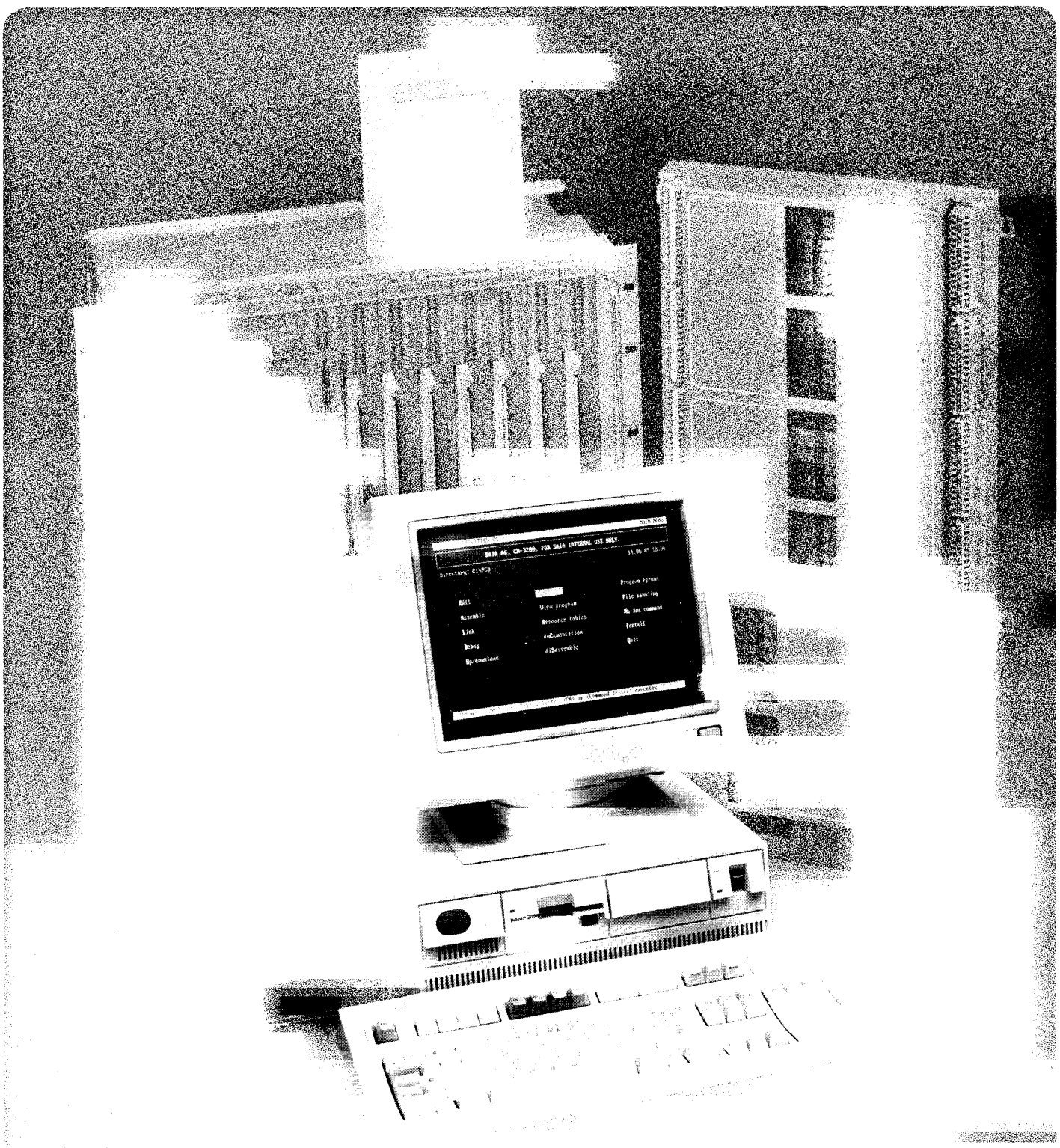
SAIA®PLC

Speicherprogrammierbare Steuerungen

Handbuch Software Stufe 1 H



Deutsche Ausgabe 26/721 D1

SOFTWARE NIVEAU 1H

KAPITEL D EINFÜHRUNG

KAPITEL E BEFEHLSSATZ UND PROGRAMMIERUNG

KAPITEL F PROGRAMMIER-BEISPIELE

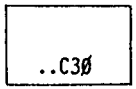
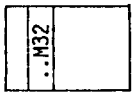
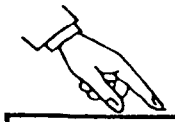
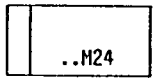
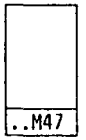
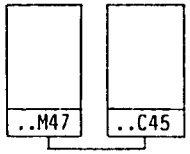
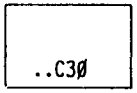
