Steuerungstechnik

6. Entwurf von Steuerungsprogrammen

OBERÖSTERREICH

V4.1

Prof. (FH) DI Dr. Franz Auinger

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

OBERÖSTERREICH

Entwurfstechniken - Statecharts

HAGENBERG | LINZ | STEYR | WELS

Ein kurzer Ausflug in den Hausbau

- Zeit in die Planung das Hauses zu investieren ist sehr wichtig!
- Nehmen Sie sich in der Konzeptphase genug Zeit!
- Strukturiertes Angehen des Entwurfes (Technik der schrittweisen Verfeinerung des Entwurfs)
  - Bauen Sie Ihr Haus auf Ihr Grundstück
  - Legen Sie die Untereinheiten (Raumeinteilung) fest
  - Kleinere Teile sind leichter zu handhaben
  - Legen Sie die Schnittstellen fest (Türen und Fenster)
  - ….
  - Die Planung ist abgeschlossen wenn alles deutlich und einfach wirkt.
- Achtung: Fangen Sie nicht zu **bauen** an bevor die Planung abgeschlossen ist!

Entwurf von Steuerungsprogrammen

- Modellierung des zu automatisierenden Prozesses
  - **Gliederung des Prozesses in einzelne Teilaufgaben (Prozesse)**
    - Prozessübersicht, Blockschaltbild, etc.
  - **Beschreibung aller Teilaufgaben**
    - Führungsgrößen, Stellgrößen, Rückführungsgrößen, usw
    - Schnittstellenspezifikation als Zuordnungsliste
  - Entwurf und Beschreibung der Sicherheitsanforderungen
    - Risiko- und Gefahrenanalyse nach EN 12100 sind bereits während des Entwurfs Pflicht (!)
  - Definition und Beschreibung der Bedien- und Anzeigeeinrichtungen
    - HMI Spezifikation
    - Graphischer HMI Prototyp
  - Konfiguration / Zusammenstellung der Automatisierungsgeräte
    - Anzahl und Art der Ein- Ausgangsbaugruppen, Kommunikationsschnittstellen etc.

Entwurf von Steuerungsprogrammen

- Dokumentation

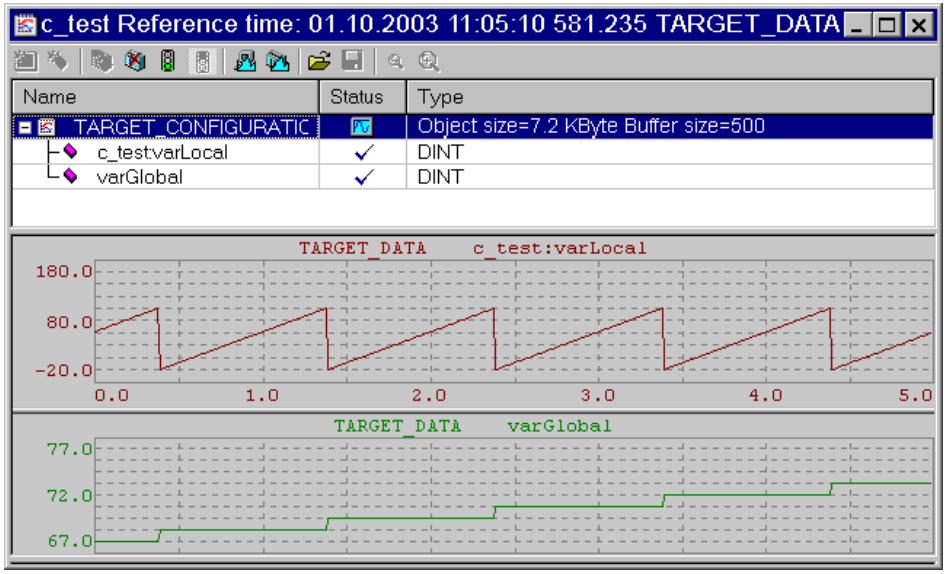
Dokument	Ersteller	Inhalt
Lastenheft	Auftraggeber	Beschreibung der Ziele / Funktionen Was soll erreicht werden
Anforderungsliste	Auftraggeber und Ausführende	Präzise Formulierung der Eigenschaften der zu erreichenden Ziele in Form einer Checkliste Basis für den Abnahmetest
Pflichtenheft	Ausführende	Wie wird das Ziel erreicht Beschreibung der Eigenschaften, Methoden und Verfahren
Spezifikation	Ausführende	Zerlegen in Module - Prozesstechnik / Mechanik - Elektrotechnik (Steuerstrecke, Aktuatoren) - Steuerungstechnik (Steuerungseinrichtung) Beschreiben der Module - Steuerungstechnik: - Auswahl der Methode - Erstellen der funktionalen Spezifikation
Umsetzung	Ausführende	Steuerungstechnik: - Programmieren der Module - Integration in die Maschine / Anlage - Test der Module - Test des Gesamtsystems
Abnahme	Auftraggeber und Ausführende	Allgemein: - Test nach Anforderungsliste - Funktion - Erfüllungsgrad Steuerungstechnik: - Test des Gesamtsystems

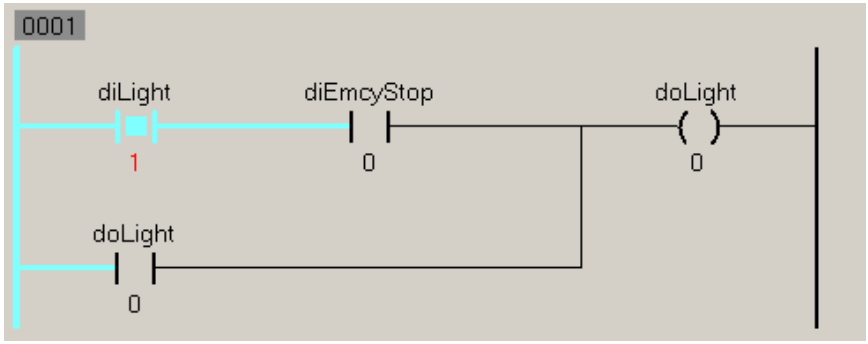
Entwurf von Steuerungsprogrammen

- Einfaches Rezept (auch für Labor bzw. Hausübungen)
  1. Prinzipschaubild zeichnen
  2. Aktoren/Sensoren einzeichnen
  3. Aufteilung in Teilprozesse, Bildung von Modulen / Klassen
  4. Zuordnungsliste global und für jeden Teilprozess erstellen
  5. Variable gemäß Zuordnungsliste definieren
     (Ggf. Variable in eigenen Structs prozessspezifisch gruppieren)
  6. Programme / FBs für jeden Teilprozess anlegen, implementieren und einzeln(!) testen
  7. Implementieren Sie für jeden FB ein passendes **Testprogramm**!
  8. Hauptprogramm zur Verkettung der Teilprozesse anlegen, implementieren und testen

Testen

- Testen ist ein entscheidender Punkt für Softwarequalität
- Definieren Sie Testfälle auf Basis der Spezifikation
- Testen Sie in Projektgruppen gegenseitig!
- Achten Sie auf Spezialfälle
- Verwenden Sie die Diagnosewerkzeuge von Automation Studio
  - Force
  - Watch
  - Trace
  - Debug





Kodierrichtlinien

- Warum braucht man Kodierrichtlinien?
  - Sie helfen beim Verbessern der Softwarequalität
  - Standardisierung
- Bevor Sie programmieren:
  - Kontrollieren Sie Softwarearchitektur und Design!
  - Fangen Sie nicht zu schnell an zu programmieren!
- Namenskonventionen
  - Begründung: Gute Variablen- und Datentypnamen sind der Grundstein für die Lesbarkeit der Programme.
  - Sprache, Anwenderdatentypen, Konstanten, lokal, global, C-local und I/O Variablen, Zeiger, Funktionsblockinstanzen

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

OBERÖSTERREICH

Logikbasierte Modellierung

HAGENBERG | LINZ | STEYR | WELS

Logik-Arten

- Kombinatorische Logik
  - Resultat: Verknüpfungssteuerung
  - Modellierung
    - Wahrheitstabelle
    - Bool‘sche Logik / KV Diagramm
- Sequentielle Logik
  - Resultat: Ablaufsteuerung
  - Modellierung
    - Flussdiagramm (nicht empfohlen!)
    - **Zustandsautomat (eigenes Kapitel)**
- Realität
  - Mischung aus beidem!!!
  - Softwarearchitektur notwendig

Modellierungstechniken
Auswahl

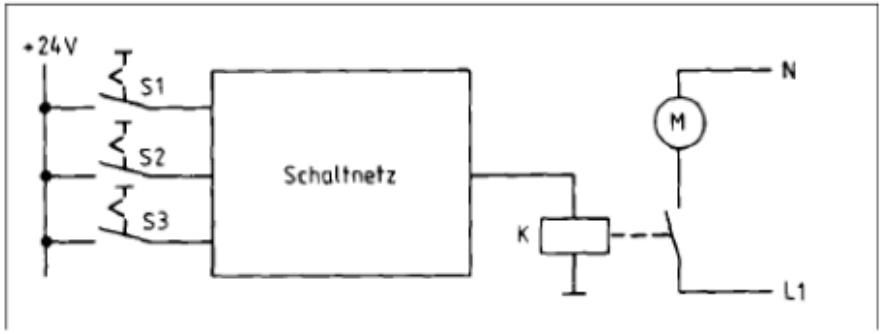
- Generelle Frage:
  - Hat die Maschine einen Zustand
    - Codeteile welche nicht immer ausgeführt werden
    - Codeteile welche auf Basis der zeitlichen Vorgeschichte des Betriebs ausgeführt werden
  - Wenn ja
    - Kombinatorische Lösung möglich ?
      - Ist Vorgeschichte kombinatorisch determinierbar
      - Können Zeitglieder zur Anwendung kommen?
      - **Wenn ja dann Programmierung direkt mit boolschen Verknüpfungen / Wertetabelle**
    - Keine kombinatorische Lösung sinnvoll möglich
      - **Modellierung/Programmierung als Zustandsautomat / Ablaufsteuerung**

Modellierungstechniken

- Kombinatorische Logik
  - Wertetabelle / RS Tabelle
  - Weg/Zeitdiagramm
  - Stromlaufpläne (eher bei bestehenden Anlagen)
- Sequentielle Logik
  - Weg/Zeitdiagramm
  - R&I-Fließschema
  - Zustandsbasierte Modellierung

Modellierungstechniken
Wertetabelle

- Verknüpfungssteuerung ohne Speicher
- Funktionen mit KV-Diagramm vereinfachen
  - Praxistipp: Vermeiden Sie zu starke Vereinfachungen
  - WARUM?
    - Führt i.d.R. zu einer schlechteren Nachvollziehbarkeit beim Fehlersuchen
    - Bei modernen Steuerung ist Anzahl an Logikverknüpfungen kein Leistungskriterium!
- Oft bei Verriegelungen / Sicherheitsfunktionen



S3	S2	S1	K1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Modellierungstechniken
RS Wertetabelle

- Verknüpfungssteuerung mit Speicher
  - Speicher sind Objekte in der Anlage welche einen Zustand haben
    - z.B. Behälter (LEER, VOLL, HALBVOLL), Motor (EIN, AUS)
  - Aufstellen einer tabellarische Übersicht der vorzusehenden Setz- und Rücksetzbedingungen für Speichervariable
- Anwendung einer RS-Tabelle erfolgt in zwei Schritten:
  - Ermittlung der Anzahl notwendiger RS-Speicherglieder auf Grund der unterscheidbaren Steuerungszustände, eventuell gegliedert in Speicher M und Hilfsspeicher HS.
    - Tipp: Speicherelemente sind i.d.R Aktoren welche ein- bzw. ausgeschaltet werden müssen (Motore, Ventile, …)
  - Eintragen der Setzbedingungen und Rücksetzbedingungen für alle Speicher M und Hilfsspeicher HS.

Modellierungstechniken
RS Wertetabelle

- **Beispiel: Anlage mit 3 Behältern zum Füllen**

Zuordnungstabelle der Eingänge und Ausgänge:

Eingangsvariable	Symbol	Datentyp	Logische Zuordnung		Adresse
Vollmeldung Behälter 1	S1	BOOL	Behälter 1 voll	S1 = 1	E 0.1
Vollmeldung Behälter 2	S3	BOOL	Behälter 2 voll	S3 = 1	E 0.3
Vollmeldung Behälter 3	S5	BOOL	Behälter 3 voll	S5 = 1	E 0.5
Leermeldung Behälter 1	S2	BOOL	Behälter 1 leer	S2 = 1	E 0.2
Leermeldung Behälter 2	S4	BOOL	Behälter 2 leer	S4 = 1	E 0.4
Leermeldung Behälter 3	S6	BOOL	Behälter 3 leer	S6 = 1	E 0.6
Ausgangsvariable					
Magnetventil Behälter 1	Y1	BOOL	Ventil offen	Y1 = 1	A 4.1
Magnetventil Behälter 2	Y2	BOOL	Ventil offen	Y2 = 1	A 4.2
Magnetventil Behälter 3	Y3	BOOL	Ventil offen	Y3 = 1	A 4.3

- Ermittlung des Speicherbedarfs:

  Drei Speicherglieder Y1, Y2, Y3 für die Behälter

  Drei Hilfsspeicher HS1, HS2, HS3 für die Leermeldungen

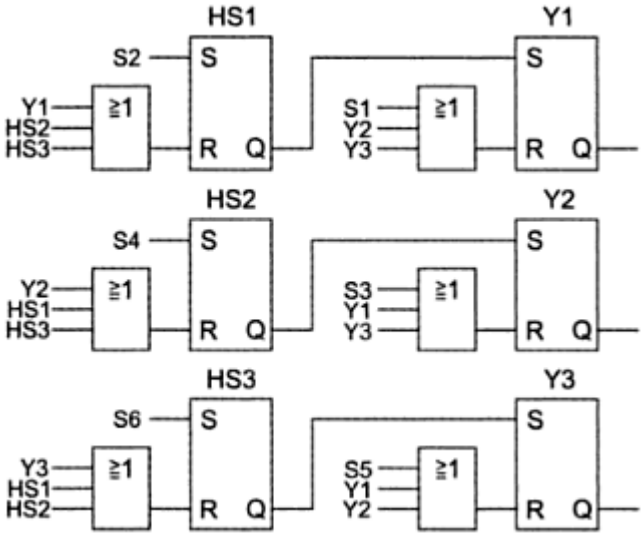
Modellierungstechniken
RS Wertetabelle

- **Eintragen der Setzbedingungen und Rücksetzbedingungen für alle Speicher M und Hilfsspeicher HS.**

Zu betätigende Speicherglieder	Bedingungen für das Setzen	Bedingungen für das Rücksetzen
HS1 für Behälter 1	S2	Y1 v HS2 v HS3
Y1 für Behälter 1	HS1	S1 v Y2 v Y3
HS2 für Behälter 2	S4	Y2 v HS1 v HS3
Y2 für Behälter 2	HS2	S3 v Y1 v Y3
HS3 für Behälter 3	S6	Y3 v HS1 v HS2
Y3 für Behälter 3	HS3	S5 v Y1 v Y2

Modellierungstechniken
RS Wertetabelle

- **Umsetzung in Funktionsbausteinsprache**



Der Entwurf von SPS-Programmen: Funktionsdiagramme nach VDI3260

Siehe auch: http://de.wikipedia.org/wiki/Funktionsdiagramm

Das **Funktionsdiagramm** ist eine grafische Methode, „Funktionsfolgen“ von Maschinen übersichtlich darzustellen.

- Ursprüngliche Beschreibung: VDI-Richtlinie VDI 3260 vom Juli 1977
- 1992 durch den Funktionsplan der DIN 40719-6 ersetzt.
- DIN 40719-6 war übergangsweise bis 1. April 2005 gültig
- wurde durch die DIN EN 60848 ersetzt.

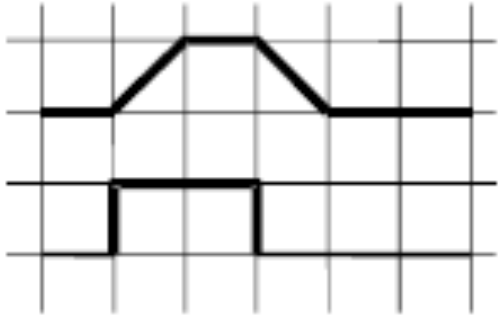
- Verwandt: Weg-Zeit-Diagramm; Weg-Schritt-Diagramm

Der Entwurf von SPS-Programmen: Funktionsdiagramme nach VDI3260

Der schrittweise Ablauf von aufeinanderfolgenden Prozeßzuständen läßt sich in sogenannten Schaltfolgediagrammen darstellen.

