endschalter oben	<b>freie_Fahrt_auf</b>	Anschlag erreicht => <b>freie_Fahrt_auf</b> = 0, sonst <b>freie_Fahrt_auf</b> = 1
Endschalter unten	<b>freie_Fahrt_ab</b>	Anschlag erreicht => <b>freie_Fahrt_ab</b> = 0, sonst <b>freie_Fahrt_ab</b> = 1
Motorschalter aufwärts	<b>Motor_auf</b>	<b>Motor_auf</b> = 1 => Motor zieht aufwärts
Motorschalter abwärts	<b>Motor_ab</b>	<b>Motor_ab</b> = 1 => Aufzug fährt nach unten

Bei der Komponentenauswahl haben wir Öffner für die Endschalter gewählt. Dies widerspricht der Intuition, sorgt aber im Fehlerfall für mehr Sicherheit. Wenn z. B. das Kabel abreißt, bekommt die Steuerung ein Signal, das das Erreichen des Endanschlags anzeigt. Ein Fahren in diese Richtung wird dann verhindert. Diese Sicherheitsüberlegungen fasst man unter dem Stichwort *Drahtbruchsicherheit* zusammen (Was soll passieren, wenn der Draht bricht? Welcher Zustand ist dann am ungefährlichsten?).

**Bild 2-11** Darstellung der Steuerung als Kontaktplan

Wenn wir diesen Kontaktplan in einer Speicherprogrammierbaren Steuerung programmieren, sollten wir das Programm in zwei *Netzwerke* aufteilen. Ein Netzwerk ist für eine Speicherprogrammierbare Steuerung ein zusammenhängender Teil der Gesamtsteuerung, bei dem alle Ausgangsgrößen sich eindeutig aus den jeweiligen Ein-

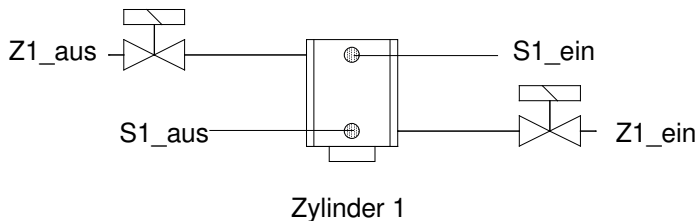
gangsgrößen ergeben. Wenn wir in einem Netzwerk nur eine Spule verwenden, ist diese Bedingung erfüllt.

c) Diese Steuerung ist sehr anfällig gegen Fehlbedienung. Wenn sowohl ein Auf-Taster als auch gleichzeitig ein Ab-Taster gedrückt werden, bekommt der Motor zwei Signale, sowohl Aufwärtsfahrt als auch Abwärtsfahrt. Dies ist nicht zulässig und daher muss die Steuerung ergänzt werden (Stichwort: Verriegelung).

## 2.2 Hinweis zu den Benennungen in diesem Buch

Viele Aufgaben in diesem Buch enthalten eine technische Skizze, um den Aufbau zu veranschaulichen. In der Regel handelt es sich dabei um einen oder mehrere Antriebe, die angesteuert werden müssen und deren Endposition erfasst wird. Um die Ventile bzw. Endschalter einfach den Antrieben zuordnen zu können, bekommt jeder Antrieb eine Nummer. In dem Beispiel aus Bild 2-12 ist es die Nummer **1**. Da es sich um einen Zylinder handelt, heißt die Komponente **z1**.

Das Ventil, das für ein Ausfahren der Kolbenstange sorgt, bezeichnen wir als **z1\_aus** und es ist *symbolisch* im Bild 2-12 dargestellt. Dies ist *nicht* das korrekte pneumatische Schaltzeichen und diese Ansteuerung ist bei einem Pneumatikzylinder auch nicht richtig, da wir die entgegengesetzte Zylinderkammer entlüften müssen. Den Endschalter, der die ausgefahrene Kolbenstange meldet, bezeichnen wir mit **s1\_aus**, da es sich um einen Sensor handelt, der bei der Komponente 1 den ausgefahrenen Zustand signalisiert.



**Bild 2-12** Benennungen der Komponenten in diesem Buch

**Tabelle 2.4** Komponenten und das jeweilige Kurzzeichen in diesem Buch

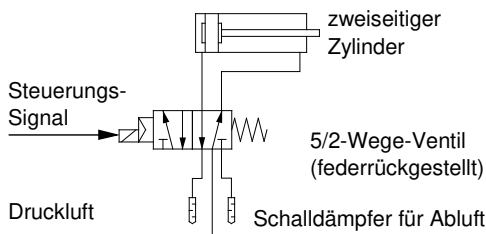
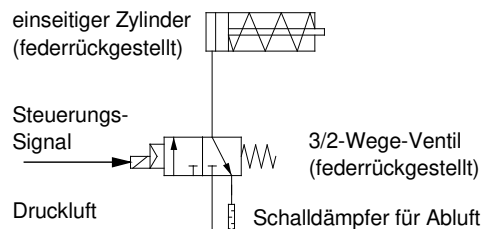
Komponente	Zeichen
Pneumatikzylinder	Z
Elektromotor	M
Endschalter, Sensor, Schalter	S
Taster	T
Lampe	L
Kontakt	K

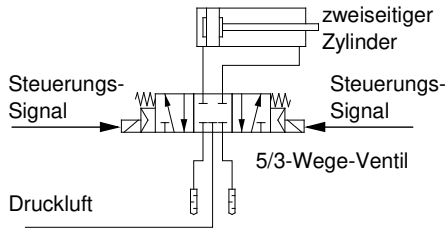
Zur Ansteuerung von Pneumatikzylindern verwenden wir Schaltventile, die direkt von der SPS angesteuert werden. Aus Sicht der Steuerungstechnik müssen wir zwei Fälle unterscheiden:

*Monostabile 5/2-Wegeventile* besitzen fünf Pneumatik-Anschlüsse, zwei Ventil-Stellungen und eine Ventilschule, die von der SPS angesteuert wird. In der einen Ventilstellung fahren sie die Kolbenstange ein, in der anderen aus. Es gibt nur *ein Eingangssignal* für das Ventil. Ob der Zylinder bei einem logischen 1-Signal einfährt oder ausfährt, hängt vom Aufbau ab und muss in der Belegungsliste definiert werden!

Diese Ventile werden häufig für pneumatische Antriebe eingesetzt, die im Störfall eine definierte Stellung einnehmen sollen. Man kann sie verwenden, um den Zylinder einzufahren und so den Arbeitsbereich zu verlassen, wenn der elektrische Strom ausfällt.