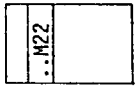
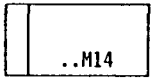
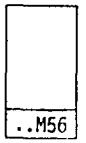
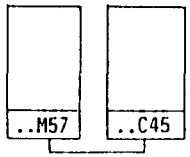
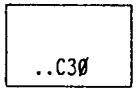
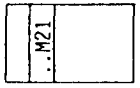
INHALTSVERZEICHNIS

TEIL D EINFÜHRUNG

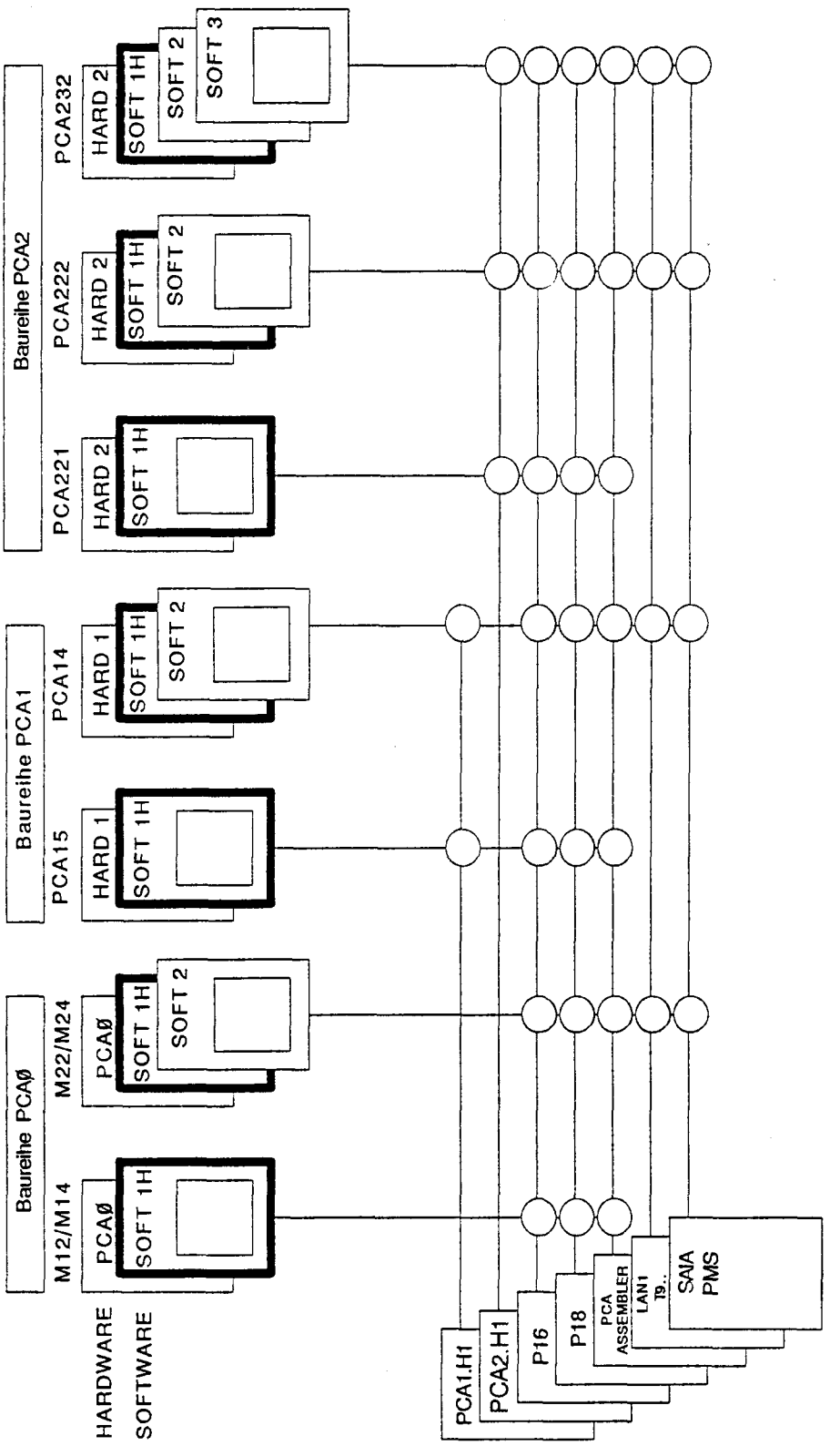
Uebersicht der SAIA°PLC, Systemfamilie PCA
Kurzüberblick über die zur Verfügung stehenden Handbücher
Uebersicht der Register der Familie PCA
Basis-Instruktionen der SAIA°PLC, Software-Stufe 1
Ergänzungsbefehle Software-Stufe 1H