Das Funktionsdiagramm nach VDI 3260 erlaubt nicht nur die übersichtliche Darstellung aller Bauglieder einer Steuerkette, sondern liefert auch Informationen über Aufgaben, Funktion und Zusammenspiel der Elemente.

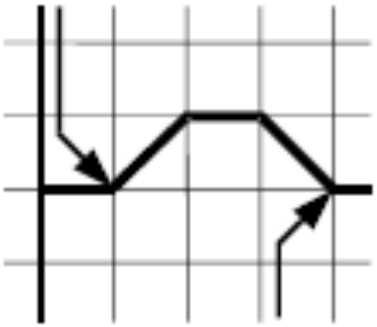
Symbole des Funktionsdiagramms:

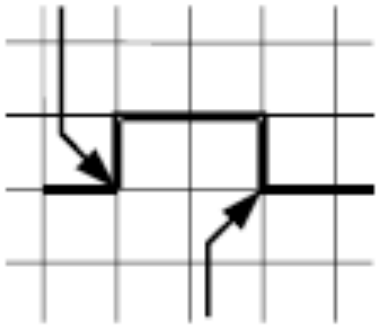


Eine dünne Funktionslinie kennzeichnet die Ruhestellung eines Baugliedes, eine dicke Funktionslinie die davon abweichende Stellung.

Mit einer Signallinie wird die Abhängigkeit der Antriebsglieder

von den Stell- und Signalgliedern dargestellt. Ein Pfeil kennzeichnet die Wirkungsrichtung.



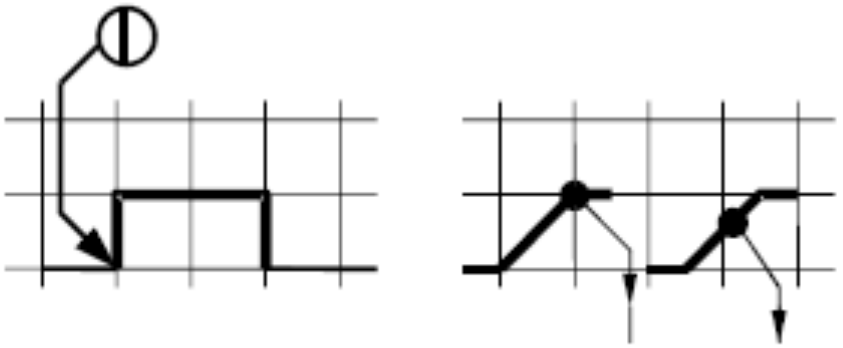


Der Entwurf von SPS-Programmen

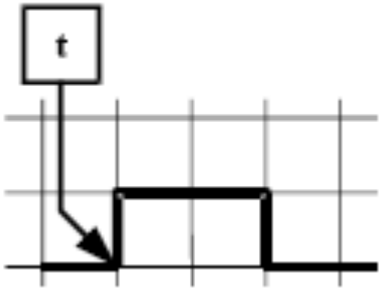
Funktionsdiagramme werden hauptsächlich in pneumatischen Anlagen eingesetzt.

Muskelkraftbetätigte Signalglieder werden durch Symbole in einem Kreis dargestellt.

Mechanisch betätigte Signalglieder werden durch einen dicken Punkt dargestellt.



Druckabhängige oder zeitabhängige Signalglieder werden mit einem speziellem Symbol in einem Quadrat dargestellt.



Die Darstellung einer Zustandsänderung bei den Steuer- und Stellgliedern erfolgt auf der Schrittlinie, senkrecht verlaufend. Die Abszissen werden mit den Buchstaben der jeweiligen Schaltstellungsbezeichnung (a, b) der Wegeventile gekennzeichnet. Bei Ventilen mit 3 Schaltstellungen werden 3 Zustände aufgetragen, und die Mittelstellung mit 0 gekennzeichnet.

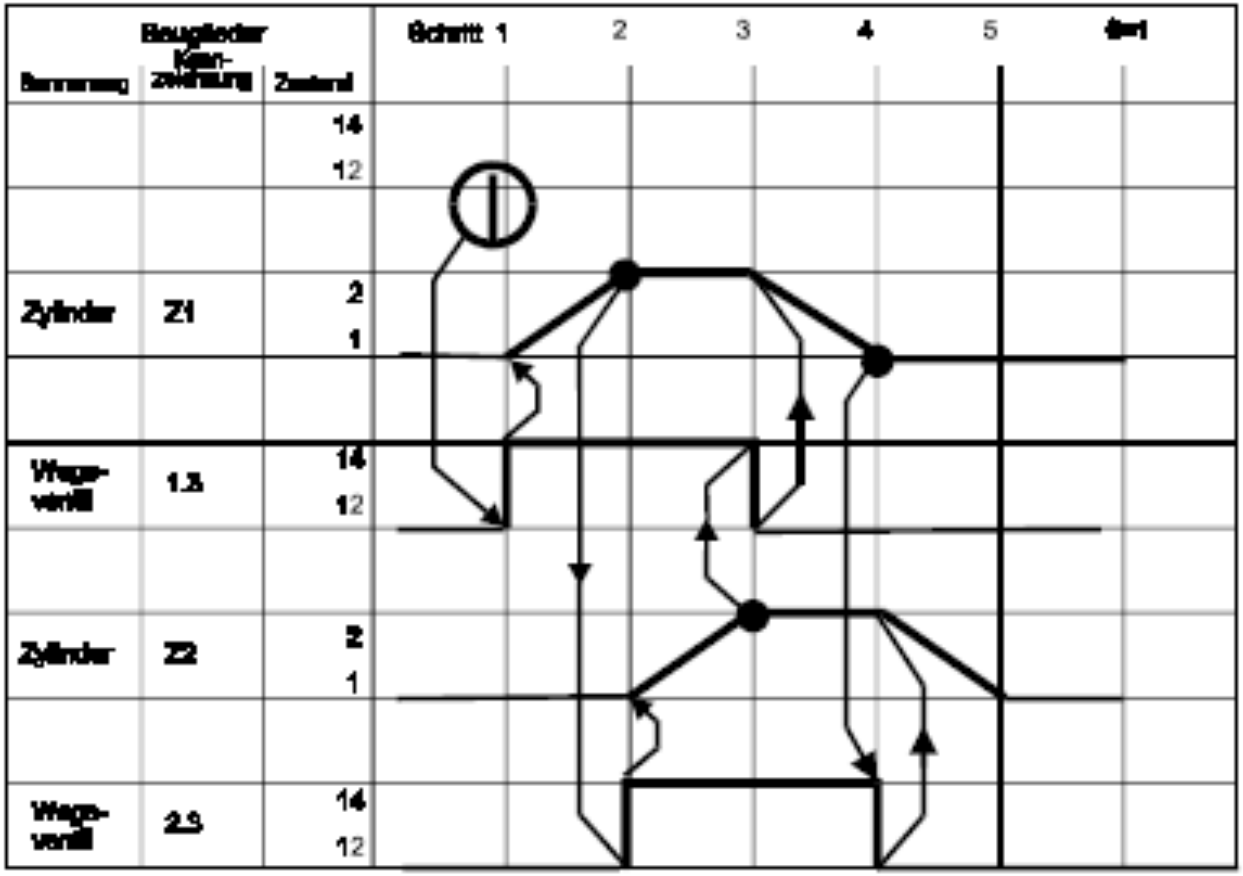
Bei den Antriebsgliedern erfolgt die Darstellung der Bewegung durch eine schräg verlaufende Funktionslinie. Durch unterschiedliche Steigungswinkel der Funktionslinie werden unterschiedliche Geschwindigkeiten dargestellt. Die waagrecht verlaufende Funktionslinie bedeutet einen Stillstand des Antriebsgliedes.

Erstellung eines Funktionsdiagrammes nach VDI 3260

In den Spalten im linken Teil werden Angaben über die Bauglieder wie Benennung, Kennzeichnung und Zustand eingetragen.

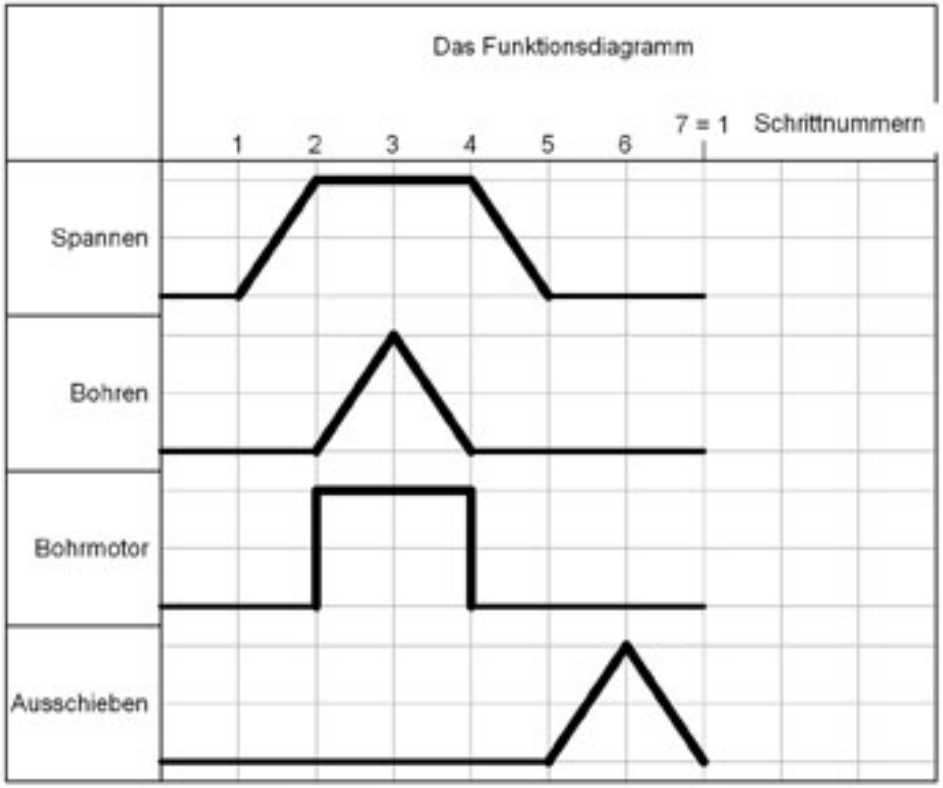
Im Mittelteil des Formblattes werden am Raster die Funktions- und die Signallinien, mit Schritt- oder Zeitaufteilung, eingezeichnet.

Die Reihenfolge der Antriebsglieder entspricht dem Steuerungsablauf, die zugehörigen Stellglieder werden jeweils nach dem Antriebsglied eingezeichnet.



Dann werden die muskelkraftbetätigten Signalglieder mit den Signallinien eingezeichnet. Anschließend wird dann der Steuerungsablauf durch Eintrag der Signalglieder (Grenztaster, druckabhängige Signalglieder und Zeitglieder) und deren Signallinien dargestellt. An der rechten Seite des Diagramms können weitere Informationen und Hinweise eingetragen werden.

Schleifen wie sie nach fertig durchgeführten Automatikabläufen in Autmatisierungsaufgaben üblich sind, werden durch Schrittnummernzuordnungen angedeutet. (z.B: 7=1) Bei Schleifen müssen die Werte der Aktuatoren im End- und Startschritt jeweils gleiche Zustände aufweisen.

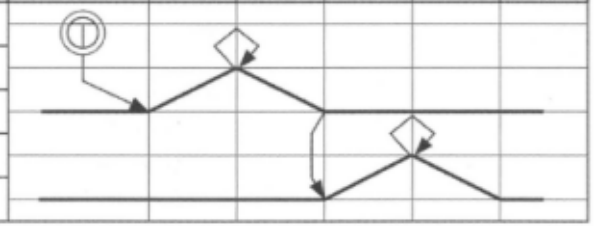


Modellierungstechniken

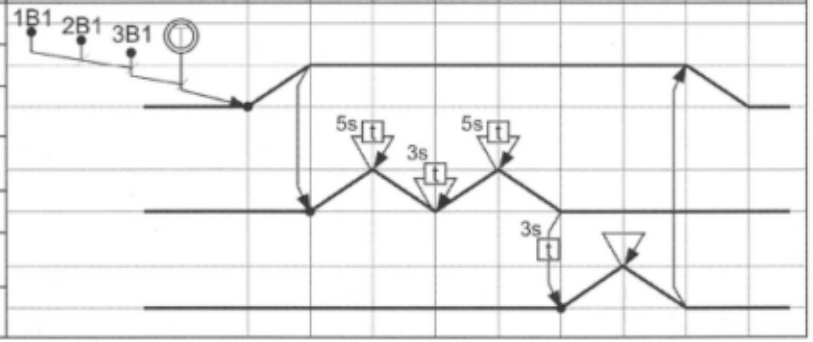
Weg/Zeitdiagramm

- Beispiele

Bauglieder			Zeit
Benennung	Kennz.	Zustand	Schritt 1 2 3 4 5
Taster	1S3	EIN	
DW-Zylinder	1A	ausgefahren	
		eingefahren	
DW-Zylinder	2A	ausgefahren	
		eingefahren	

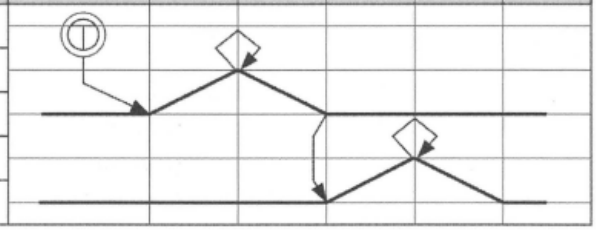


Bauglieder			Zeit
Benennung	Kennz.	Zustand	Schritt 1 2 3 4 5 6 7 8 9
Taster	S1	EIN	
DW-Zylinder	1A	ausgefahren	
		eingefahren	
DW-Zylinder	2A	ausgefahren	
		eingefahren	
DW-Zylinder	3A	ausgefahren	
		eingefahren	

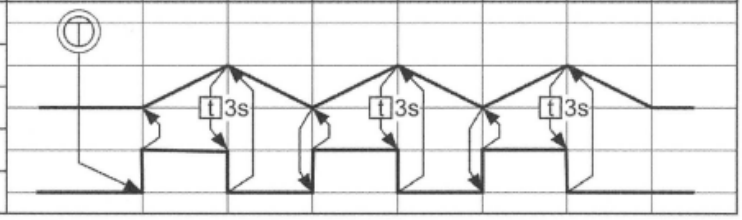


Beispiele zum Funktionsdiagramm

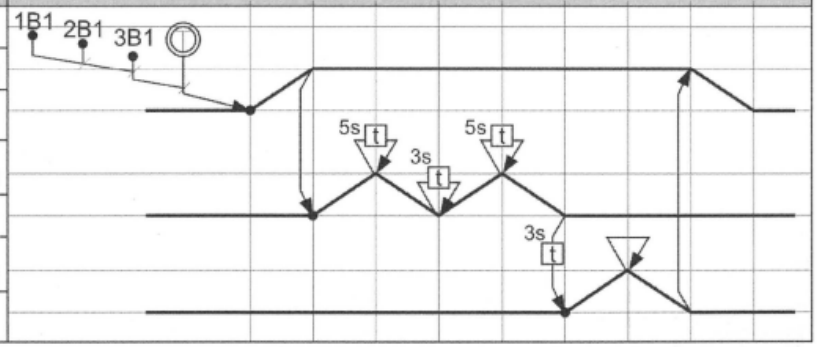
Bauglieder			Zeit
Benennung	Kennz.	Zustand	Schritt 1 2 3 4 5
Taster	1S3	EIN	
DW-Zylinder	1A	ausgefahren	
		eingefahren	
DW-Zylinder	2A	ausgefahren	
		eingefahren	



Bauglieder			Zeit
Benennung	Kennz.	Zustand	Schritt 1 2 3 4 5 6 7
Taster	S1	EIN	
DW-Zylinder	1A	ausgefahren	
		eingefahren	
5/2-Wegeventil	1V1	B	
		A	