Das Verhalten Ventil-Zylinder ähnelt hier einem einseitig wirkenden Zylinder mit Rückstellfeder. Bei dem beaufschlagen wir eine Zylinderkammer z. B. bei einem logischen 1-Signal mit Druckluft und entlüften sie entsprechend bei einem logischen 0-Signal. Dies ist mit einem 3/2-Wegeventil in Bild 2-14 dargestellt.

**Bild 2-13** 5/2-Wegeventil**Bild 2-14** 3/2-Wegeventil




**Bild 2-15** 5/3-Wegeventil

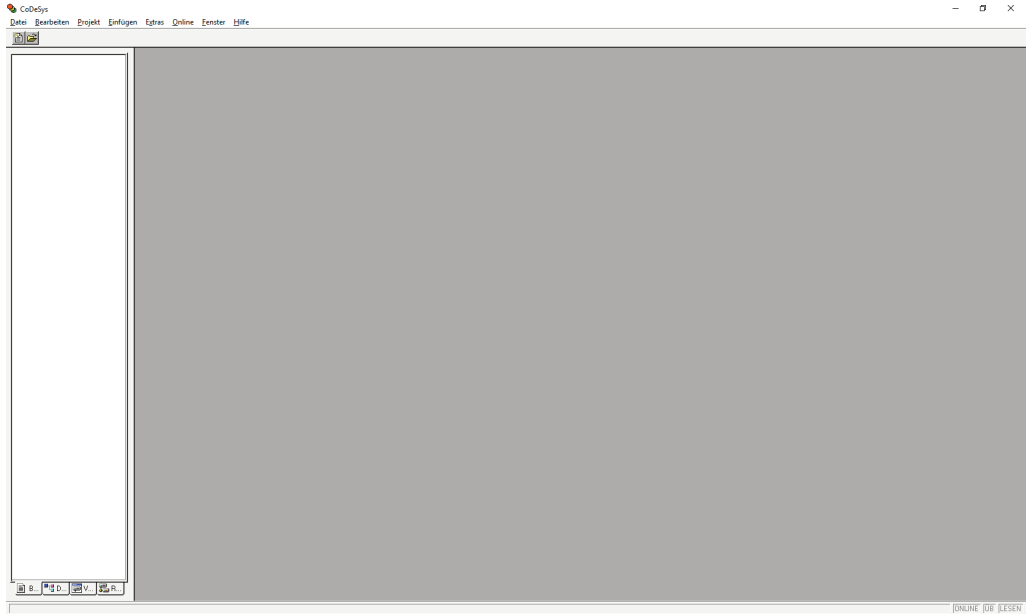
*5/3-Wegeventile* besitzen fünf Pneumatik-Anschlüsse, drei Ventil-Stellungen und zwei Ventilspulen. In einer Ventil-Stellung fahren sie die Kolbenstange ein, in der Mittelstellung bleibt die Kolbenstange stehen und in der dritten Stellung fahren sie die Kolbenstange aus. Es gibt zwei Ventilmagnete und daher *zwei Eingangssignale* für das Ventil. Die Mittelstellung wird eingenommen, wenn keiner der beiden Magnete angesteuert ist.

Bei elektrischen Antrieben haben wir aus Sicht der Steuerung häufig das gleiche Verhalten: Bei einem logischen 1-Signal an einem Anschluss dreht sich der Motor vorwärts, bei einem logischen 1-Signal an dem anderen Anschluss dreht sich der Motor rückwärts. Liegt kein Steuersignal an, bleibt der Motor stehen. Durch die Steuerung muss gewährleistet werden, dass nicht beide Eingänge gleichzeitig angesteuert werden.

Dieses Bezeichnungsschema wurde für dieses Buch gewählt, um eine Komponente und ihre Funktion bzw. ihre Wirkungsrichtung einfach und kurz miteinander zu verknüpfen. Im industriellen Umfeld sind andere Kennzeichnungen gebräuchlich und z. B. in der Norm DIN EN 61346 festgelegt.

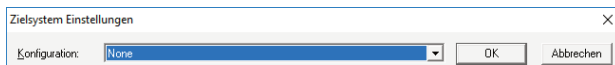
## 2.3 CODESYS Beispiel zu Kapitel 2

Den Kontaktplan aus Bild 2-11 wollen wir mit CODESYS programmieren und im Simulator überprüfen. Wir gehen davon aus, dass CODESYS neu installiert wurde und nun zum ersten Mal aufgerufen wird, Bild 2-16. Das Hauptfenster ist noch leer und wir können entweder aus der Menüleiste den Befehl **Datei** oder aus der Funktionsleiste das Symbol  auswählen.

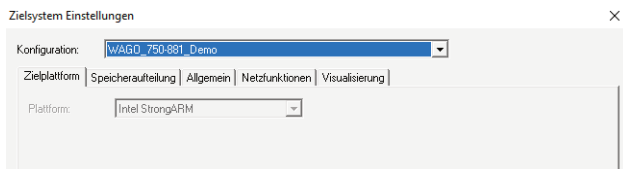


**Bild 2-16** Erster Aufruf von CODESYS nach der Installation

Mit dem Befehl **Datei -> Neu** öffnet sich die Programmierfläche für Bausteine. Da aber noch kein Zielsystem bei der Installation ausgewählt wurde, ist dies jetzt erforderlich. Im Labor verwenden wir das Wago System 750\_881\_Demo.



**Bild 2-17** Zielsystem auswählen

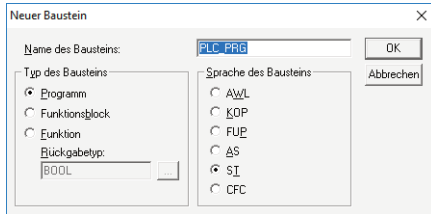


**Bild 2-18** Zielsystem im Labor WAGO 750-881



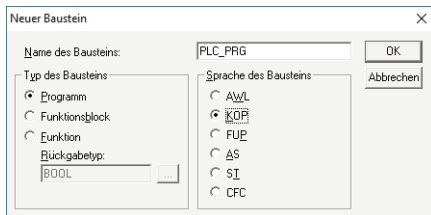
An den vorgegebenen Einstellungen sollte man am Anfang keine Änderungen vornehmen. Wenn keine SPS eingesetzt werden soll, kann man **none** oder ein beliebiges Zielsystem auswählen und später die Simulation auf dem PC nutzen<sup>3</sup>.