Übersicht der SAIA® PLC, Systemfamilie PCA

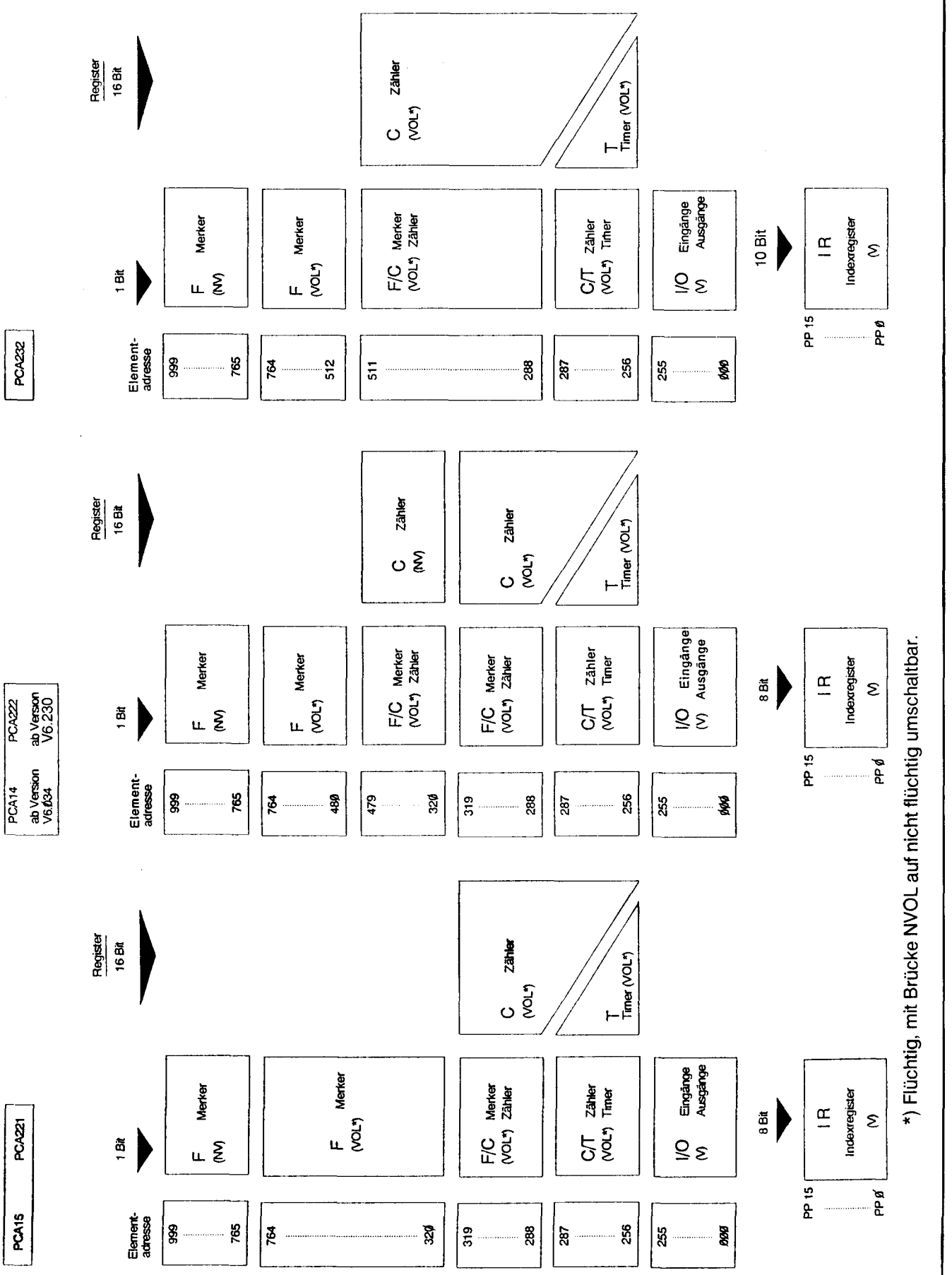
System PCA

	Baureihe PCAØ	Baureihe PCA1	Baureihe PCA2
Software-Stufe 3 Software-Stufe 2 + 32 Wortbefehle für - Arithmetik, ± 9 Digits - Datentransfer - Wortregister			PCA232  Anwenderspeicher 8K Programmzeilen + 8K Textcharakter + 8K Byte Daten  256 bzw.512 E/A
Software-Stufe 2 Software-Stufe 1H + Datenschnittstelle serielle + Datum-Uhr + Datenregister + Parameterbefehle (Soft-Interrupt, FIFO, PID) 	Standard - + OEM-Ausführungen PCAØ.M22  max. 32 E/A PCAØ.M24  max. 64 E/A Anwenderspeicher max. 4K Programmzeilen max. 4K Textcharakter/Daten	PCA14 PCA141  32(56) PCA147  64(112) PCA147 + ..C45  128(224) E/A Anwenderspeicher