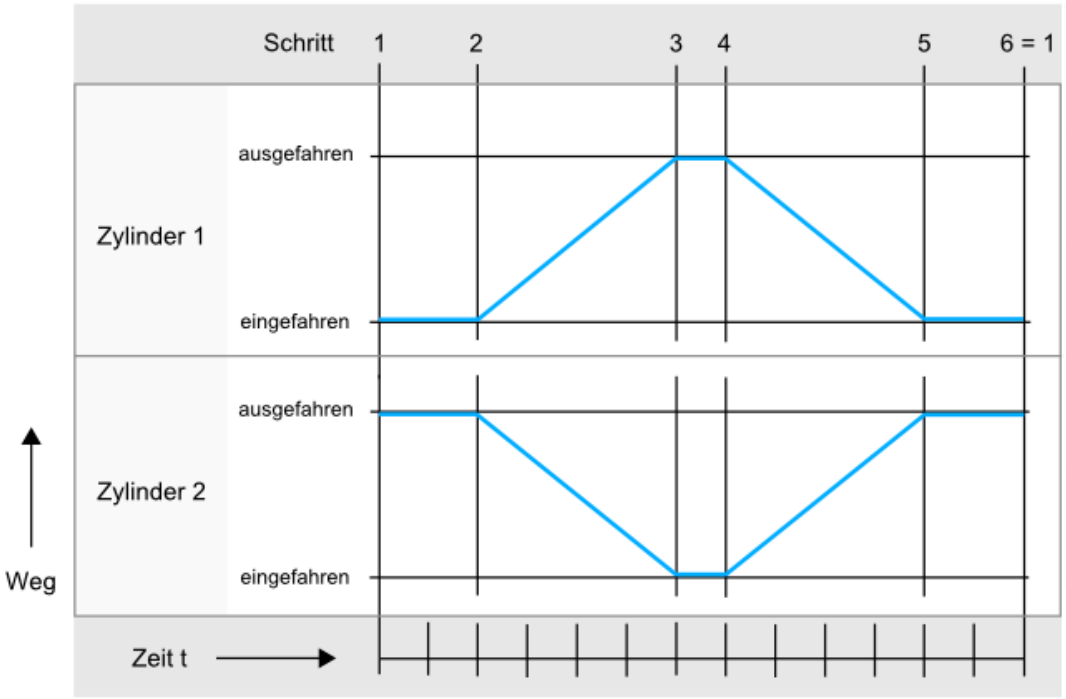


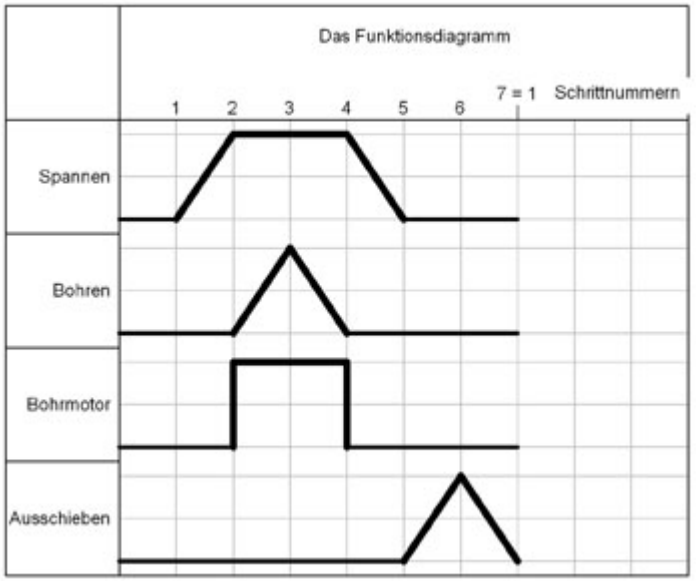
Bauglieder			Zeit
Benennung	Kennz.	Zustand	Schritt 1 2 3 4 5 6 7 8 9
Taster	S1	EIN	
DW-Zylinder	1A	ausgefahren	
		eingefahren	
DW-Zylinder	2A	ausgefahren	
		eingefahren	
DW-Zylinder	3A	ausgefahren	
		eingefahren	



Modellierungstechniken
Weg/Zeitdiagramm

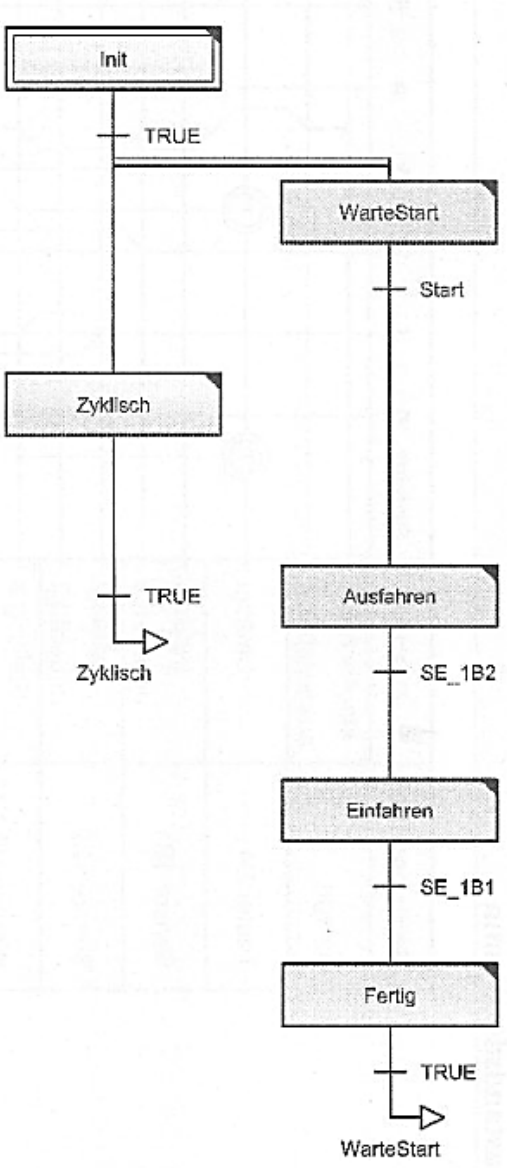
- Für komplexe sequentielle Bewegungsabläufe
- Wird meist für Pneumatiksysteme / Stellzylinder verwendet (z.B. mit FESTO FluidDraw)





Modellierungstechniken
Weg/Zeitdiagramm

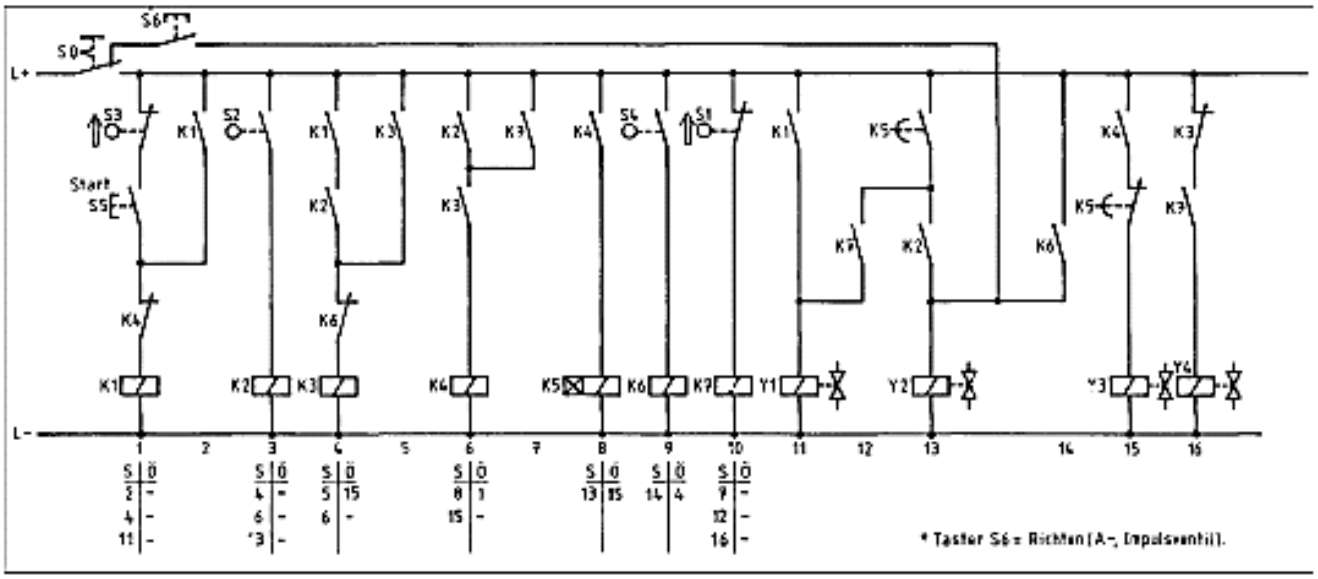
- Umsetzung
  - > Als AS – Schrittkette
  - > Zyklische Programmteile werden in einem Parallelzweig verarbeitet



Modellierungstechniken
Stromlaufpläne

Aus bestehenden Schaltplänen von z.B. Schützensteuerungen läßt sich die Schaltung unmittelbar in ein Steuerungsprogramm umsetzen. Serienschaltungen entsprechen UND-Verknüpfungen, parallele Zweige stellen eine ODER-Verknüpfung dar. Zeitrelais werden durch Zeitglieder mit Ein- oder Ausschalt-verzögerung ersetzt.

Insbesondere wenn bestehende Einrichtungen durch eine SPS ersetzt werden soll, bildet der Stromlaufplan alle Informationen für den Programmcode.



Achtung: Als Entwurfsmethode für neue Projekte ist der Stromlaufplan nicht besonders geeignet, da eine völlig widerspruchsfreie Darstellung der Steuerungsaufgabe verhältnismäßig aufwendig ist.

Regeln für das Umsetzen von Schützschaltungen in SPS-Prog.

Eine gegebene Schützsteuerung, die z. B. zur Ansteuerung von Elektromotoren oder Elektropneumatik eingesetzt wurde, kann durch eine SPS-Steuerung unter sinngemäßer Anwendung der nachfolgenden Umsetzungsregeln für Stromlaufpläne ersetzt werden:

1. Der Hauptstromkreis wird unverändert außerhalb der SPS beibehalten.

2. Hauptschütze werden von den SPS-Ausgängen angesteuert und werden als OUT- oder INOUT -Variable deklariert. Der Schaltzustand des Hauptschützes wird somit außerhalb des Bausteins in einem SPS-Ausgang gespeichert. Wird ein Kontakt Q1 des Hauptschützes Q1am Bausteineingang abgefragt, ist Q1 als Durchgangsvariable (IN_OUT) zu deklarieren.

3. Hilfsschütze werden durch temporäre oder statische Variablen innerhalb des Bausteins ersetzt. Wenn Hilfsschütze interne Speicherfunktionen ausüben, muss für jedes Hilfsschütz eine statische Speichervariable (STAT) deklariert werden.

4. Parallelschaltungen von Schützkontakten werden durch ODER-Verknüpfungen und Reihenschaltungen durch UND-Verknüpfungen der entsprechenden Variablen ersetzt.

5. Öffner von Schützkontakten werden negiert und Schließer bejaht im Programm abgefragt.

Regeln für das Umsetzen von Schützschaltungen in SPS-Prog.

6. Öffner- und Schließerkontakte von Signalgebern wie Taster, Schalter, Kontakte von Überstromschutzeinrichtungen etc. werden im Programm stets bejaht abgefragt und als Eingabevariablen deklariert, wenn derselbe Kontakttyp beibehalten wird.

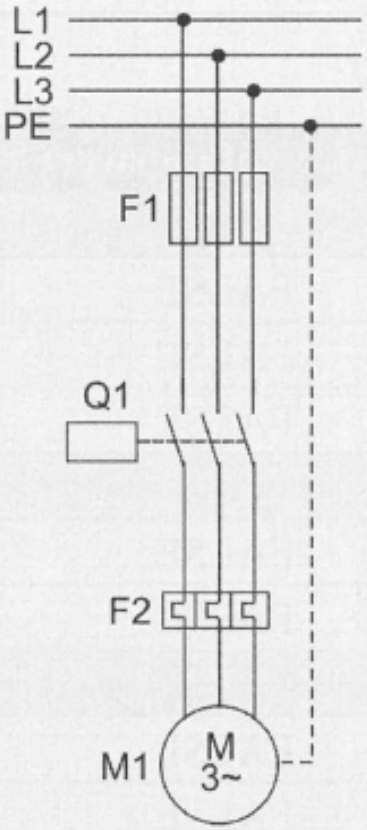
7. Die bejahte Abfrage von Signalgebern in Regel 6 gilt nicht bei Verwendung der Speicherfunktion anstelle der Selbsthaltung.

8. Die Umsetzungsregeln 1 bis 6 verändern nicht die vorgegebene Steuerungsstruktur, wenn die Schütze keine Zeitverzögerungen oder Wischerkontakte enthalten. Einschalt- und Ausschaltverzögerungen müssen mit Zeitgliedern und Wischerkontakten mit Flankenauswertung nachgebildet werden. Impulse von Wischerkontakten beim Einschalten (Ausschalten) entsprechen steigenden (fallenden) Flanken.

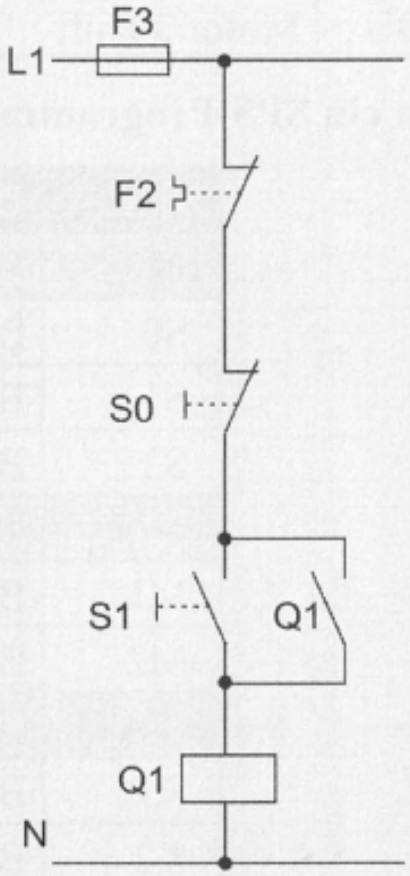
Beispiel zur Umsetzung von Schützschaltungen in SPS-Prog.

Beispiel:

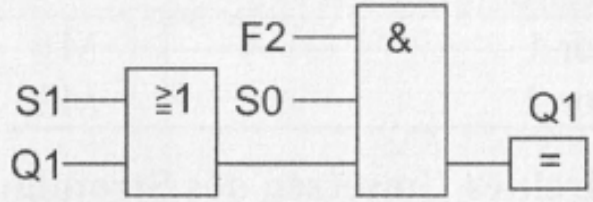
Hauptstromkreis bleibt erhalten



Steuerstromkreis wird ersetzt

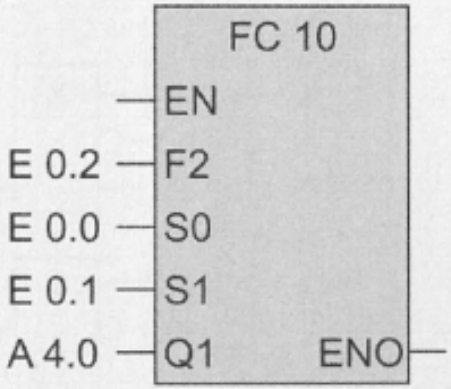


Ersatz-Funktionsplan:



Aufruf im OB 1:

Die Signalspeicherung für das Hauptschütz Q1 erfolgt im SPS-Ausgang A 4.0, daher genügt der Bausteintyp FC. Q1 ist als Durchgangsvariable (IN_OUT) deklariert, auf die SPS-Ausgangsadresse A 4.0 kann lesend und schreibend zugegriffen werden.



Modellierungstechniken
R&I Fließschema

- Rohrleitungs- und Instrumentenfließschema enthält
  - Art und Bezeichnung der Apparate und/ oder Maschinen
  - Rohrleitungen, Armaturen mit Nennweiten, Druckstufen, Werkstoffen
  - Antriebe
  - Aufgaben der Einrichtungen zum Messen, Steuern, Regeln

- Die Mess- und Regelstellen werden nach ISO 3511 in Ovalen gemäß den PCE-Kategorien dargestellt.

- Kommt zur Darstellung bei verfahrenstechnischen Anlagen zur Anwendung (Chemie, Pharmazie, etc.)