Als nächstes erscheint das Menü zur Konfiguration des Bausteins PLC\_PRG. Dies ist eine Vorlage, die bei einfachen Aufgaben eine Menge Arbeit erspart.

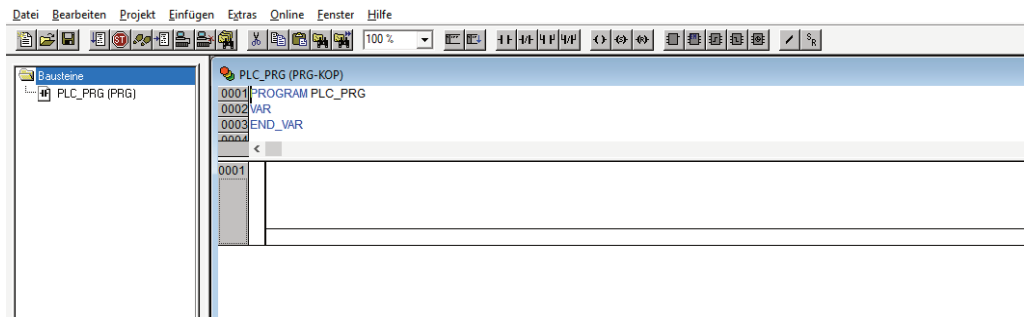


**Bild 2-19** Neuer Baustein

Wir wählen als Sprache KOP und es erscheint der entsprechende Editor.



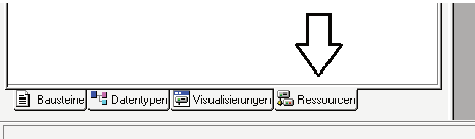
**Bild 2-20** Baustein in KOP



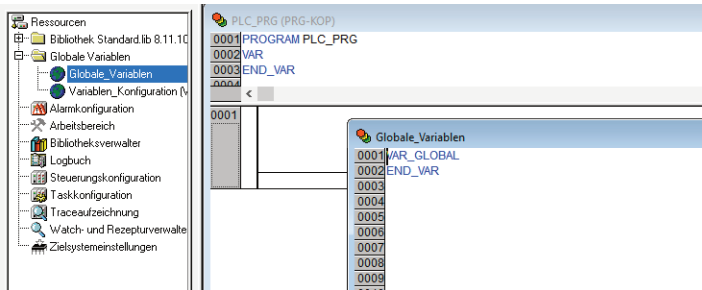
**Bild 2-21** Editor für KOP

<sup>3</sup> Bei **none** ist der Simulator im Menü **online** bereits fest eingestellt. Je nach Programmiersprache muss man dann noch die Bibliothek **standard.lib** mit Hilfe des Bibliotheksverwalters laden, s. a. Kap. 6.4.

Da wir einige globale Variablen benötigen, ist es arbeitssparend, diese als nächstes im entsprechenden Fenster einzugeben. Dazu wählen wir unten links den Reiter **Ressourcen** und dann im Menü oben **Globale Variable**.

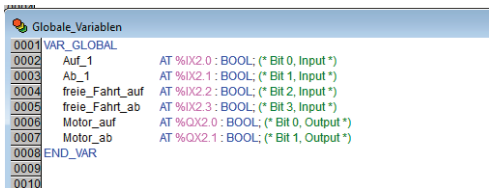


**Bild 2-22** Reiter Ressourcen zur Eingabe der globalen Variablen



**Bild 2-23** Globale Variable Eingabemaske

Hier geben wir die globalen Variablen für das Beispiel ein.



**Bild 2-24** Globale Variablen für das Beispiel

In Textform lautet die Liste:

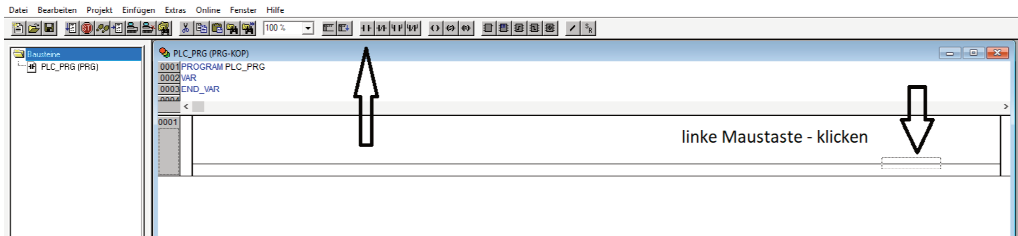
#### **VAR\_GLOBAL**

<b>Auf_1</b>	<b>AT %IX2.0 : BOOL; (* Bit 0, Input *)</b>
<b>Ab_1</b>	<b>AT %IX2.1 : BOOL; (* Bit 1, Input *)</b>
<b>freie_Fahrt_auf</b>	<b>AT %IX2.2 : BOOL; (* Bit 2, Input *)</b>
<b>freie_Fahrt_ab</b>	<b>AT %IX2.3 : BOOL; (* Bit 3, Input *)</b>
<b>Motor_auf</b>	<b>AT %QX2.0 : BOOL; (* Bit 0, Output *)</b>
<b>Motor_ab</b>	<b>AT %QX2.1 : BOOL; (* Bit 1, Output *)</b>

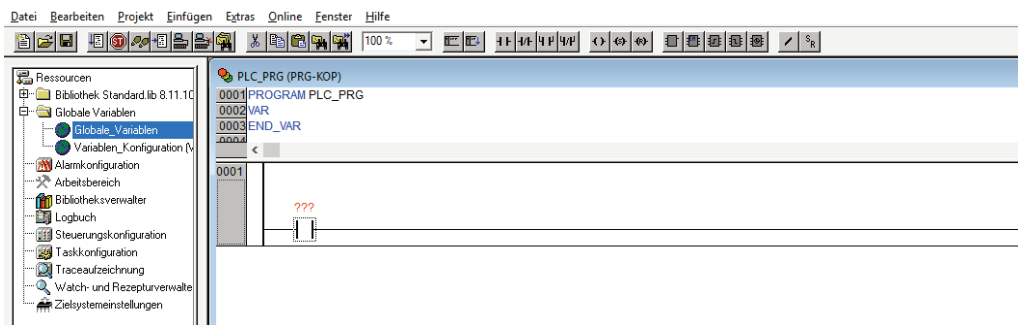
#### **END\_VAR**

Das Schlüsselwort **AT** gibt die Klemme an, an der das entsprechende Kabel angeschlossen wird. Mit **%IX2** wird die jeweilige Gruppe der Eingänge beschrieben; mit **%QX2** die Gruppe der Ausgänge. Die Zahl nach dem Punkt gibt die jeweilige Position an.