max. 8K Programmzeilen max. 8K Textcharakter/Daten	PCA222   256 bzw. 512 E/A Anwenderspeicher max. 8K Programmzeilen max. 8K Textcharakter/Daten
Software-Stufe 1H 32 Basis-Befehle, u.a. für - Zeit- und Zählfunktionen - Parallel- und Unterprogramme - Indexierung usw. 2Ø Zusatzbefehle für - Arithmetik - Datentransfer - Check-Sum	Standard-Ausführungen PCAØ.M12  24/32 E/A PCAØ.M14  48/64 E/A Anwenderspeicher max. 4K Programmzeilen	PCA15 PCA151  32(56) PCA156  64(112) PCA157 + ..C45  128(224) E/A Anwenderspeicher max. 4K Programmzeilen	PCA221   256 bzw.512 E/A Anwenderspeicher max. 8K Programmzeilen

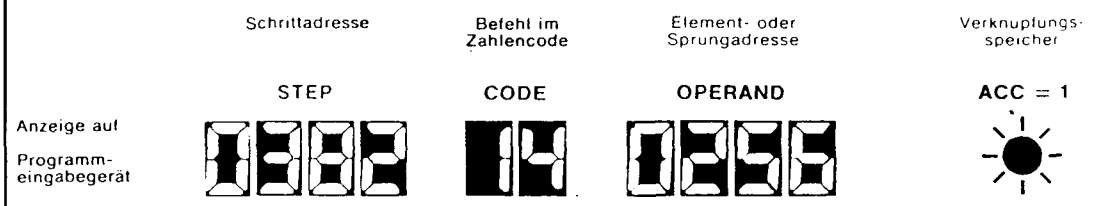
Kurzüberblick über die zur Verfügung stehenden Handbücher