Modellierungstechniken

R&I Fließschema

- Erstbuchstabe – PCE-Kategorie
  - A Analyse (Analysis)
  - B Flammenüberwachung (Burner Combustion)
  - C Leitfähigkeit (Conductivity)
  - D Dichte (Density)
  - E elektrische Spannung (Voltage)
  - F Durchfluss (Flow)
  - G Abstand, Länge, Stellung (Gap)
  - H Handeingabe oder Handeingriff (Hand)
  - I Elektrischer Strom (Current)
  - J Elektrische Leistung
  - K Zeitbasierte Funktion (Time Schedule)
  - L Füllstand (Level)
  - M Feuchte (Moisture)
  - N Stellglied (Motor)
  - O Darf vom Anwender als Erstbuchstabe frei verwendet werden
  - P Druck (Pressure)
  - Q Anzahl, Menge (Quantity)
  - R Strahlung (Radiation)
  - S Geschwindigkeit, Drehzahl, Frequenz (Speed)
  - T Temperatur (Temperature)
  - V Schwingung (Vibration)
  - W Gewicht, Masse, Kraft (Weight)
  - X Darf vom Anwender als Erstbuchstabe frei verwendet werden
  - Y Stellventil

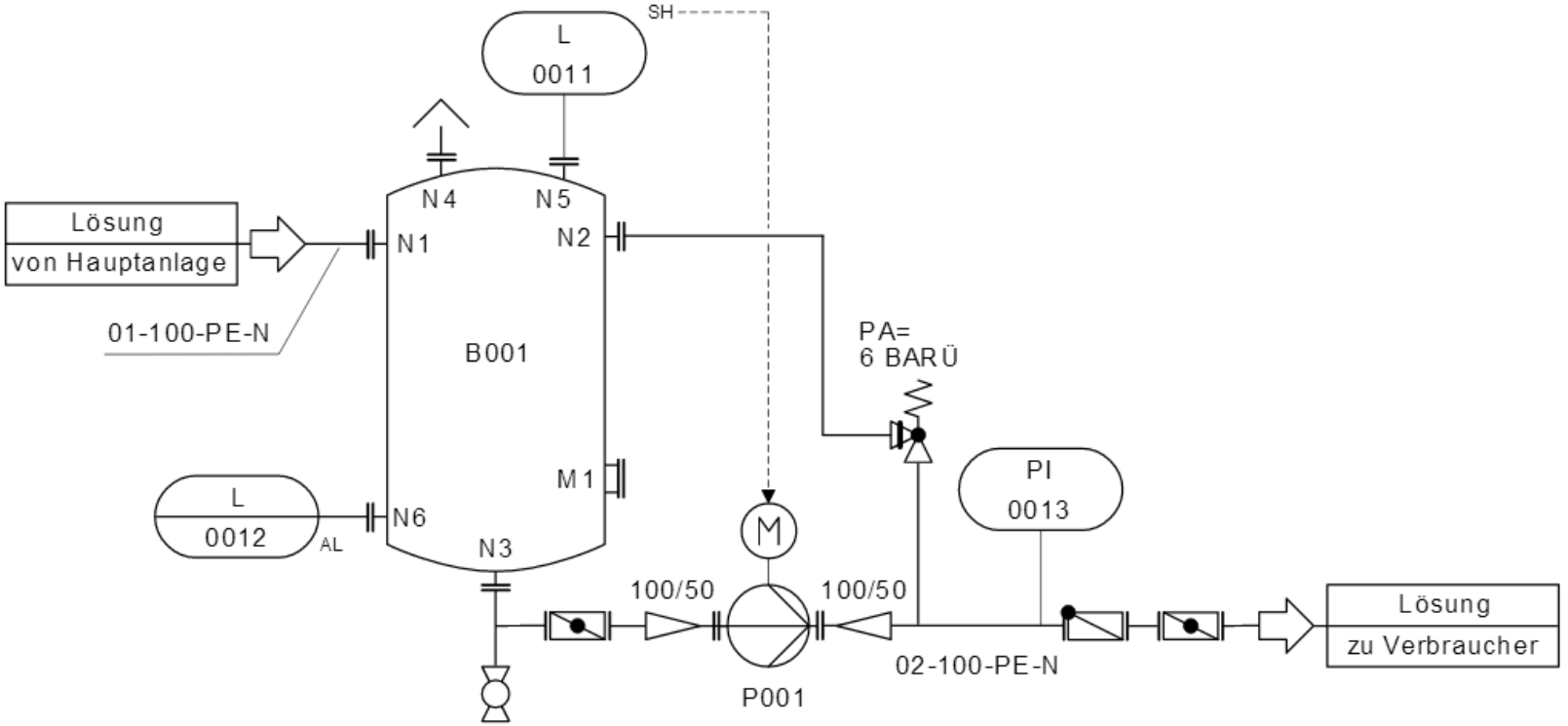
Modellierungstechniken
R&I Fließschema

- Folgebuchstabe – PCE-Kategorie
  - A Alarm, Meldung (Alarming)
  - B Beschränkung, Eingrenzung
  - C Regelung/Steuerung (Controlling)
  - D Differenz (Difference)
  - F Verhältnis (Fraction)
  - H oberer Grenzwert, an, offen
  - I Analoganzeige (Indicating)
  - L unterer Grenzwert, aus, geschlossen
  - O Sichtzeichen, Ja/Nein-Anzeige (keine Störungsmeldung)
  - Q Integral, Summe
  - R Speicherung/Aufzeichnung (Recording)
  - S Binäre Steuerungsfunktion oder Schaltfunktion (nicht sicherheitsrelevant) (Switching)
  - T Transmitter, für Analogwertverarbeitung (Monitoring)
  - Y Rechenfunktion
  - Z Binäre Steuerungsfunktion oder Schaltfunktion (sicherheitsrelevant)(Emergency)

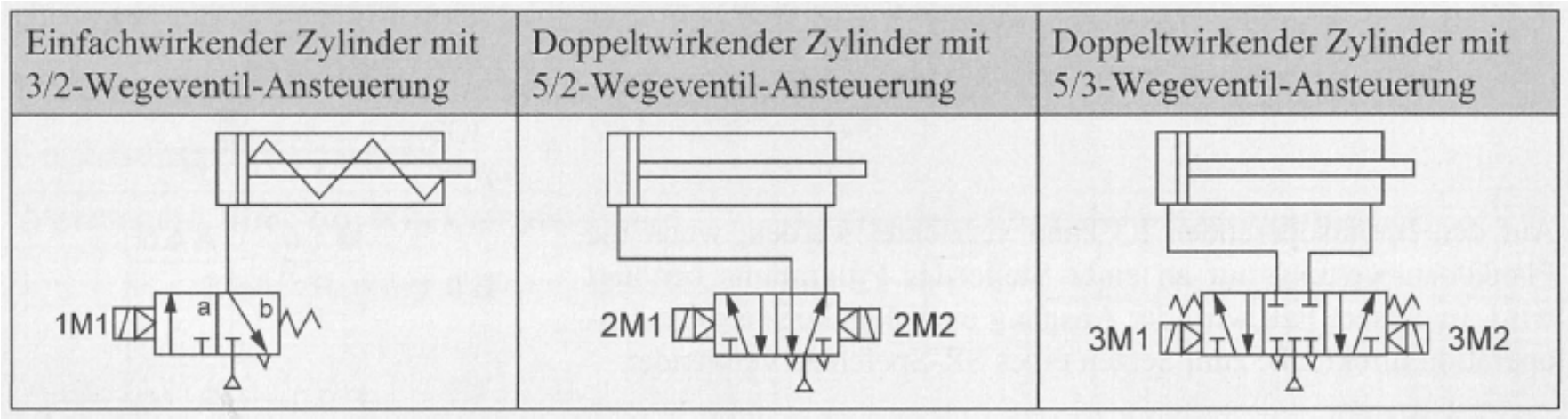
Modellierungstechniken
R&I Fließschema

- Beispiel
  - PI-512: Druckanzeige
  - PDI-512: Druckdifferenzanzeige
  - PICAH-512: Druckregelung mit Anzeige mit Alarm bei Überschreitung eines oberen Grenzwerts
  - F-100: aktueller Durchfluss
  - FQ-100: durchgeflossene Gesamtmenge
  - TIRCSHAHL-618: Temperaturregelung mit Anzeige, Speicherung, Schaltung bei Erreichen des oberen Grenzwertes und Alarm bei Erreichen des oberen und unteren Grenzwertes

	B001	P001
Benennung	Pufferbehälter	Nachspeisepumpe
Technische Daten	D = 1000 H = 3000 V = 2,4 m3	Q = 5 m3/h dp = 2,5 bar
Zulässiger Überdruck	10 bar	10 bar
Zulässige Temperatur	50 °C	50 °C



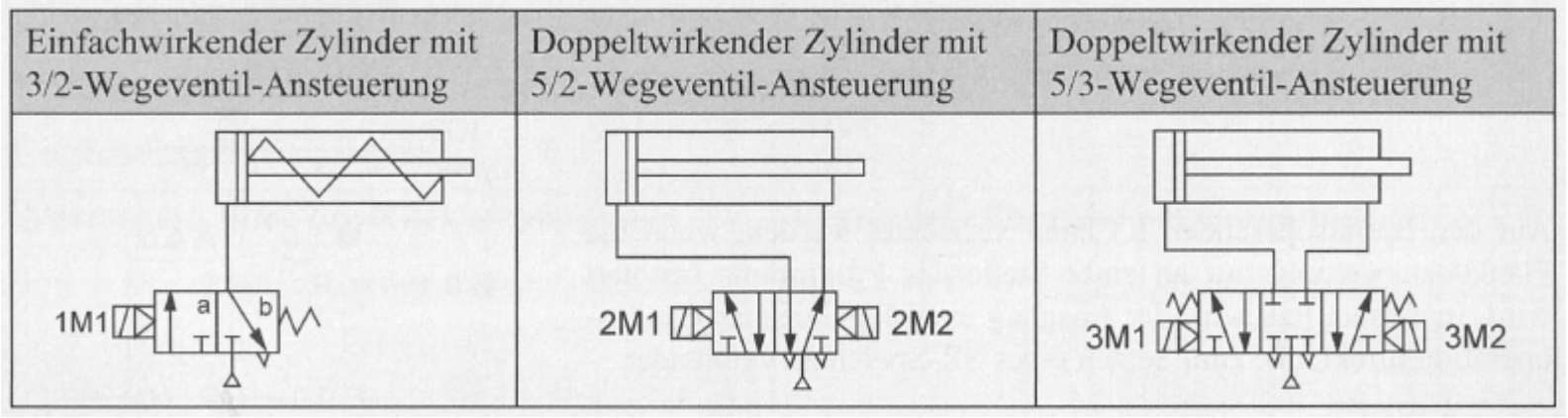
Einschub: Darstellung der Eigenschaften elektropneumatischer Stellglieder



3/2-Wegeventil:

- Elektromagnetisches Ventil mit Rückstellfeder
- hat nur einen elektrischen Steuereingang 1M1 und
- kann durch ein Stellsignal aus der Schaltstellung b in die Stellung a geschaltet werden.
- Nach Beendigung des Stellsignals erfolgt eine federmechanische Rückstellung des Ventils
- Vorteil: die definierte Schaltstellung im unbetätigten Zustand.

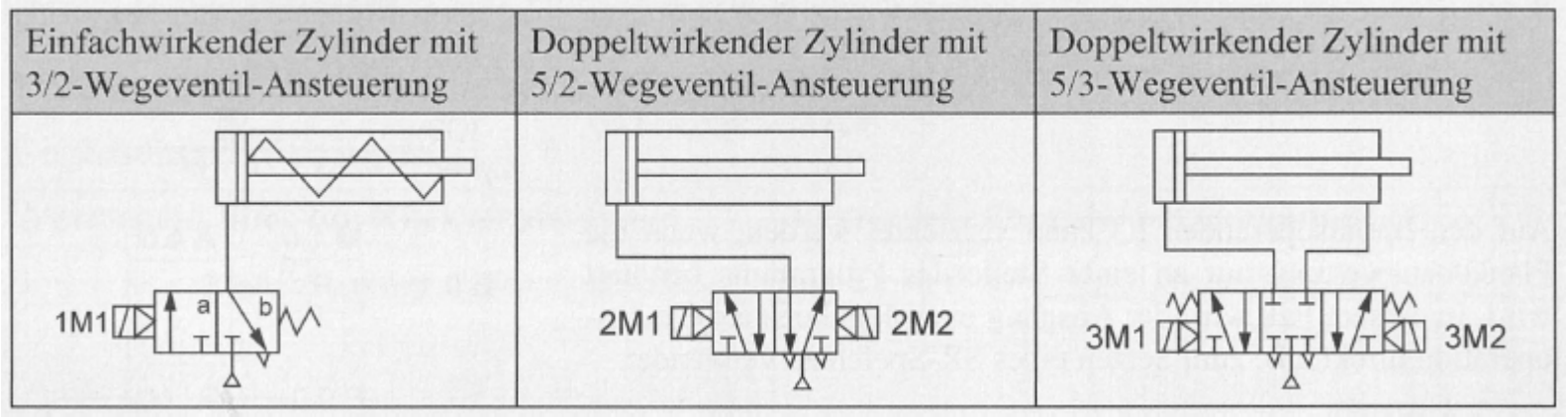
Einschub: Darstellung der Eigenschaften elektropneumatischer Stellglieder



5/2-Wegeventil:

- Elektromagnetische Impulsventile haben zwei elektrische Steuereingänge 2M1 und 2M2
- sie können durch kurze Ansteuerimpulse aus einer Schaltstellung in die andere umgeschaltet werden
- Die Ventile übernehmen die RS-Speicherfunktion der Steuerung
- Nachteil des Speicherverhaltens ist die nicht definierte Schaltstellung im unbetätigten Zustand
- daher erfolgt meistens Ansteuerung von 2Ml und 2M2 mit inversen Signalen

Einschub: Darstellung der Eigenschaften elektropneumatischer Stellglieder



5/3-Wegeventil:

- Bei elektromagnetischen Impulsventilen mit Federzentrierung geht das Ventil im unbetätigten Zustand in die Mittelstellung
- Daher kann neben der Vorwärts- und Rückwärtsbewegung des Zylinderkolbens auch eine Halteposition veranlasst werden.

Die Schaltverläufe von Ventilen und Zylindern werden häufig durch Funktionsdiagramme dargestellt.

- Funktionsdiagramme zeigen den Bewegungsverlauf von Zylindern.
- In der Ordinate wird der zurückgelegte Weg und in der Abszisse werden Schritte oder Zeiten aufgetragen.
- Zusätzlich können die Zustände von Magnetspulen der Ventile dargestellt werden.

Der Entwurf von SPS-Programmen: Zustandsdiagramme („Alte“ Form)

In Schaltwerken, die im Unterschied zu Schaltnetzen Speicherverhalten besitzen, muß bei einer Änderung der externen Variablen der innere Zustand des Schaltwerkes berücksichtigt werden. Dieser Zustand als zusätzliche Einflußgröße wird durch die Einführung von Zustandsvariablen berücksichtigt.

Die Ablaufbeschreibung in einem Zustandsgraphen zeigt übersichtlich alle Zustände und Zustandsänderungen, die ein Schaltwerk annehmen kann. Dabei können sowohl sequentielle Steuerungen für schrittweisen Ablauf als auch kombinatorische Steuerungen dargestellt werden.

Der Entwurf von SPS-Programmen

Die Methode der Zustandsgraphen bietet einen Weg, schnell und sicher SPS-Programme zu erstellen.

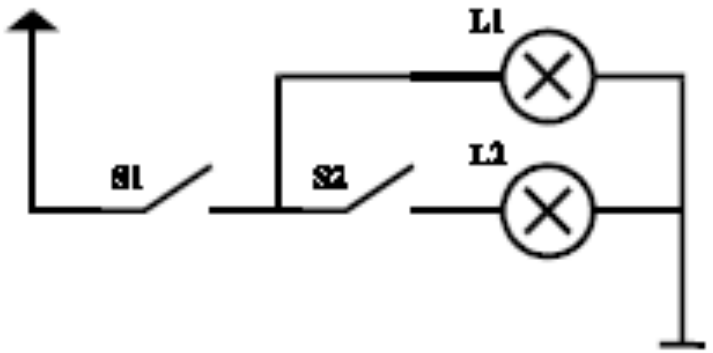
Diese Methode erlaubt die graphische Beschreibung von sequentiellen, insbesondere aber auch von nichtsequentiellen asynchronen Prozessen. Der entscheidende Vorteil für den Anwender besteht darin, daß die Darstellung nicht nur für den SPS-Programmierer geeignet ist, sondern auch für den Maschinenbauer (Technologen), den Inbetriebsetzer und den Service-Ingenieur verständlich ist.