Wir kehren zu dem Fenster **PLC\_PRG** zurück und klicken mit der linken Maustaste auf das Eingabefeld und wählen dann einen Kontakt aus, **IE**. Dieser wird im ersten Netzwerk links eingefügt.

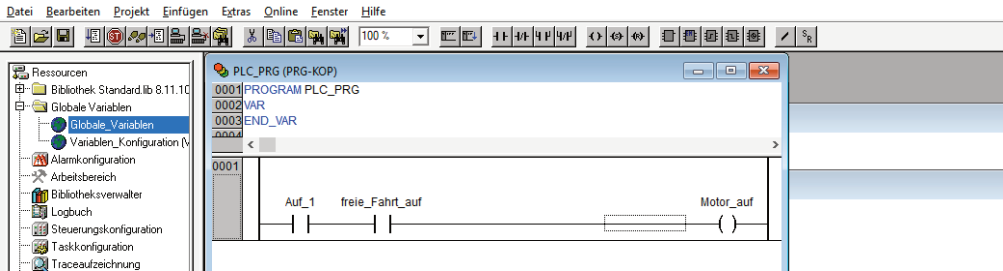


**Bild 2-25** Fenster **PLC\_PRG** um Kontakt einzufügen



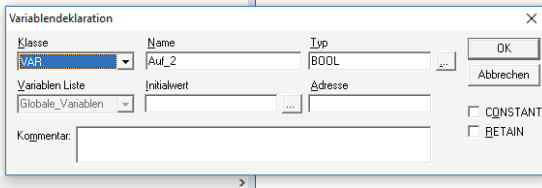
**Bild 2-26** Fenster **PLC\_PRG** mit erstem Kontakt

Um dem Kontakt eine Variable zuzuordnen zu können, klicken wir mit der linken Maustaste auf die drei roten Fragezeichen, **???**, um diese zu markieren, und dann drücken wir die Funktionstaste **F2**. Es erscheint die Eingabehilfe. Wir wählen die Liste der globalen Variablen, aus der wir dann **Auf\_1** auswählen. Danach ergänzen wir den Kontakt **freie\_Fahrt\_auf** und die Spule **Motor\_auf**.



**Bild 2-27** Zwei Kontakte und eine Spule

Um die Parallelschaltung zu programmieren, markieren wir den Kontakt **Auf\_1** und verwenden wir dann das dritte Symbol in der Funktionsleiste, . Diesen Kontakt wollen wir **Auf\_2** nennen und als lokale Variable definieren. Wir geben den Namen in das Feld mit den drei Fragezeichen, **???**, ein und drücken die Eingabetaste: bzw. **Enter**. Es erscheint ein Menü zur Variablendefinition, das wir mit OK bestätigen. Dann fügt CODESYS die lokale Variable **Auf\_2** in den Deklarationsteil ein<sup>4</sup>.



**Bild 2-28** Definition der lokalen Variablen **Auf\_2**

Da die Struktur für den Abwärtsbetrieb identisch ist, klicken wir mit der rechten Maustaste in das Netzwerk, wählen kopieren, schieben den Mauszeiger nach unten, klicken rechts und wählen einfügen. Dann ändern wir die Variablennamen. Dabei ist es erfahrungsgemäß sinnvoll, mit der Funktionstaste **F2** sich das Auswahlménü anzeigen zu lassen, um Schreibfehler zu vermeiden. Die Variable **Ab\_2** müssen wir ergänzen.

<sup>4</sup> Der Name darf keine Leerstellen und keine Umlaute oder Sonderzeichen enthalten und nicht identisch mit einem Schlüsselwort sein. Zwischen Groß- und Kleinschreibung bei Variablen wird nicht unterschieden, das heißt VAR1, Var1 und var1 sind keine unterschiedlichen Variablen. Einfache Unterstriche, **\_**, sind erlaubt, mehrfach aufeinander folgende Unterstriche sind nicht zulässig. Die Länge des Namens ist in CODESYS unbegrenzt.