Übersicht der Register der Familie PCA



Basis-Instruktionen der SAIA°PLC, Software-Stufe 1



	Zahlen-code	Mnemo-code	Befehl englisch	Beschreibung
Logikbefehle	Ø1	STH	Start High	Beginn einer Verknüpfung mit Element abgefragt auf High
	Ø2	STL	Start Low	Beginn einer Verknüpfung mit Element abgefragt auf Low
	Ø3	ANH	AND High	UND-Verknüpfung zwischen ACCU und Element abgefragt auf High
	Ø4	ANL	AND Low	UND-Verknüpfung zwischen ACCU und Element abgefragt auf Low
	Ø5	ORH	OR High	ODER-Verknüpfung zwischen ACCU und nächstem Element abgefragt auf High
	Ø6	ORL	OR Low	ODER-Verknüpfung zwischen ACCU und nächstem Element abgefragt auf Low
	Ø7	XOR	Exclusive OR	Exklusiv-ODER-Verknüpfung des Elementes
	Ø8	NEG	Negate Accu	Negiere Accu (Verknüpfungsergebnis)
	Ø9	DYN	Dynamic Control	Dynamisierung der Verknüpfung (Accu wird nur im 1. Zyklus beeinflusst)
Schaltbefehle	1Ø	OUT	Set Output with Status of Accu	Setze Ausgang oder Merker mit Inhalt des Accu
	11	SEO	Set Output	Setze Ausgang oder Merker speichernd
	12	REO	Reset Output	Setze Ausgang oder Merker zurück
	13	COO	Complement Output	Frage den Ausgang oder Merker ab und setze ihn umgekehrt
Zeit- und Zählbefehle	14*	STR*	Set Timer	Setze Zeitglied auf vorgewählten Wert und starte es
	15*	SCR*	Set Counter	Setze Zähler auf vorgewählten Wert
	17	INC	Increment Counter	Erhöhe den Stand des Zählers um 1
	18	DEC	Decrement Counter	Erniedrige den Stand des Zählers um 1
Sprungbefehle	2Ø	JMP	Unconditional Jump	Unbedingter Sprung auf Schrittadresse
	21	JIO	Jump if Accu is One	Springe, wenn Accu = 1 auf Schrittadresse
	22	JIZ	Jump if Accu is Zero	Springe, wenn Accu = Ø auf Schrittadresse
	23	JMS	Jump to Subroutine	Springe in Unterprogramm
	24	RET	Return from Subrout.	Rücksprung vom Unterprogramm
Wartebefehle	25	WIH	Wait if High	Warte, solange Element High
	26	WIL	Wait if Low	Warte, solange Element Low
Hilfsbefehle	ØØ	NOP	No Operation	Keine Operation
	19	SEA	Set Accu	Setze Accu = 1
	16	SEI	Set Index	Setze Indexregister auf vorgewählten Wert
	27	INI	Increment Index	Erhöhe das Indexregister 1
	28	DEI	Decrement Index	Erniedrige das Indexregister 1
	29**	PAS**	Program Assignment	Zuweisung des Parallelprogrammes
	3Ø	DOP	Display Operand	Anzeige eines Operanden
	31	DTC	Display Timer or Counter	Anzeige des Zeit- oder Zählerstandes

* zweizeilige Befehle (zweite Zeile enthält vorgewählten Wert)
** zweizeiliger Befehl (zweite Zeile enthält Anfangsadresse des Programmes)

Ergänzungsbefehle Software-Stufe 1H

	Mnemo-Code	Zahlen-Code	Befehl englisch	Beschreibung
Transfer-Befehle	STR	14	Set Timer	
	SCR	15	Set Counter	
		19	2. Zeile	Einlesen 5 x 4 Bit BCD
		2Ø		Ausgeben 5 x 4 Bit BCD
		21		Ausgeben 8 Bit binär
		22		Ausgeben 12 Bit binär
		23		Ausgeben 16 Bit binär
		24		Einlesen 8 Bit binär
		25		Einlesen 12 Bit binär
		26		Einlesen 16 Bit binär
		31		Transfer Zähler → Zähler bzw. Indexregister → Zähler
Arithmetik-Befehle	SCR	15	Set Counter	
		27	2. Zeile	Addiere +
		28		Subtrahiere -
		29		Multipliziere x
		3Ø		Dividiere :
Indexier-Befehle	SEI	16	Set Index	Setze Indexregister auf vorgewählten Wert
	INI	27	Increment-Index	Erhöhe } das Indexregi-
	DEI	28	Decrement-Index	Erniedrige } ster um 1
	Mnemo-Code	Zahlen-Code	Operand	Beschreibung
Spezial-Befehle (diese Befehle sind zweizeilig)	PAS	29	18	Veränderung der Anzahl aktiver Parallelprogramme
	PAS	29	30 ... 38	Check-Sum