Dazu sind erforderlich

- die Festlegung der für den Prozeßablauf wichtigen **Prozeßzustände**
- die Festlegung der notwendigen **Operation zur Überführung** eines Prozeßzustandes in einen anderen
- die Festlegung der **Reihenfolge**, in der sich die Prozeßzustände durch innere (Signale vom Prozeß) oder äußere (z.B. Bediener) Einwirkungen verändern können
- die **Prüfung** der durch die Steuerung zu realisierenden Prozeßzustände **auf Vollständigkeit**

Der Entwurf von SPS-Prog.: Zustandsdiagramme („Vollständige“ Form)

Nebenstehendes Beispiel zeigt eine Schaltung, in der zwei Lampen L1 und L2 durch die Schalter S1 und S2 eingeschaltet werden können.

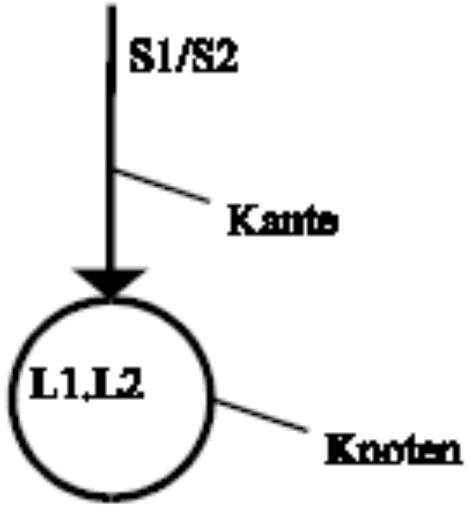


Die zwei Binärwerte in einem Knoten zeigen den Zustand der Lampen in der Reihenfolge L1, L2:

0.0	beide Lampen ausgeschaltet
1.0	nur Lampe L1 leuchtet
1.1	beide Lampen leuchten
1.2	

Die Binärwerte an den Kanten stellen den Zustand der Schalter S1/S2 dar:

0/0	beide Schalter geöffnet
0/1	Schalter S2 geschlossen usw .

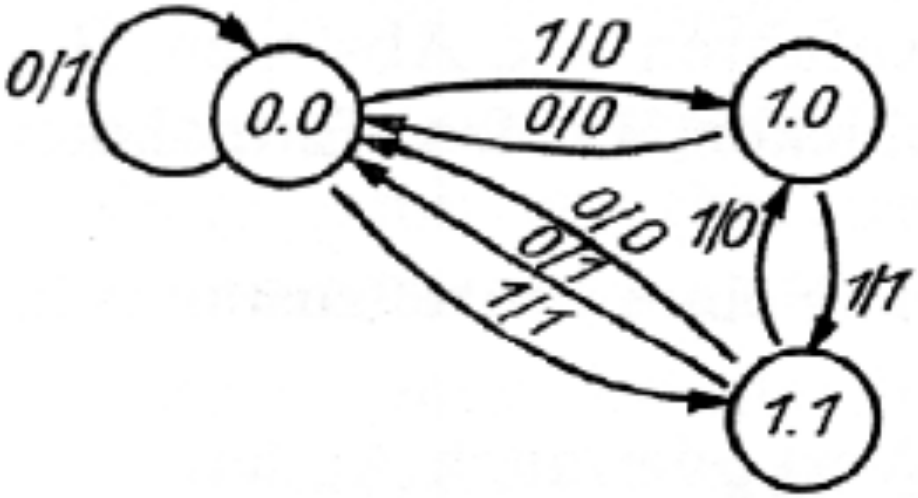


Der Entwurf von SPS-Programmen

Bei Veränderung einer Schalterstellung erfolgt der Übergang über die betreffende Kante zu einem neuen Prozeßzustand.

Bei dieser Darstellung stehen die Bedingungen für die Zustandsübergänge an den Kanten, die Signalzustände selbst sind in den Knoten der Prozeßzustände eingetragen. Diese Form der Zustandsgraphen wird auch als Petri-Netz bezeichnet.

Es werden für die Zustandsvariablen aller möglichen Prozeßzustände je ein Speicher benötigt. Die Ausgangsvariablen (Ausgabebefehle) werden mit den jeweilgen Zustandsvariablen entsprechend verknüpft.



Zustandsdiagramme: Neue Formen

- Zustandsdiagramme (auch Statemachines, Statecharts, Statediagrams, Zustandsgraphen, Zustandsautomaten, Automaten) sind in Steuerungstechnik und Informatik ein wichtiges Beschreibungsmittel für Zustandsorientierte Algorithmen und Protokolle geworden.
- Einfache Mealey und Moore-Automaten wurden durch höhere HAREL-Automaten abgelöst
- Innerhalb der UML (Unified Modelling Language der OMG) ist das State-Diagram in Form von erweiterten Harel-Automaten standardisiert (siehe UML 2.xx – Standard der OMG) und wird vorwiegend zur Verhaltensbeschreibung von Classifiern und für Protokollbeschreibungen eingesetzt.
- „Matlab Simulink“ – eine CFC-Sprache für die Beschreibung kontinuierlicher Regelungstechnischer Aufgaben (DGL-Modelle) besitzt mit „Stateflow“ eine Erweiterung mit der auch diskrete, zustandsorientierte Verhalten modelliert und automatisch in Programme übersetzt werden können.
- In der Steuerungstechnik wird zwar mittlerweile vieles in Zustandsgraphen modelliert, zu frühe „Vorstöße“ (z.B. Siemens Simatic S7 HiGraph) wurden aber vom sehr konservativen Steuerungstechnik- Markt nur sehr zögerlich angenommen.
- Häufig werden Zustandsdiagramme im Steuerungstechnik-Bereich manuell gezeichnet und dann über „Umsetzungsmuster“ in beliebige Programmiersprachen übersetzt. Fortschrittlichere Varianten erzeugen aus CASE-modellierten Diagrammen den Steuerungscode automatisch.

Zustandsdiagramme (Neue Formen)

Siehe auch: HiGraph und UML2.xxx Zustandsgraphen sowie Simulink Stateflow-Statecharts

Lernziel:

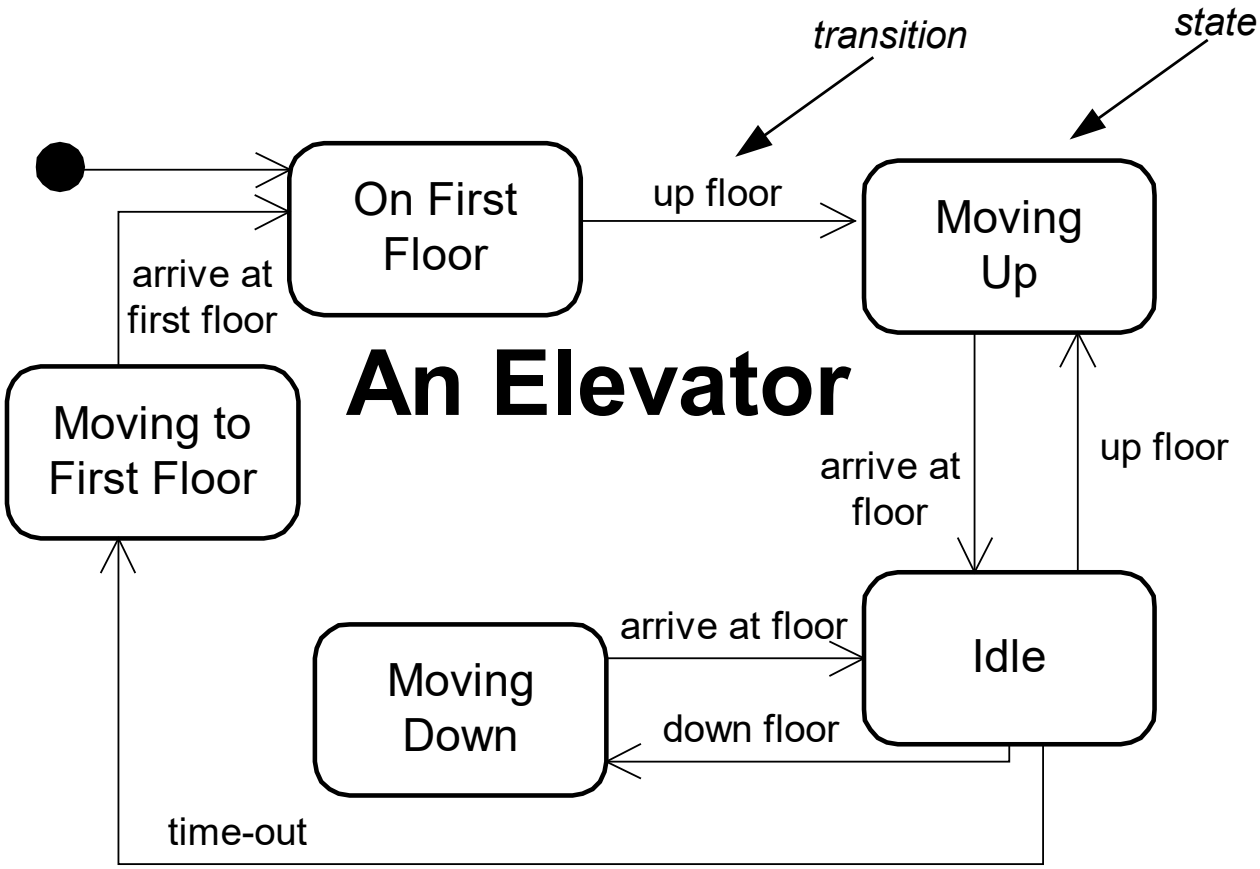
- Zustandsdiagramme einfacher bis mittlerer Komplexität selbst erstellen können
- Umsetzungsmuster in unterschiedliche Programmiersprachen kennen und selbst erstellen können
- Komplexe Zustandsdiagramme in Stateflow, S7 HiGraph und UML 2.xxx lesen und verstehen können

Modellierungstechniken
Zustandsautomat

- Elemente
  - *States* – der Zustand in welchem ein Objekt zu einem bestimmten Zeitpunkt ist.
  - *Transitions* – Der Weg um von einem Zustand in einen anderen zu kommen.
  - *Events* – Der Auslöser um einen bestimmten Weg von einem Zustand zu einem andern zu wählen.
  - *Actions* – Aktion welche beim Wechseln des Zustands ausgeführt wird (optional)

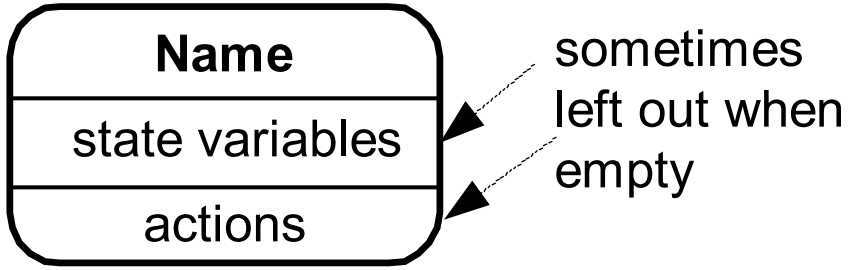
- Steuerungstechnisch relevante Erweiterungen in UML :
  - Hierarchische Diagramme
  - Guards – Bedingungen auf Transitionen

States and Transitions

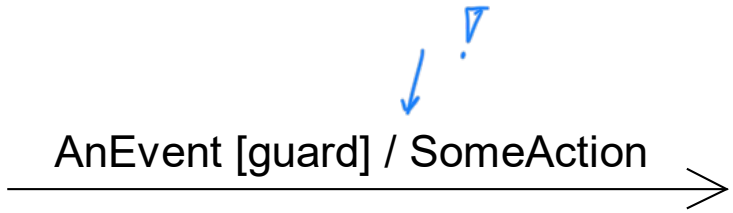


UML State Diagram Syntax

- Darstellung eines Zustands als abgerundetes Rechteck mit
  - Name
  - Zustandsvariable (falls vorhanden)
  - Actions



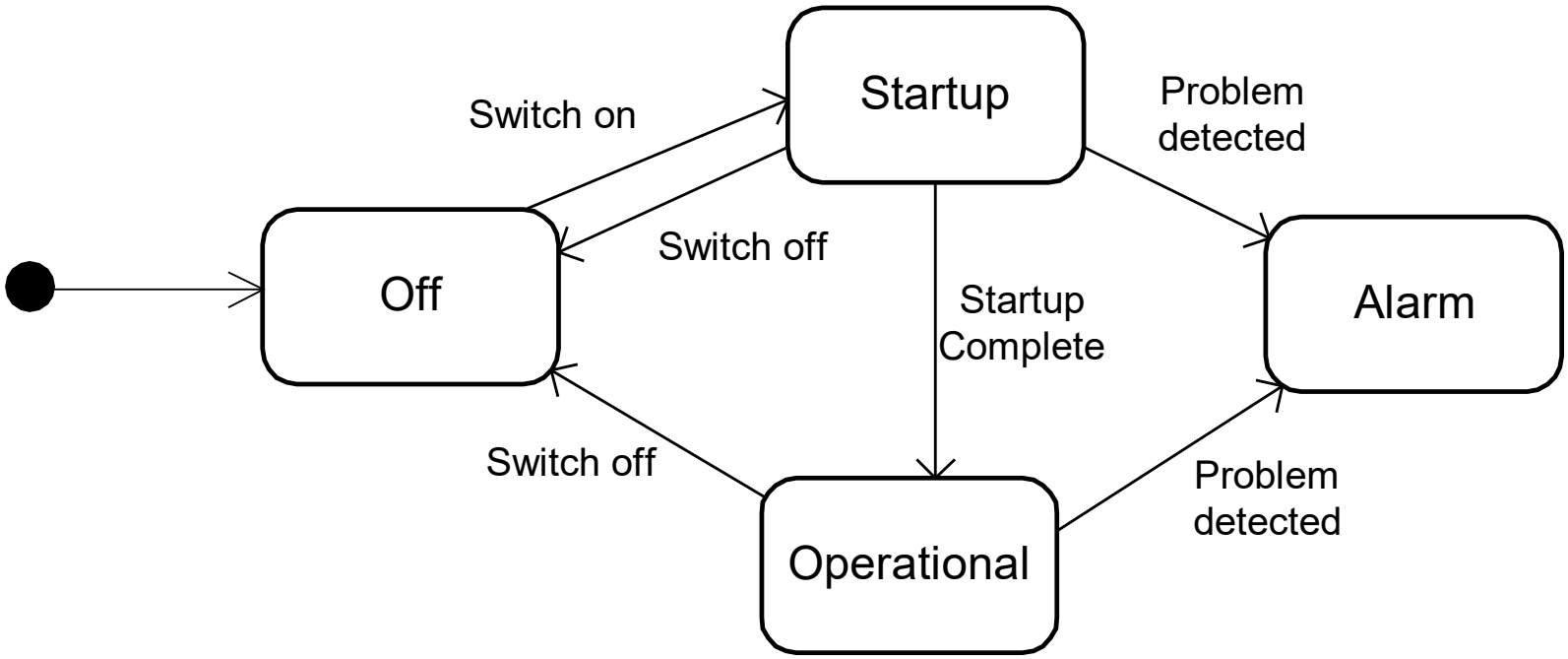
- Darstellung einer Transition als Pfeil mit Beschriftung
  - Ereignis/Signal welches die Transition auslöst
  - Guard Bedingung (boolsch)
  - Action to perform