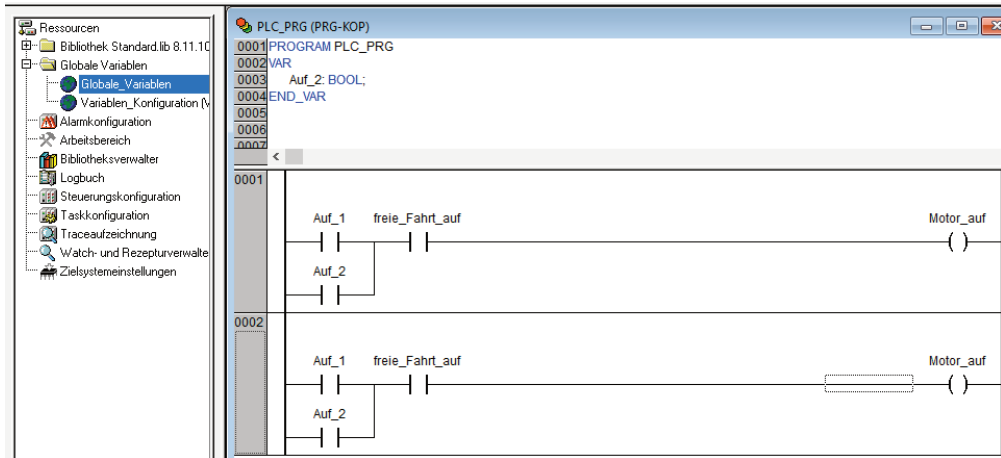


Bild 2-29 Netzwerk 1 kopiert

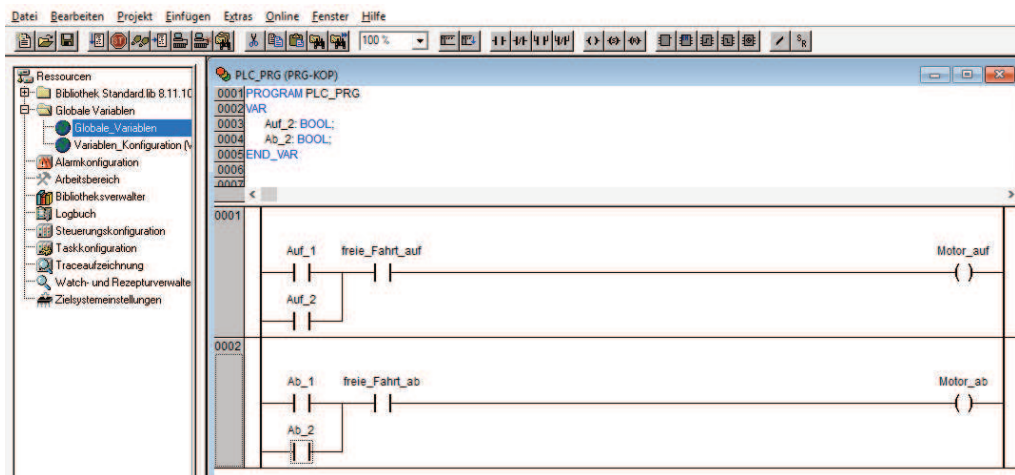


Bild 2-30 Netzwerk 2 mit geänderten Variablennamen

Über den Befehl **online** in der Menüleiste gelangen wir in ein Menü, in dem wir **Simulation** anklicken<sup>5</sup>. Das Programm wird nun nicht in die SPS geladen, sondern im PC ausgeführt. Danach wählen wir **Einloggen** und **Start**.

<sup>5</sup> Wenn keine SPS am Anfang ausgewählt wurde, ist **Simulation** bereits ausgewählt.

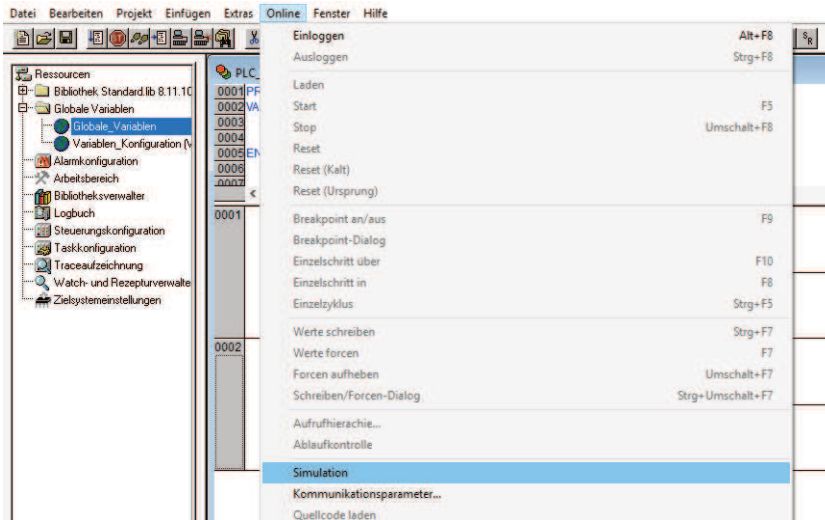


Bild 2-31 Auswahl der Simulation unter Online

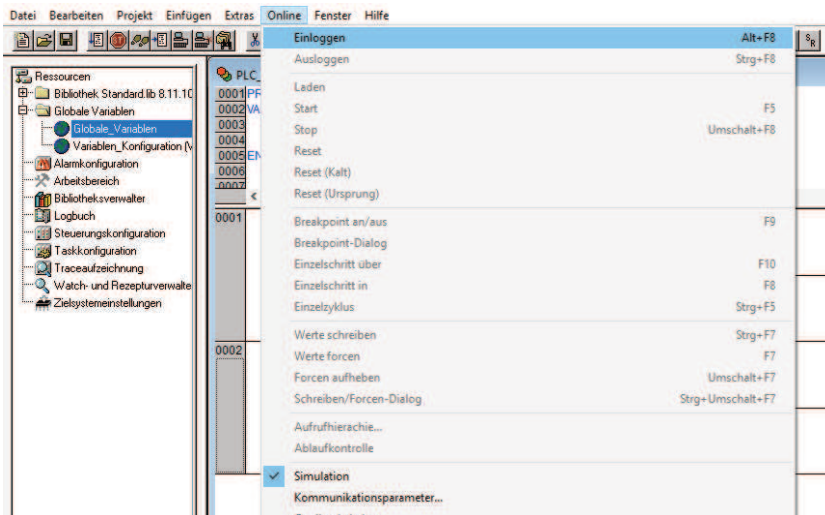
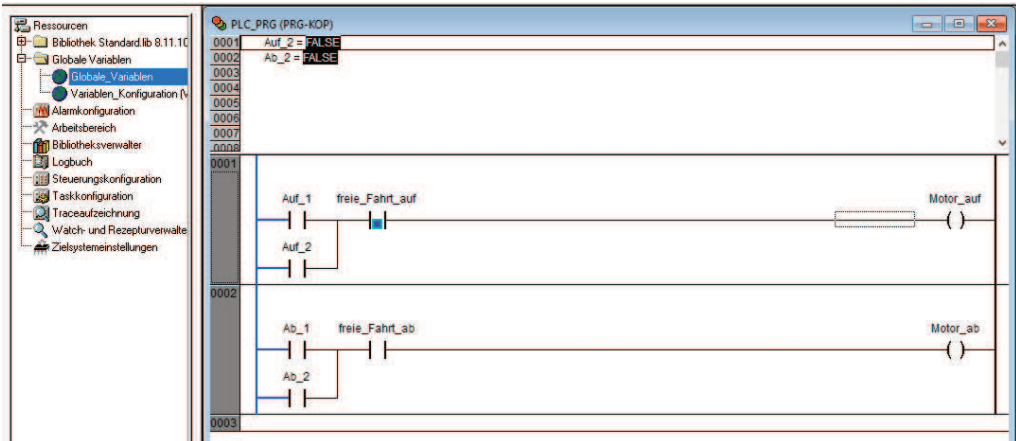