Wie aus der Übersicht der vorangehenden Seiten ersichtlich wird, ist der Instruktionssatz der gesamten SAIA°PCA-Familie in 4 Stufen vollkommen aufwärtskompatibel.

Die Stufe (1), zu welcher die 32 Basis-Instruktionen gehören, war den früheren Typenreihen PCA13 und PCA21Ø vorbehalten.

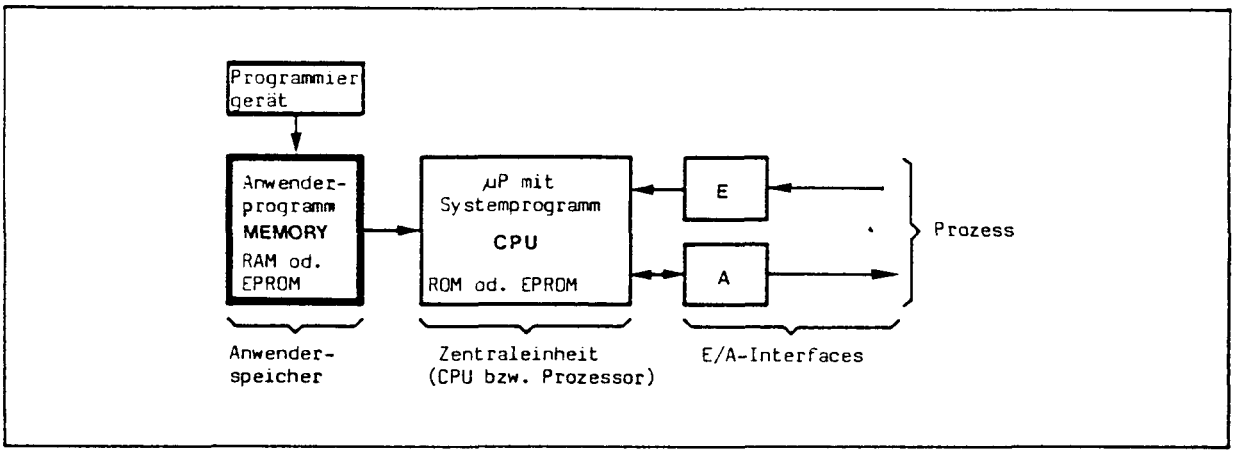
Heute heisst die niedrigste Stufe (1H).
Sie stellt mit den Baureihen PCAØ, PCA15 und PCA221 zusätzlich 2Ø komfortable Befehle zur Verfügung.
Diese Intelligenz-Stufe ist Gegenstand des vorliegenden Handbuches.

Mit lediglich 5 Befehlen können bereits einfache Verknüpfungsprogramme geschrieben werden. Bei voller Ausschöpfung der Stufe (1H) lassen sich jedoch Zeit- und Zählfunktionen, Parallel- und Unterprogramm-Technik sowie Arithmetikfunktionen ausführen.

Die höheren Intelligenz-Stufen (2) und (3) der Baureihen PCA14, PCA222 und PCA232 enthalten alle diese Grundfunktionen, so dass Programme der Stufe (1H) bei steigenden Ansprüchen jederzeit weiter benutzt werden können.

TEIL D EINFÜHRUNG

D 1 Allgemeines



Wie bereits in der Einleitung zur Hardware aufgezeigt, sind die Eigenschaften der Zentraleinheit durch das zum µP-System gehörende Systemprogramm bestimmt. Mit dem Systemprogramm ist die ganze Charakteristik der PLC unveränderlich festgelegt. Für den Anwender besteht kein Zugriff zum Systemprogramm. Die individuelle Anpassung an die verschiedenen Prozessbedingungen erfolgt über das Anwenderprogramm.