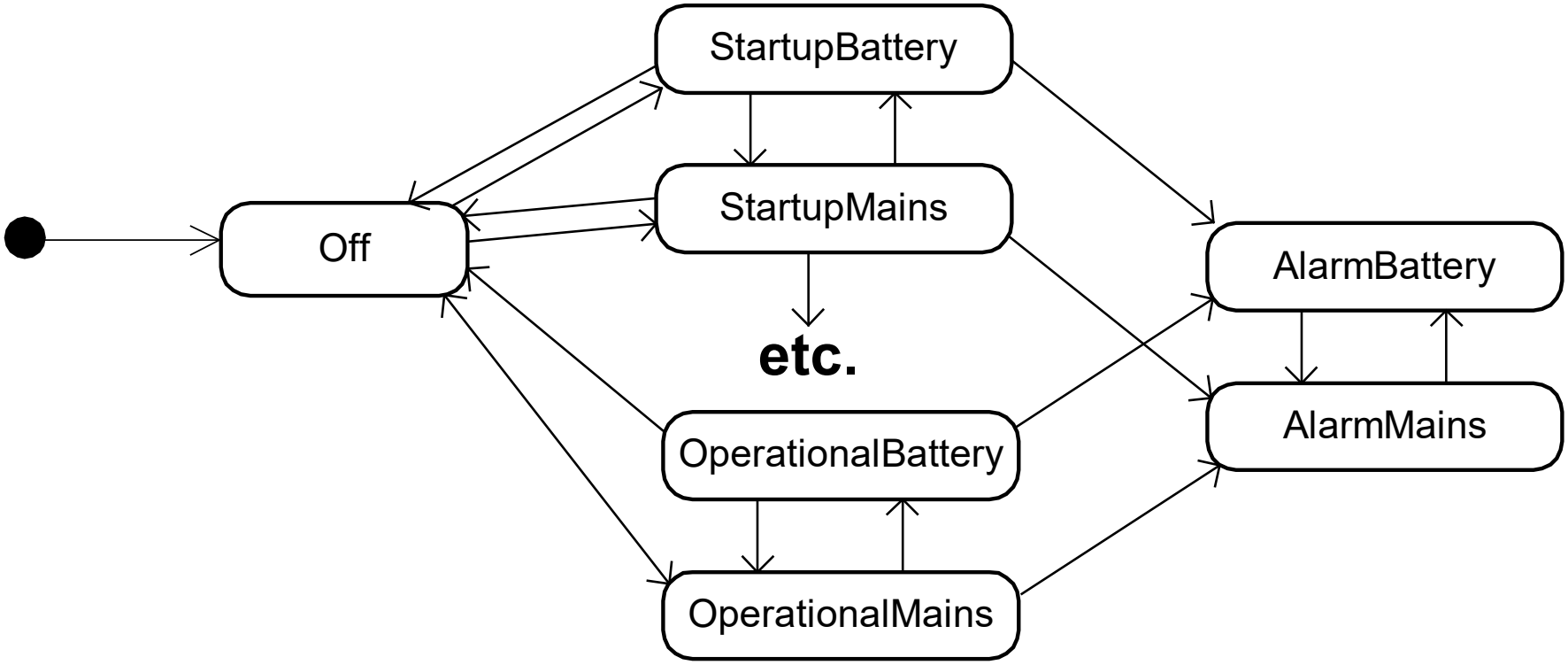
Vordefinierte Actions innerhalb eines Zustands

- **“entry/”**
  - Entry Action: wird 1x ausgeführt wenn ein Zustand betreten/aktiv wird.
- **“exit/”**
  - Exit Action: wird 1x ausgeführt wenn ein Zustand verlassen wird
- **“do/” (in IEC 61131 AS: “during/”, “cyclic/”)**
  - Kennzeichnet eine andauernde Aktivität.

Einfaches Beispiel:



Detaillierter Blick

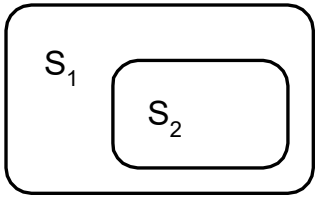


Kombinatorische Explosion

- Die Anzahl von Zuständen und Transitionen wird sehr schnell sehr groß
  - Problem auch bei Flussdiagrammen
- Klassische Mealy/Moore Automaten haben hierfür keine wirkliche Lösung, schränken aber die Modellierung ein:
  - Mealy Automaten erlauben Actions nur wenn eine Transition aktiv ist
  - Moore Automaten erlauben Actions nur beim Eintritt in einen State
- Für Steuerungstechnik daher eigentlich nicht geeignet,
  - daher auch in UML und in der Steuerungstechnik – sog. HAREL Automaten

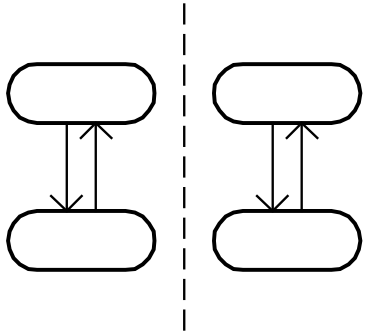
Harel Automaten

- Erlauben verschachtelte und hierarchische Zustandsautomaten
  - Reduziert Komplexität drastisch
  - Bildet Realität besser nach:
    - Maschinen sind oft hierarchisch aufgebaut
    - z.B: Handlingsystem > Servoachse > Motor + Drehgeber + Servoregler, …



- Erlauben parallel laufende Zustandsautomaten

- Erlauben:
  - Entry Actions
  - During Actions
  - Exit Actions

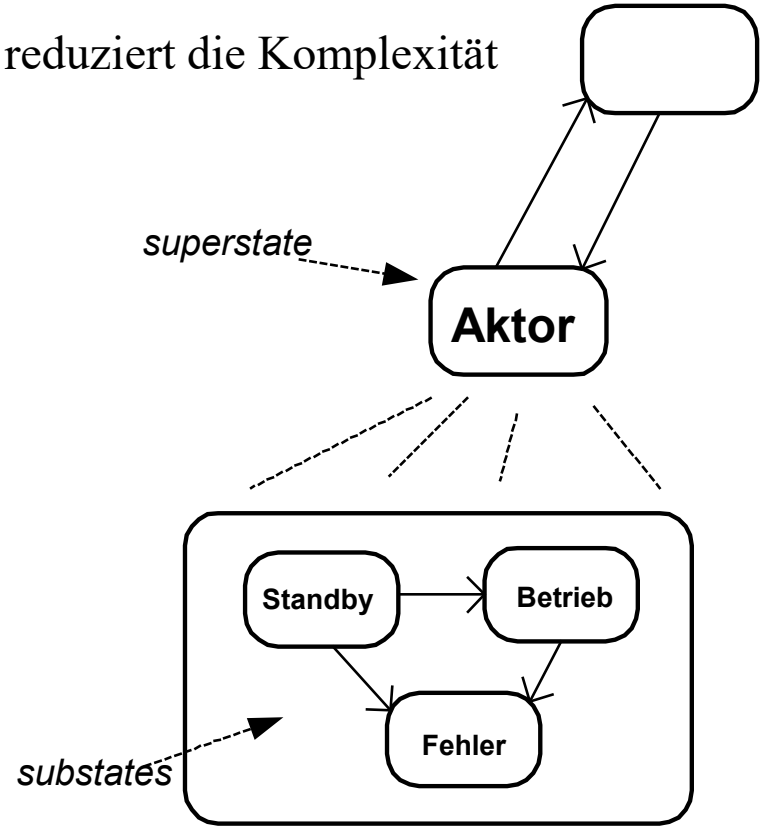


- Sind die Basis für Zustandsmodellierung in UML, Matlab Stateflow, etc.

Concurrent State Diagrams

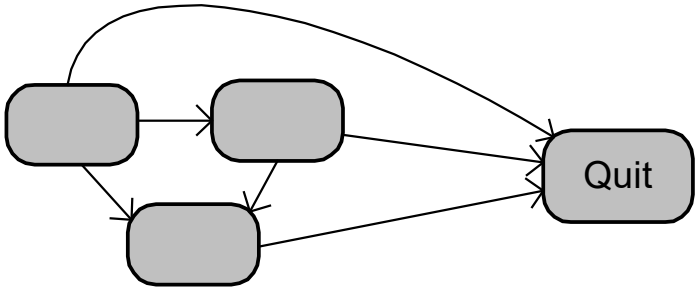
Verschachtelte Zustandsdiagramme

- Hierarchische Organisation reduziert die Komplexität
  - superstates
  - substates

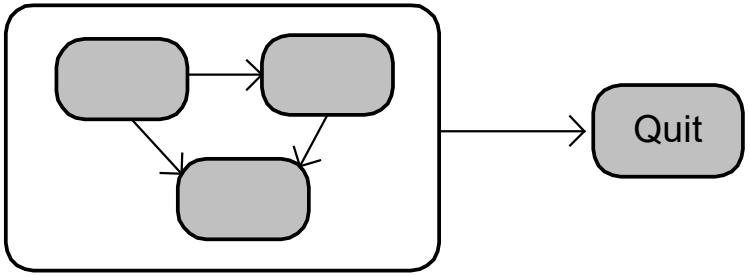


Mehrfach-Transitionen vermeiden

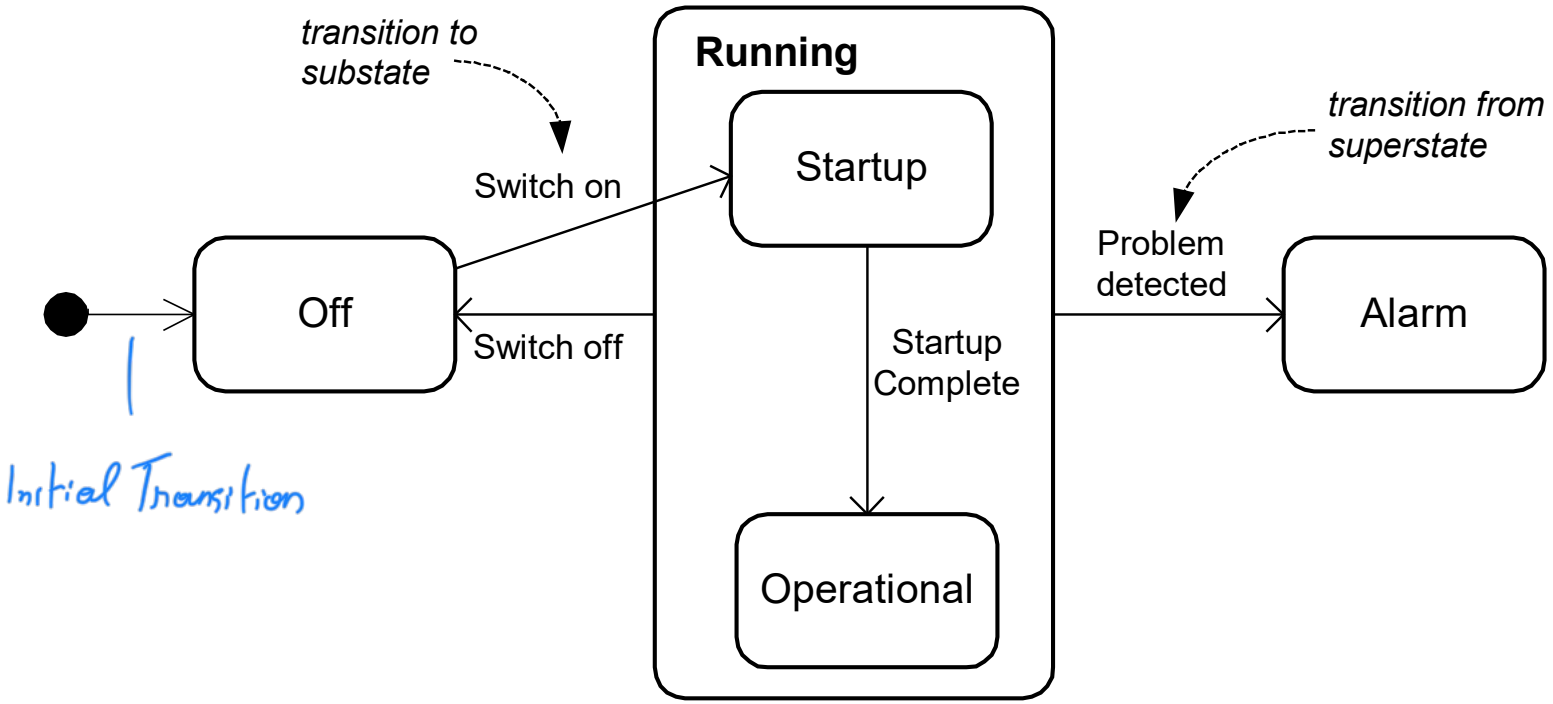
- Transitionen sind oft mehrfach vorhanden wenn eine Transition von mehreren oder allen Zuständen aktiviert werden kann:
  - “error”
  - “quit”
  - “abort”



- Diese Duplikate können mit einem Superstate zu einer Transition zusammengefasst werden

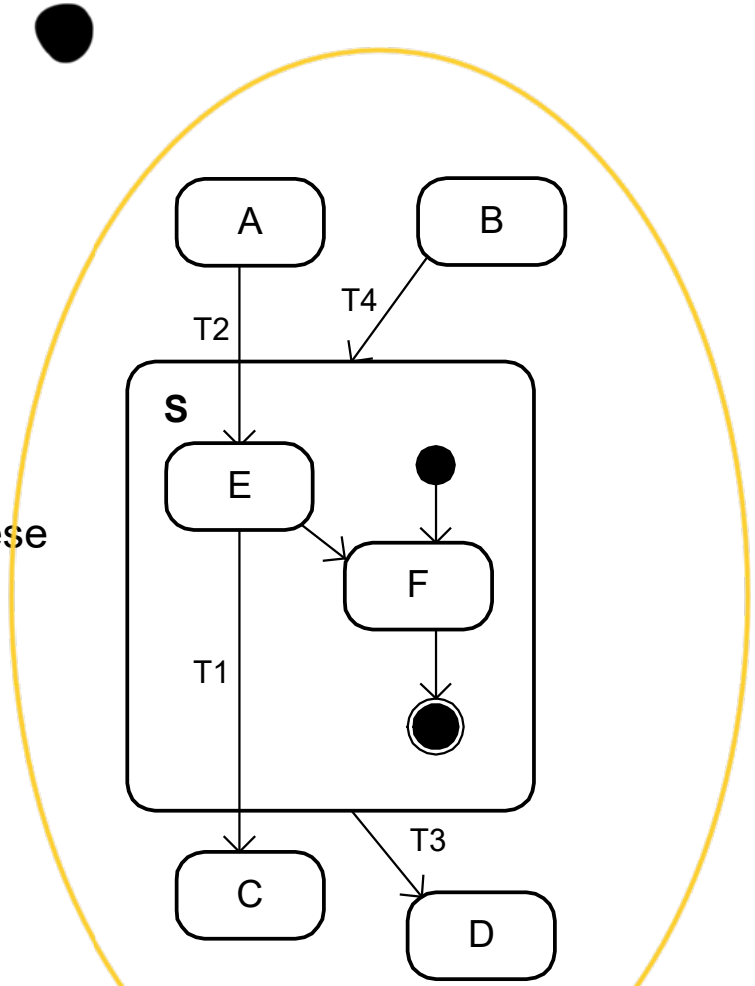


Anlage mit Super-States



Transitionen von und zu verschachtelten Zuständen

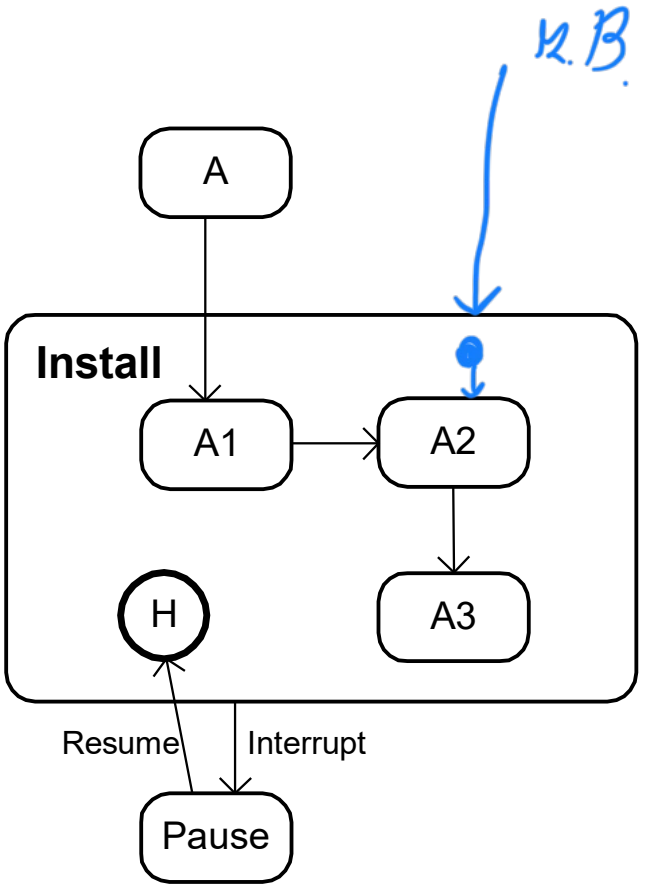
- Transitionen können spezifisch …
  - Von einem Substate (T1) wegführen
  - Zu einem Substate (T2) hinführen
- Transitionen können generell
  - Von einem SuperState wegführen. Diese gilt dann für alle Substates (T3).
  - In einen Superstate führen (T4). Dann wird der Default-Zustand aktiviert (F)