Bild 2-32 Einloggen



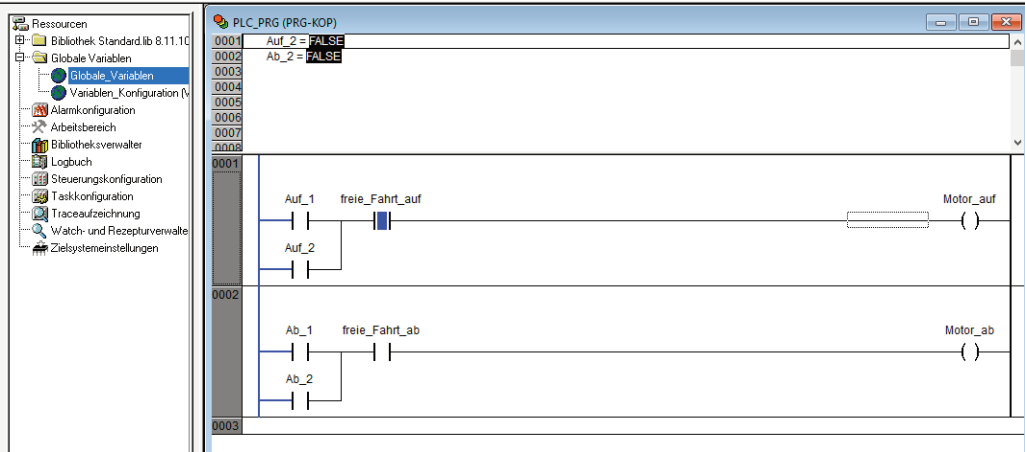
**Bild 2-33** Start der Simulation

Im Kontaktplan sind jetzt die linke Stromschiene und die Linien zu den Spulen blau. Diese Farbe symbolisiert den Zustand logisch 1. Wir können den Zustand der Eingänge von logisch 0 auf logisch 1 ändern, indem wir zuerst mit der linken Maustaste in einen Kontakt klicken, Bild 2-34. Dadurch wird die untere Hälfte des Kontaktes blau markiert. Danach drücken wir gleichzeitig die Tasten **F7** und **Strg** und schreiben damit die logische 1 in die Variable.



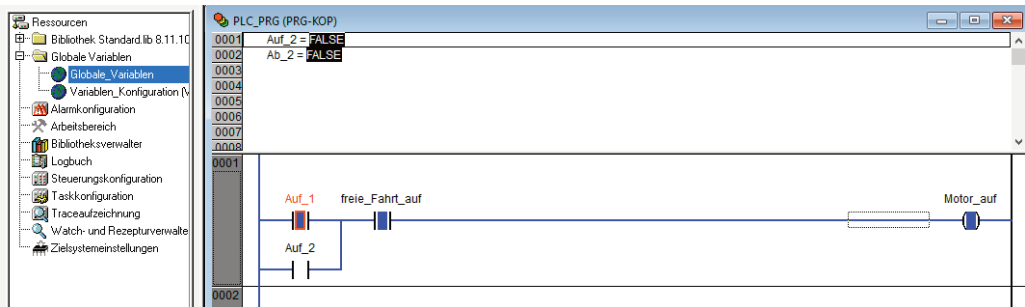
**Bild 2-34** Markieren der Kontakte mit der linken Maustaste

Der geschriebene Wert bleibt solange erhalten, bis wir ihn entweder von Hand ändern oder die Steuerung einen neuen Wert für diese Variable berechnet.



**Bild 2-35** Schreiben des Wertes mit Tasten **F7** und **Strg**

Wenn wir einen Kontakt markiert haben, können wir statt zu schreiben auch *forcen*. Damit ist gemeint, dass diese Variable dann immer wieder auf den erzwungenen Wert gesetzt wird, auch wenn das Programm zwischenzeitlich einen anderen berechnet. Dieses Forcen geschieht mit der Funktionstaste **F7** und wird durch einen roten Rahmen um das blaue Rechteck symbolisiert.



**Bild 2-36** Forcen des Wertes mit Funktionstaste **F7**

Nachdem wir **Auf\_1** und **freie\_Fahrt\_auf** auf logisch 1 gesetzt haben, berechnet der Simulator für unsere Variable **Motor\_auf** ebenfalls den Wert logisch 1 und markiert die Linie blau und die Spule entsprechend mit einem blauen Rechteck.

Mit **online** / **Ausloggen** verlassen wir den Simulator. Spätestens jetzt sollte man das Projekt unter **Datei** / **Speichern** auf der Festplatte sichern.

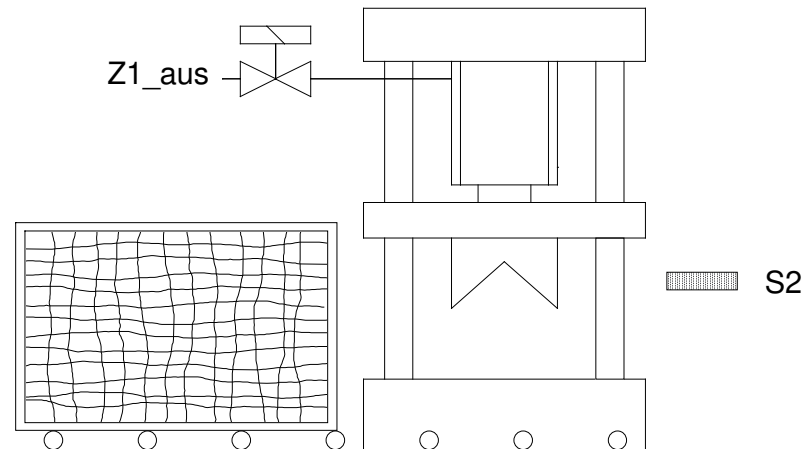


## 2.4 Aufgabe zu Kapitel 2

### Aufgabe 2 – 1

Eine Stanze soll das Werkstück bearbeiten, wenn sowohl der Startschalter **S3** betätigt ist als auch der Schutzgitterkontakt **S2** ein geschlossenes Schutzgitter meldet.

Bild 2-17 zeigt den Aufbau, wobei der Startschalter **S3** nicht dargestellt ist. Die Stanze bewegt sich nach unten, wenn das Ventilsignal **Z1\_aus** logisch 1 ist.



**Bild 2-37** Aufbau der Stanze

### Aufgaben

- Erstellen Sie eine Belegungsliste.
- Zeichnen Sie den Kontaktplan.

### Lösung

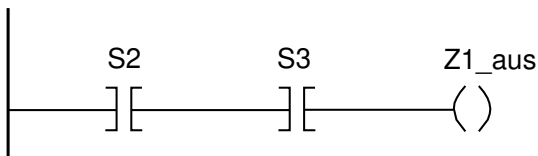
- Die Belegungsliste enthält Tabelle 2.5. Diese Liste zeigt an, welche Komponenten verwendet werden und enthält später auch die Klemmenbezeichnungen an der Spei-

cherprogrammierbaren Steuerung. Wichtig ist, dass wir in dieser Liste eindeutig festlegen, wann das Signal bei einem Sensor logisch 1 ist, bzw. wie ein Antrieb reagiert, wenn das Eingangssignal logisch 0 oder logisch 1 ist.