Dieses Anwenderprogramm wird in der problemorientierten Sprache der SAIA°PLC in den Anwenderspeicher geschrieben. Seine Programmierung erfolgt durch verschiedene zur Verfügung stehende Programmiergeräte.

Die Zentraleinheit (auch Prozessor genannt) hat nun die Fähigkeit, das Anwenderprogramm zu "lesen" und die darin enthaltenen Anweisungen auszuführen, wie z.B. Eingangszustände abzufragen, diese zu verknüpfen und das Ergebnis auf Ausgänge zu übertragen und damit den Prozess in der gewünschten Weise zu steuern.

Je nach Aufgabenstellung kann das Programm beliebig nach einer der folgenden Steuerungsbeschreibungen erstellt werden:

- Kontakt plan (Stromlaufplan)
- Logigplan (Signallaufplan)
- Flussdiagramm (Ablaufplan)
- Funktionsplan nach DIN 44719 oder nach GRAFCET (Schrittsteuerung)

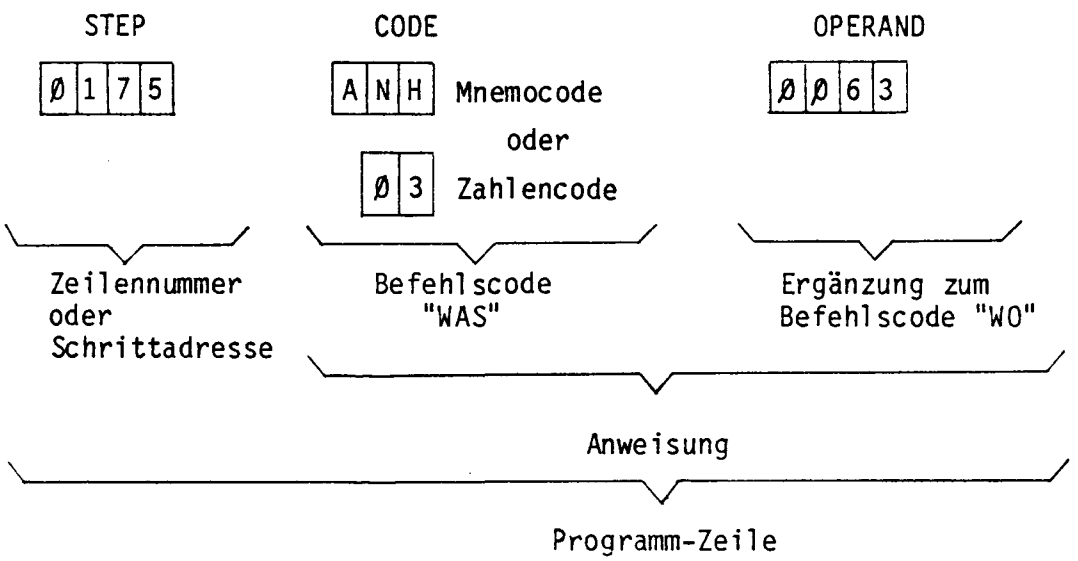
Die verschiedenen Programmierungsarten können auch miteinander kombiniert werden.

D 2 Die Programmzeile

Jede Anweisung im Anwenderprogramm besteht aus 1 Programmzeile (in gewissen Fällen aus 2 oder 1Ø Zeilen). Eine Zeile enthält ausser der Zeilennummer oder Schrittadresse (STEP) auch den Befehlscode (CODE) und den Operanden (OPERAND). Im Befehlscode wird ausgesagt "WAS" für ein Befehl auszuführen ist, und im Operanden wird festgelegt "WO" dieser Befehl zu wirken hat.

Jede Programmzeile besteht aus 16 Bit, sodass in einem Anwenderspeicher von 16K Bit (z.B. EPROM 2716) 1K = 1Ø24 Programmzeilen abgespeichert werden können.