Nicht korrektes Pkg.
Es fehlt die übergeordnete Initial Transition

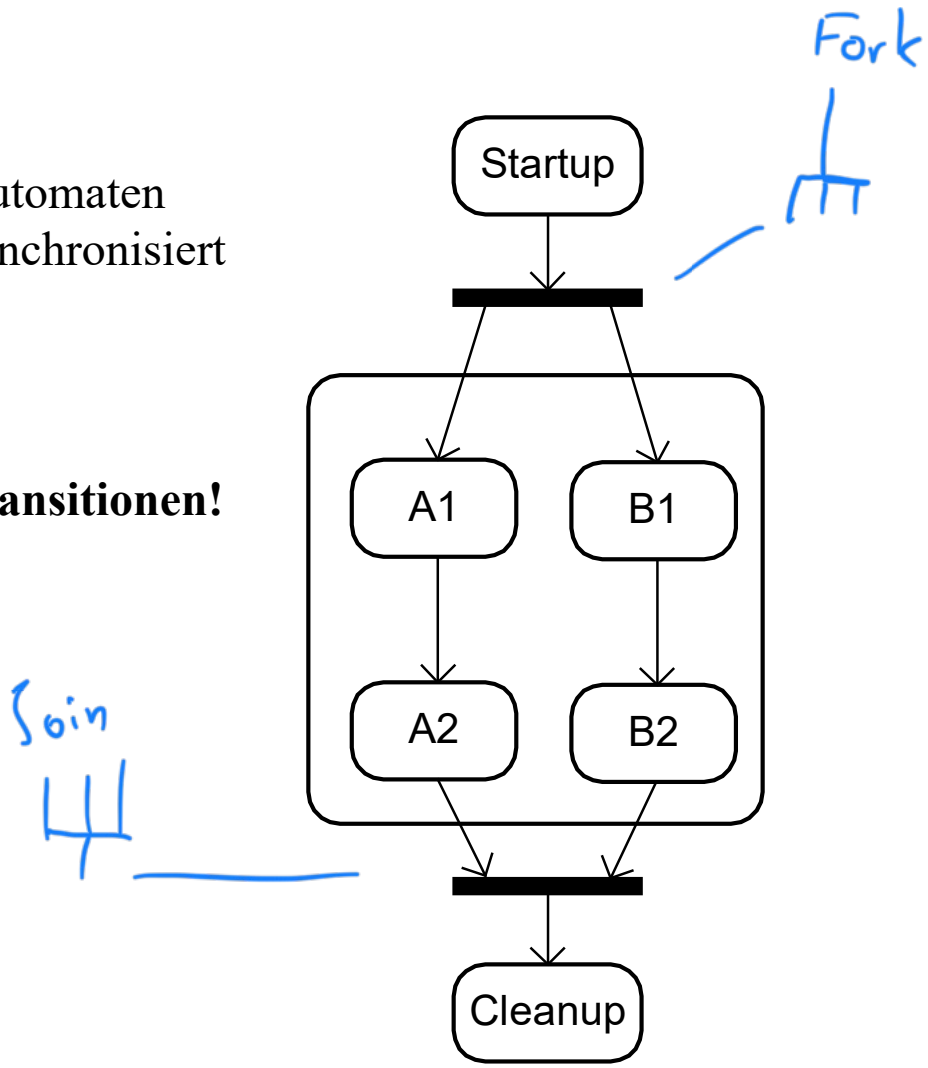
State-Ausführung anhalten und wiederaufnehmen

- Manchmal ist es notwendig eine komplexe Prozedur zu pausieren um kurz auf ein anderes Signal reagieren zu können und dann in der ursprünglichen Prozedur weiterarbeiten zu können (= Interrupt).
- Dafür gibt es in State-Charts des History-Indicator
- Die Transition “Interrupt” unterbricht den “Install”
  Superstate in jedem Zustand.
- Die Transition “Resume” führt in den “Install”
  Zustand an jene Position zurück wo vorher
  unterbrochen wurde (History)



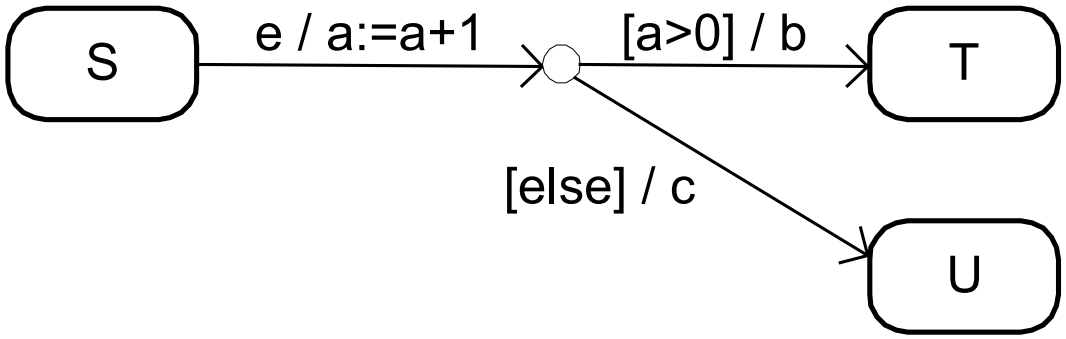
Komplexe Transitionen

- Parallel laufende Zustandsautomaten müssen unter Umständen synchronisiert werden:
  - Gemeinsamer Start
  - Re-sychronize am Ende
- **Dafür gibt es komplexe Transitionen!**



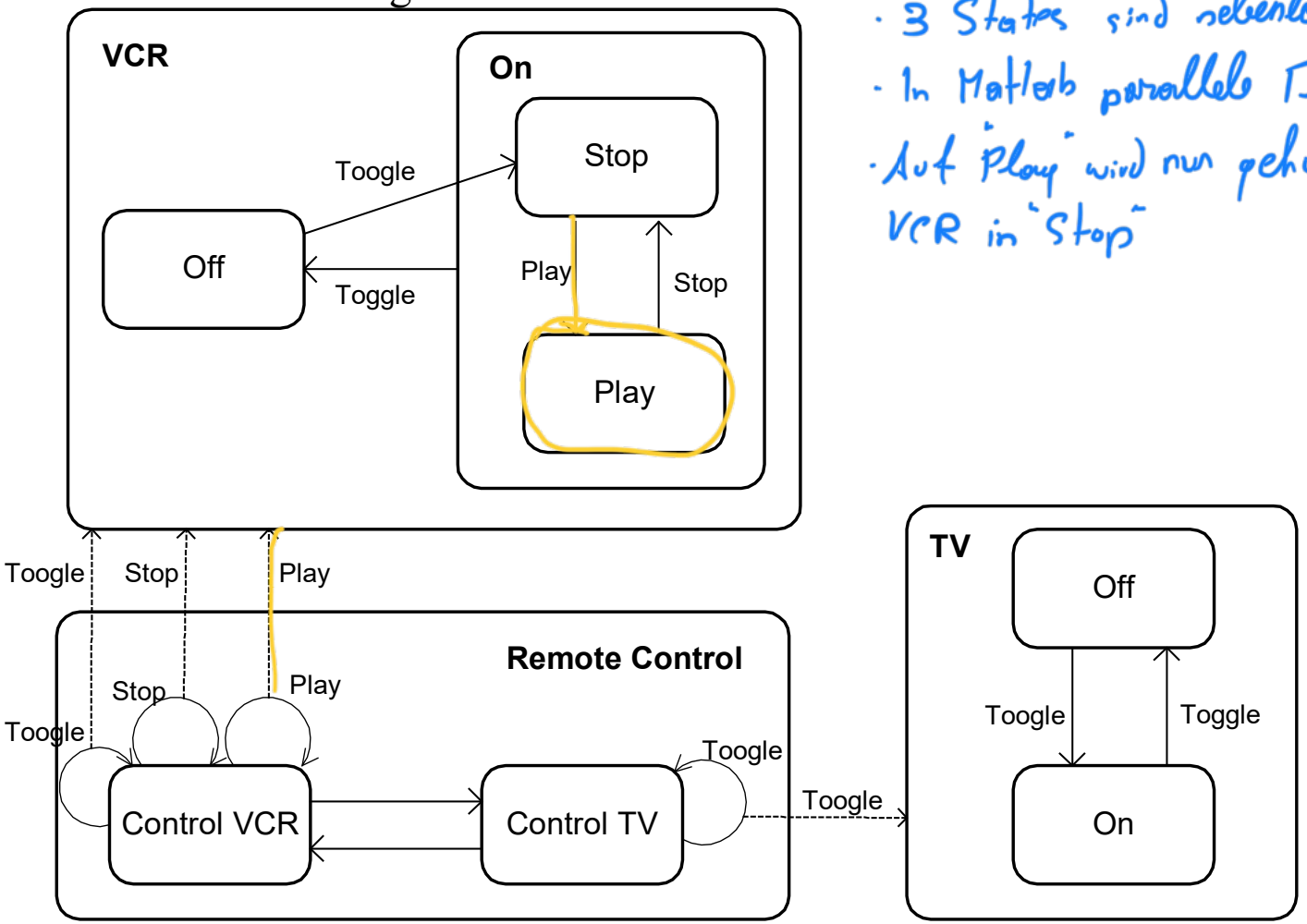
Dynamic Choice Points

- Die ausgehende Transition dessen Guard TRUE ist wird aktiviert.
- Wenn mehrere Guards TRUE sind dann wird die 1. Transition genommen.
  - Was ist die 1. Transition?
- Man kann/Muss explizit eine Überpüfungsreihenfolge vorgeben.



Synchronsierung über Messages

- Messages werden in IEC als globale Variable realisiert!
- Lese/Schreibzugriffe beachten

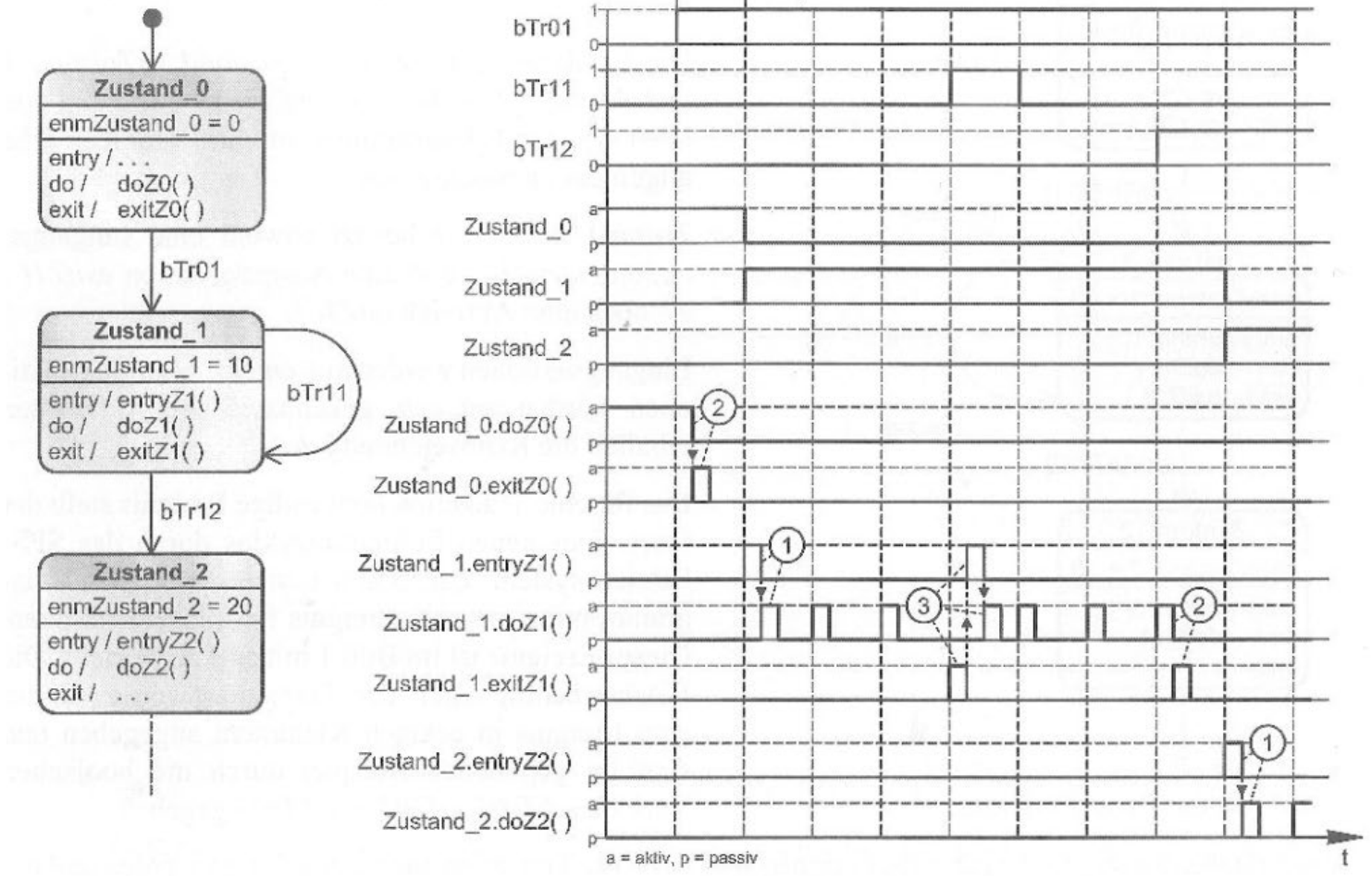


- 3 States sind nebenläufig
- In Matlab parallele Threads ----
- Auf "Play" wird nur gehört wenn VCR in "Stop"

Strategie für Implementierung

- Definieren der Eingangsvariablen
- Definieren der Ausgangsvariablen
- Definieren der Statusvariablen und der Zustände
- Definieren der Events / Ereignisse / Signale
- Definieren der Aktionen
- Definieren der Transitionen
- Zeichnen des Zustandsdiagrammes / Flussdiagramm
  - Anmerkung: Jetzt sollten Sie die Maschine vollständig kennen.
- Implementierung
  - Zustandsautomat in C oder ST
  - Vermeiden Sie „überlange“ Schrittketten in SFC
    - Eher in Prozessleittechnik üblich!

Beispiel und Ausführung



Beispiel: Implementierung State Chart in ST

Umsetzungsmuster
- Ein State pro Zyklus

```
-   CASE currentState OF
        STATE_XXX:
                    // Entry action
                    IF (lastState <> currentState) THEN
                                lastState := currentState;

                                // TODO: Write your entry action code here
                    END_IF

                    // During
                    // TODO: Write your during code here

                    // Exit action / Transition
                    IF exit_condition_1 THEN
                                lastState := currentState;
                                currentState := STATE_YYY;

                                // Write your exit action code here
                    ELSIF exit_condition_2 THEN
                                lastState := currentState;
                                currentState := STATE_ZZZ;
                    END_IF

        STATE_YYY:

-   END_CASE
```

Beispiel: Implementierung State Chart in ST

- Implementierungshinweise:
  - Zustandsvariable als ENUM definieren
    - Benannten Zustände sind besser zu testen als 0,1,2,3..
  - LastState, CurrentState
    - Alten Zustand merken, damit man einen State-Entry detektierten kann
  - Zustandsaktionen können wie folgt implementiert werden:
    - Inline Code
    - Als Funktionsbaustein (ermöglicht Wiederverwendung)
    - Als eigenen Task, der über eine bool‘sche (globale) Variable freigegeben wird

```
IF currentState = STATE_XXX THEN
// during code
END_IF
```

    - Als B&R IEC Action (nicht empfohlen, da schlecht testbar)

Tools zur Modellierung

- Nur Zeichnen:
  - Microsoft Visio
  - yEd (gratis)
- UML-konform mit Codegenerierung
  - Enterprise Architect
  - Visual Studio
  - Rational Software Modeller
  - StarUML
  - MagicDraw
- Steuerungstechnik
  - Matlab Simulink Stateflow + Codegenerierung für
    - B&R
    - Beckhoff
    - Bachmann
    - Siemens

> Siemens
- S7 HiGraph
-

Der Entwurf von SPS-Programmen: Funktionsplan („Alte“ Form)

6.5 Funktionsplan als Struktur für Ablaufsteuerungen

Der Funktionsplan nach DIN 40 719 Teil 6 ist eine prozeßorientierrte Darstellung einer Steuerungsaufgabe, unabhängig von deren Realisierung, z.B. der verwendeten Betriebsmittel, der Leitungsführung, dem Einbauort oder dem Steuerungssystem.