**Tabelle 2.5** Deklaration der Variablen für die Steuerung der Stanze

Komponente	Variable	Kommentar
Schutzgitterkontakt	<b>s2</b>	Schutzgitter geschlossen => <b>s2</b> = 1, sonst <b>s2</b> = 0
Startschalter	<b>s3</b>	Taster gedrückt => <b>s3</b> = 1, sonst <b>s3</b> = 0
Ventil	<b>z1_aus</b>	<b>z1_aus</b> = 1 => Kolbenstange fährt aus, sonst Stillstand

## b) Kontaktplan der Steuerung



**Bild 2-38** Kontaktplan der Steuerung

# Literatur

- Adam H-J, Adam M (2000) Programmieren in Anweisungsliste nach IEC 1131-3. Eine systematische und handlungsorientierte Einführung in die strukturierte Programmierung. 2. Aufl. Elektor-Verl.
- anon. (2007) Handbuch für SPS Programmierung mit CoDeSys 2.3. 3S - Smart Software Solutions GmbH. Kempten
- anon. (2008) GRAFCET. Struktur, Darstellung und Anwendung. Festo Didactic GmbH & Co. KG - Bildungsverlag Elns Troisdorf
- Antonsen T M, (2020) SPS Programmierung mit Strukturierter Text (ST), V3: IEC 61131-3 und bewährte Praktiken der ST-Programmierung. Books on Demand
- Aspern J von (2020) SPS Grundlagen: Aufbau, Programmierung (IEC 61131, S7), Simulation, Internet, Sicherheit. 3.Aufl. VDE Verlag
- Auer A (1997) SPS-Praktikum der IEC1131-Syntax. Electronic Media, Detmold
- Beater P (2019) Lehr- und Übungsbuch zur Regelungstechnik: Eine beispielorientierte Einführung in die Theorie mit vielen gelösten Aufgaben. Books on Demand
- Beater P (2019) Aufgabensammlung zur Steuerungstechnik: 56 mit Papier und Bleistift oder CoDeSys gelöste Aufgaben. Books on Demand
- Brouër B (1995) Steuerungstechnik für Maschinenbauer. Teubner
- Fasol K H (1988) Binäre Steuerungstechnik. Springer
- Hanssen D H (2015) Programmable Logic Controllers: A Practical Approach to IEC 61131-3 Using Codesys. John Wiley & Sons
- John K J, Tiegelkamp M (2009) SPS-Programmierung mit IEC 61131-3. Konzepte und Programmiersprachen, Anforderungen an Programmiersysteme, Entscheidungshilfen. 4. Aufl. Springer
- Lepers H (2009) SPS-Programmierung nach IEC 61131-3. Mit Beispielen für CoDeSys und Step 7. 3. Aufl. Franzis
- Lewis R W (1998) Programming industrial control systems using IEC 1131-3. Institution of Electrical Engineers, London
- Montenegro S (1999) Sichere und fehlertolerante Steuerungen. Hanser
- Neumann P (1998) SPS-Standard IEC 1131. Programmierung in verteilten Automatisierungssystemen. 2. Aufl. 1998
- Oestereich B (1998) Objektorientierte Softwareentwicklung. R. Oldenbourg

- Olsson G, Piani G (1993) Steuern, Regeln, Automatisieren. Hanser
- Petry J (1996) Modicon Micro. Programmierung mit ConCept. AEG Schneider Automation
- Petry J (2007) SPS Programmierung nach IEC 61131-3 mit Multiprog 4.0. IBP Ingenieurbüro Petry
- Petry J (2014) SPS-Programmierung mit CODESYS V2.3: Praxisorientiert - Realitätsnah - Erprobt! REINHOLZ Software & Technology
- Plagemann B (2008) Crashkurs GRAFCET. Dr.-Ing. Paul Christiani GmbH & Co. KG
- Pusch K (1999) Grundkurs IEC 11 31. Vogel
- Reuter H, Machalek K (2000) SPS-Programmierung nach IEC 61131. Verl. Europa-Lehrmittel
- Rolle I, Lehmann A (1998) IEC 61131 – wozu? SPS: Programmiersprachen, Kommunikation, Fuzzy control. VDE-Verlag
- Schmitt K (2019) SPS-Programmierung mit ST: nach IEC 61131 mit CoDeSys und mit Hinweisen zu STEP 7 im TIA-Portal. Vogel Business Media
- Tröster F (2015) Regelungs- und Steuerungstechnik für Ingenieure: Band 2: Steuerungstechnik. De Gruyter Studium
- Wellenreuther G, Zastrow D (2015) Automatisieren mit SPS - Theorie und Praxis. 6. Aufl. Springer
- Wellers H (1996) SPS-Programmierung nach IEC 1131-3. Cornelsen
- Wratil P (1996) Moderne Programmiertechnik für Automatisierungssysteme. EN 61131 (IEC 1131) verstehen und anwenden. Vogel

# Stichwortverzeichnis

- 3/2-Wegeventil 14
- 5/2-Wegeventil 14
- 5/3-Wegeventil 14
  
- Abarbeitungsreihenfolge 88, 121, 134
- Ablauf 81 ff
- Ablaufauswahl 93
- Ablaufsprache AS 99 ff
- Ablaufsteuerung 81 ff
  - zeitgeführt 81
  - prozessabhängig 82
- Ablaufzusammenführung 93
- Absorptionsgesetz 53
- AE s. Aktuelles Ergebnis
- Aktion 83 ff
  - programmieren 101
- Aktualparameter 138
- Aktuelles Ergebnis in AWL 130
- Alternativbetrieb in AS 93
- Anfangsschritt 85 ff
- Anweisung
  - AWL 129
  - ST 153
- Anweisungsliste AWL 129–151
- Anweisungsteil 170, 177
- Array 159, 188
- AS s. Ablaufsprache
- Assoziativgesetz 51
- Aufruf von Funktionen
  - in AWL 137
  - in ST 162
- Aufruf von Funktionsbausteinen
  - AWL 138
  - ST 162
  
- Ausgangsbaugruppe 9, 119–120
- Auswahl 92 ff
- Automatikbetrieb 123
- Ausdruck in ST 153
- Ausschaltverzögerung 68
  - Funktionsbaustein TOF 197–199
- AWL s. Anweisungsliste
  
- Bauaufzug (Beispiel)
  - CODESYS 34
  - AWL 141
  - FBS 65
  - KOP 10, 67
  - ST 165
- bedingter Sprung 134, 136
- Belegungsliste 11
- Bestimmungszeichen 87–92
- Betriebsarten 123, 124
- Bewegungsdiagramm 95
- Boole, George 48
- Boolesche Gleichung 47
  