Der Funktionsplan ersetzt bzw. ergänzt die verbale Beschreibung der Steuerungsaufgabe.

Der Funktionsplan dient als Verständigungsmittel zwischen Hersteller und Anwender sowie zwischen Projektant, Konstrukteur und Programmierer.

Er erleichtert das Zusammenwirken verschiedener Fachdisziplinen, z.B. Verfahrenstechnik, Maschinenbau, Elektrotechnik, Informatik, Hydraulik, Pneumatik usw.

Der Funktionsplan stellt eine Steuerungsaufgabe mit ihren wesentlichen Eigenschaften (Grobstruktur) oder mit den für die jeweilige Anwendung erforderlichen Details (Feinstruktur) übersichtlich und eindeutig dar.

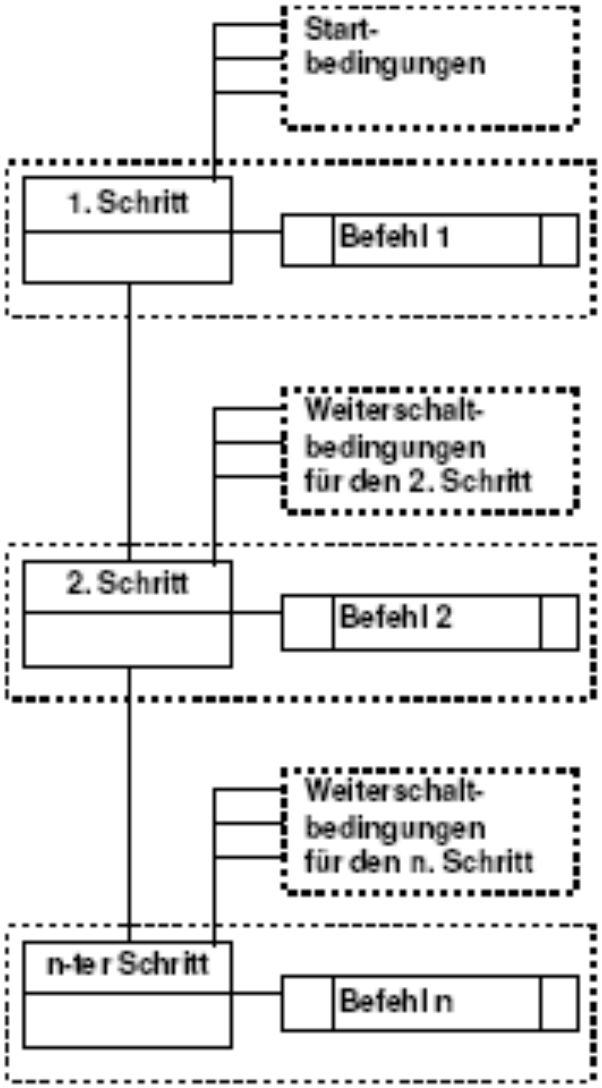
Der Entwurf von SPS-Programmen

Die Steuerungsaufgabe wird mit Hilfe von graphischen Symbolen dargestellt.

Die Beschreibung durch Funktionspläne ist besonders geeignet für Steuerungsprozesse mit schrittweisem Ablauf.

Die Struktur der Ablaufsteuerung und deren graphische Darstellung orientiert sich streng am schrittweisen Ablauf des Prozesses. Bezüglich der einzelnen Steuerungsschritte gilt folgendes:

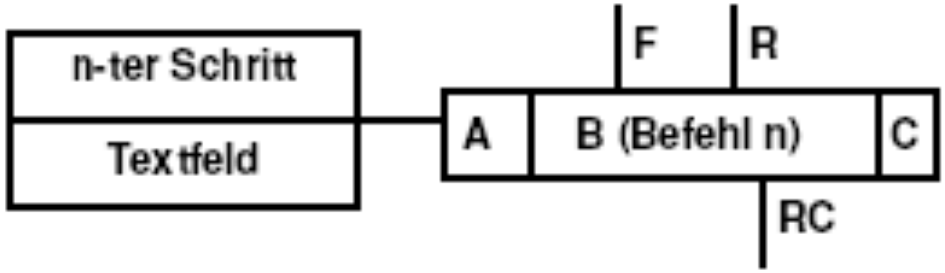
- Die Abarbeitung eines Schrittes erfolgt nur dann, wenn der vorherige Schritt abgearbeitet wurde (Ergebnis gleich Logik-1) und die Weiterschaltbedingungen erfüllt sind. Dies ergibt eine Logik-UND-Verknüpfung der genannten Signale.
- Mit den vorher genannten Bedingungen für die Aktivierung eines Schrittes wird ein RS-Flipflop gesetzt. Mit dem gesetzten Ausgang wird der vorherige Schritt zurückgesetzt. So kann immer nur der gerade betrachtete Schritt den Signalwert Logik-1 führen.
- Es muß auch die Möglichkeit gegeben sein, mit bestimmten Signalen jeden Schritt zurückzusetzen, z.B. mit dem Signal AUS.
- Wie sich der Signalwert Logik-1 eines Schrittes auf die Stellgeräte des gesteuerten Prozesses auswirkt, wird in der Befehlsausgabe bearbeitet.



Der Entwurf von SPS-Programmen

Feld A: Art des Befehls

- NS nicht gespeichert,
- NSD nicht gespeichert und verzögert,
- S gespeichert,
- SD gespeichert und verzögert,
- ST gespeichert und zeitlich begrenzt,
- SH gespeichert, auch bei Energieausfall,
- T zeitlich begrenzt.



Schrittsymbol Befehlssymbol

Der Logik-1-Wert eines Schrittes kann, wie im Feld A symbolisch bezeichnet, auf unterschiedliche Art ausgegeben werden. Die funktionelle Verknüpfung für Speicherung und Verzögerung eines Signals wird z.B. symbolisch mit SD gekennzeichnet. Die detaillierte Verknüpfung wird aber letztlich im Steuerungsprogramm dargestellt.

Feld B: Wirkung des Befehls, z.B. Motor EIN, Ventilklappe AUS.

Feld C: Kennzeichnung für Abbruchstelle eines Befehlsausganges.

Das Weiterschalten von Schritt n auf n+1 und die folgenden ist manchmal von der Ausführung der von Schritt n erteilten Befehle abhängig. Im Feld C werden, zur Vereinfachung der Darstellung, diesen Befehlen und deren weiterschaltenden Rückmeldungen Ziffern zugeordnet.

6.6 Bausteindarstellung als Grobstruktur

Programmstruktur

Programme können erstellt werden als

- lineare Programme

  Ein lineares Programm ist zur Gänze in einem einzigen zusammenhängenden Baustein abgelegt

  Das lineare Steuerungsprogramm hat eine einfache, geradlinige Struktur. Es gibt immer nur einen Codebaustein, der alle Operationen für das Programm enthält.

  Da sich alle Operationen innerhalb eines Bausteins befinden, ist diese Programmiermethode am besten geeignet für (kleine) Projekte, an denen nur ein Programmierer arbeitet.

6.6 Bausteindarstellung als Grobstruktur

Programmstruktur

Programme können erstellt werden als

- lineare Programme

- gegliederte Programme

  Ein gegliedertes Programm ist in mehrere Bausteine (Funktionen) unterteilt, von denen jeder die Logik für eine bestimmte Gruppe von Geräten oder Aufgaben enthält. Die Operationen des auf-rufenden Bausteins bestimmen, wie die einzelnen Bausteine des CPU-Programms bearbeitet werden.

  So kann ein gegliedertes Programm beispielsweise die folgenden Elemente enthalten:

  - Baustein für das Steuern einzelner Geräteteile
  - Baustein für das Steuern der einzelnen Betriebszustände der Geräte
  - Baustein für das Steuern der Bedien- und Beobachtungssysteme
  - Baustein für das Bearbeiten der Diagnoselogik

  In einem gegliederten Programm werden weder Daten getauscht noch ein Code mehrfach verwendet. Jeder Funktionsbereich wird aber in verschiedene Bausteine zerlegt. Dadurch können mehrere Programmierer gleichzeitig arbeiten, ohne daß die gleiche Datei bearbeitet werden muß.

ACHTUNG: nach IEC 1131-3 hat eine Funktion immer nur einen Rückgabewert!

6.6 Bausteindarstellung als Grobstruktur

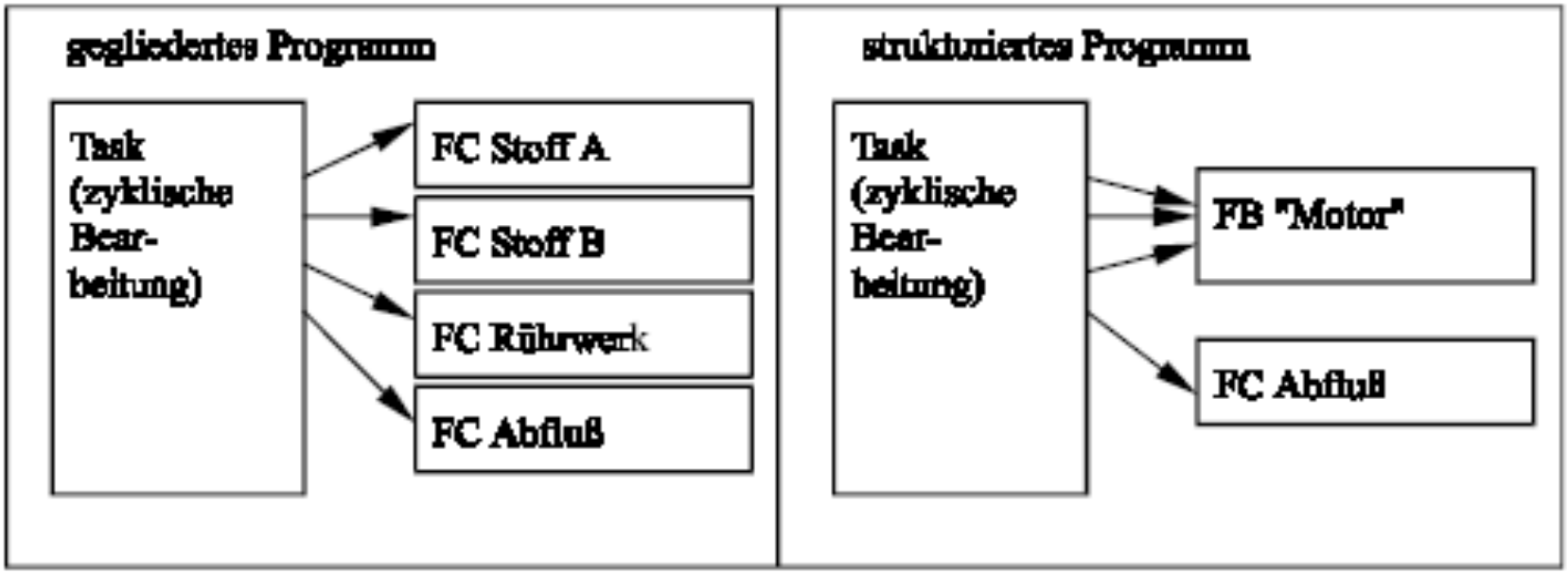
Programmstruktur

Programme können erstellt werden als

- lineare Programme
- gegliederte Programme
- strukurierte Programme

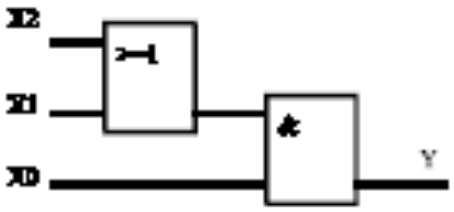
  Ein strukturiertes Programm enthält anwenderdefinierte Bausteine mit Parametern, vergleichbar mit anwenderdefinierten Anweisungen. Wenn Sie ein Programm für einen Prozeß oder eine Anlage erstellen, werden für gemeinsame Geräte oder logische Funktionen häufig Teile der Steuerungslogik wiederholt. Statt diese Anweisungen zu wiederholen und die verschiedenen Adressen für bestimmte Gerate zu ersetzen, können Sie die Operationen in einen Baustein schreiben und vom Programm die Parameter an den Baustein übergeben lassen (z. B. die verschiedenen Adressen der Geräte und Betriebsdaten).

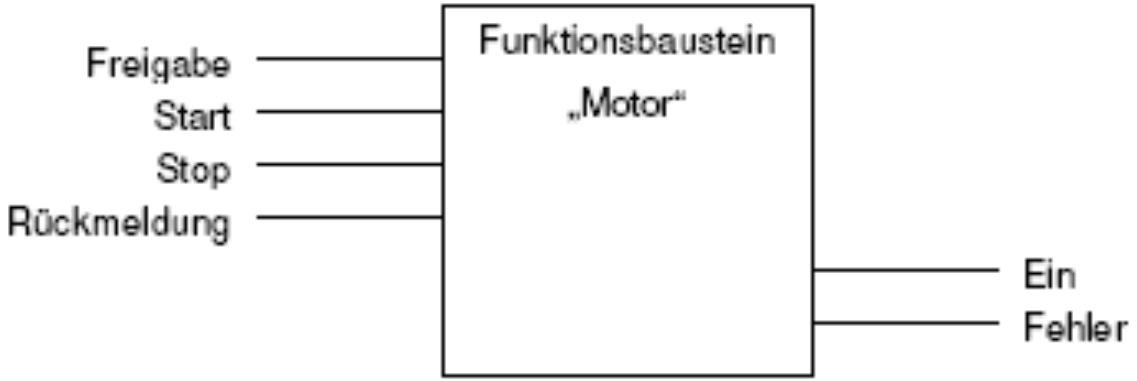
Der Entwurf von SPS-Programmen



Beispiel: Baustein für alle Motorfunktionen

Details können im Entwurf mit den bekannten Logiksysmbole dargestellt werden.





6.7 Ablaufstrukturmethode

Programmabläufe von Steuerungsprogrammen, denen ein Algorithmus zu Grunde liegt, verwenden häufig die drei folgenden Ablaufstrukturen:

- Folge: die Verarbeitung von Schritten nacheinander;
- Verzweigung: die Auswahl von bestimmten Schritten;
- Wiederholung: die Wiederholung von Schritten (Schleifen).

Mit diesen drei elementaren Ablaufstrukturen kann nach Festlegung des Algorithmus das Beschreibungsmittel Programmablaufplan oder Struktogramm zur Lösung von Steuerungsaufgaben eingesetzt werden.

In Programmsblaufplänen werden die Schritte symbolisch durch Sinnbilder dargestellt und der Steuerungsablauf durch Ablauflinien hergestellt. Zur besseren Übersichtlichkeit und leichteren Verständlichkeit von Programmablaufplänen werden in der DIN 66263 Programmkonstrukte zur Bildung von Programmen mit abgeschlossenen Zweigen festgelegt. Damit ist eine Grundlage gegeben, auf der jeder Programmablauf in einheitlicher Weise aus wenigen Bausteinen gebildet werden kann. In Struktogrammen werden sog. Strukturblöcke (rechteckig) kantendeckend aneinander gereiht, wobei die Ausgangskante zugleich die Eingangskante des nächsten Strukturblockes ist.

Der Entwurf von SPS-Programmen

Ein Programmkonstrukt oder ein Strukturblock ist eine abgeschlossene funktionale Einheit, welche keine Überlappung mit anderen Programmkonstrukten zuläßt.

Jedes Programmkonstrukt/jeder Strukturlock hat einen Eingang und einen Ausgang und besteht aus einem Steuerungsteil und einem oder mehreren Verarbeitungsteilen.

Während im Steuerungsteil die Reihenfolge und Häufigkeit der Ausführungen der Verarbeitungsteile festgelegt wird, ist ein Verarbeitungsteil ein abgeschlossener Zweig oder eine elementare Anweisung, jedoch keine Sprunganweisung.

In einem Programmkonstrukt/ Strukturblock kann zur Detaillierung jeder Verarbeitungsteil wieder durch einen Programmkonstrukt/ Strukturblock ersetzt werden.

Der Programmablauf wird durch Schachtelung von Programmkonstrukten/Strukturblöcken gebildet.

- Verarbeitung: Beschreibung einer einzelnen Anweisung;
- Folge: Zusammenfassung einer Sequenz von Programmkonstrukten/Strukturblöcken;
- Verzweigung: Alternative Ausführung von Programmkonstrukten/Strukturblöcken;
- Wiederholung: Wiederholte Ausführung eines Programmkonstruktes/Strukturblocks (Schleifen).

G: gemeinsamer Bedingungsteil

V: Verarbeitung

B: Bedingung