- CASE-Anweisung in ST 157, 167
- CFC Editor CODESYS 71 ff
- CPU-Baugruppe 119
- CTU 200–204
- CTD 200–204
- CTUD 200–204
  
- Datentyp 186 ff
- Dauerzyklus 123
- Deklaration
  - Funktionsbaustein 180
  - globale Variablen 18 ff, 177, 194

lokale Variablen 20 ff, 190  
Deklarationsteil 20, 70, 177  
De Morganscher Satz 53  
direkt dargestellte Variable 177, 183, 186  
Distributivgesetz 52, 55, 57  
doppeltes Komplement 49  
Drahtbruchsicherheit 12, 126

Eingangsbaugruppe 9, 119  
Einschaltverzögerung 68, 197 ff  
Einschaltverzögerung TON 69, 70, 197 ff  
Einzelschrittbetrieb 123  
Einzelzyklus 123  
EXIT-Anweisung in ST 160  
Exklusiv-ODER-Verknüpfung 32

FB s. Funktion  
FBS s. Funktionsbaustein-Sprache  
Feld 188  
Flanke 192, 203  
Flankenwechsel 78, 200 ff  
FOR-Anweisung in ST 158  
Forcen in CODESYS 24  
Formalparameter 138  
Flipflop 61 ff  
FUN s. Funktion  
Funktion (POE) 173 ff  
Funktionsaufruf  
in AWL 137  
in ST 161

Funktionsbaustein 28  
CTD 200 ff  
CTU 200 ff  
CTUD 200 ff  
RS 70  
SR 70  
TOF 197–199  
TON 70, 197–199  
TP 197–199

Funktionsbausteinaufruf  
AWL 138  
ST 162

Funktionsbaustein-Sprache 28

Funktionsbaustein-Plan 28  
Funktionsdiagramm 81, 95  
Funktionsplan 81–95  
Funktionstabelle 28  
FUP Editor CODESYS 34 ff

globale Variablen 177, 187, 194  
CODESYS 74  
Grundstellung 87  
Grundverknüpfung  
Exklusiv-ODER 32  
NAND 31  
Negation 30  
NOR 30  
ODER 29  
UND 27

IF-Anweisung in ST 155, 156  
Instanz 181  
Inversionsgesetze 53

Klammer ()  
AWL als Modifizierer 131  
ST 163  
klassische SPS 189  
kombinatorische Operation 31, 32  
kombinatorische Schaltungen 47  
Kommentar  
AWL 129  
ST 164  
Kommutativgesetz 51  
Kompakt-SPS 119  
Komplementgesetz 49  
Konstante s. a. VAR\_CONSTANT 70, 189  
Kontakt 5, 6  
negiert 6, 7  
Kontaktplan 6 ff  
KOP s. Kontaktplan

Label s. Sprungmarke  
Logikplan 28  
lokale Variablen 177, 194  
CODESYS 20

- Marke 129, 134
- Merker 120, 133
- Modifizierer 130, 131
- Multiauswahl CASE 157
  
- NAND-Verkpfung 31
- Negation 6, 30
- negierter Kontakt 7, 9
- Netzwerk 12, 19
- NOR-Verkpfung 30
- NOT-AUS 124–126
  
- ODER-Verknpfung 29
- ffner 6
- Operand in AWL 129 ff
- Operatoren
  - AWL 129 ff
  - ST 164 ff
  
- Parallelbetrieb 92–95
- POE s. Programm-Organisationseinheit
- PRG s. Programm
- Prioritt s. a. Rangfolge
  - AWL-Operatoren 135
  - Boolesche Verknpfungen 50
  - ST-Operatoren 154, 164, 174
  - Task 189
- Programm 183
- Programm-Organisationseinheit 176 ff.
  - Funktion 177
  - Funktionsbaustein 180
  - Programm 183
- Prozessabbild der Ausgnge 121–122
- Prozessabbild der Eingnge 121–122
  
- Rangfolge
  - in AWL 135
  - in ST 164
- REPEAT-Anweisung in ST 160
- Richtbetrieb 124
- RS-Flipflop 61 ff, 70 ff
- Rcksetzdominanz 63
- Rcksetzspule in KOP 64
- Rumpf 177
  
- Schaltfolge-Diagramm 62
- Schaltalgebra 48
- Schaltfunktion 47
- Schaltgleichung 47
- Schaltlogik 48
- Schalttabelle 28
- Schaltwerk 61
- Schleifen
  - AS 95
  - AWL 134
- Schritt 81, 83 ff
- sequentielle Steuerung 81
- Setzdominanz 63
- Setzspule in KOP 64
- simultan 184, 189 ff
- Speicherglied 61 ff
- Soft-SPS 118
- Sprungoperator in AWL 134
- Sprungmarke 134, 140
- SPS
  - klassische 119
  - Kompakt-SPS 119
  - Soft-SPS 118
- Spule in KOP 7
  - mit Gedchtnis 64
  - Rcksetzspule 64
  - Setzspule 64
- ST s. Strukturierter Text
- Steuerdiagramm 95
- Steuerung 1 ff.
  - binre 3
  - kontinuierliche 3
- Stromlaufplan 5–8
- Stromschiene in KOP 8
- Struktur (STRUC) 184
- Strukturierter Text 153 ff
- symbolische Variablen 179
  
- Task 189–192
- Tautologiesgesetz 49
- TOF 197–199
- TON 197–199
- TP 197–199

- Transition 84 ff
- Transitionsbedingung 85 ff
- Übergangsbedingung 82, 83
- UND-Verknüpfung 27, 28
- VAR
  - VAR\_EXTERNAL 181, 189, 194
  - VAR\_IN\_OUT 181, 189
  - VAR\_INPUT 177, 181, 189
  - VAR\_OUTPUT 181, 189
- Variablen 187, 189
  - direkt dargestellt 177, 183, 186
  - global 177, 187, 194
  - symbolisch 183
- Vergleichsfunktion 178,
- Verknüpfungssteuerung 47 ff
- Verriegelung 13, 126
- Verteilungsgesetz 52
- Verzögerungsglied (binäres) 68 ff
- Wahrheitstabelle 28
- Weg-Schritt-Diagramm 96 ff, 104 ff
- Weg-Zeit-Diagramm 96 ff
- WHILE-Schleife in ST 159
- Wiederanlauf 92, 120
- Wirkungslinie in AS 85
- Zeitgeber 68
- Zeitglied 61 ff, 197
- Zähler 200ff
- Zuweisung 153, 162, 152
- Zyklus 92, 93
- Zykluszeit 122, 192