Aufbau der Programm- bzw. Befehlszeile:



STEP — Mit der Zeilen-Nr. wird der Platz der Anweisung im Anwender-Speicher festgelegt. Dezimale Numerierung von Ø...2Ø47 (2K) bzw. 4Ø95 (4K).

CODE — Je nach Programmiergerät kann der Befehlscode als 3-stelliger Mnemocode oder als Zahlencode von Ø bis 31 eingegeben werden; der Mnemocode ist die Abkürzung des englischen Ausdruckes für den entsprechenden Befehl. Er ist daher leicht im Gedächtnis zu behalten und auch international verständlich.

OPERAND — Hier wird die Adresse eines Elementes (Eingang, Ausgang, Timer, Zähler oder Merker) oder die Zieladresse (Zeilen-Nr.) bei Sprungbefehlen eingesetzt.

Zeit- und Zählbefehle bestehen aus 2 Programmzeilen. In der zweiten Zeile erscheint im Operand der entsprechende Zeit- bzw. Zählwert.

D 3 Die Operanden

Wie in der vorgängigen Definition bereits erwähnt, handelt es sich bei den Operanden um sog. Element-Adressen oder Schritt-Adressen (Zeilen-Nr.).

D 3.1 Element-Adressen

Alle adressierbaren Elemente (Eingänge, Ausgänge, Timer, Zähler, Merker und Haftspeicher) sind dezimal numeriert im Bereich von Ø...999.

- Die Eingänge und Ausgänge sind in Form von Interface-Modulen auf das Grundgerät der PLC steckbar. Ihre Adressierung erfolgt entweder durch den Montageplatz (PCAØ und PCA1) oder durch einen DIL-Schalter (PCA2).

  Der Adressbereich kann beliebig, jedoch nur entweder durch ein Eingangsmodul oder durch ein Ausgangsmodul belegt werden. (Eine Ausnahme bildet hier der Rack-Einschub PCA2.C3Ø, mit welchem 256 E + 256 A = 512 E + A erreicht werden können).

  Die Signalzustände von Eingängen können lediglich abgefragt werden.

  Die Ausgänge lassen sich setzen (einschalten) und rücksetzen (ausschalten) aber auch auf ihren Signalzustand abfragen. (Ausnahmen bei gewissen Modulen).

- Die Timer (Zeitglieder) und Zähler sind programmierbare Register. Sie befinden sich hardwareseitig im CPU-Modul. Sie können alternativ als Timer oder als Zähler eingesetzt werden.

- Die Merker und Haftspeicher sind Speicherzellen, welche wie Ausgänge behandelt werden, d.h. sie können gesetzt bzw. rückgesetzt aber auch auf ihren Signalzustand abgefragt werden. Merker und Haftspeicher eignen sich demnach gut zum Abspeichern irgendwelcher Informationen.

  Hardwareseitig befinden sie sich ebenfalls auf dem CPU-Modul in Form eines separaten RAM-Speichers. Da dieser RAM Speicher mit einer Pufferbatterie versehen ist, behalten die 235 Haftspeicher ihre Information auch bei Spannungsausfall für mehr als 1 Monat bei. Die 477 Merker sind zwar auch batteriegepuffert, werden jedoch beim Einschalten der PLC in den Signalzustand "L" zurüchgesetzt. Dadurch wirken sie als nullspannugsrückstellende Merker.

  Durch Einhängen der Hardwarebrücke NVOL auf dem CPU-Modul kann erreicht werden, dass alle 712 Speicherzellen und alle Timer, Zähler nullspannungssichere Haftspeicher werden.

D 3.2 Schritt-Adressen

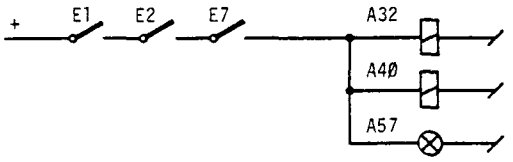
Die zweite Art von Operanden sind die sogenannten Schritt-Adressen oder Zeilen-Nr. Sie werden in Zusammenhang mit Sprung-Befehlen benötigt. In dieser Kombination kann festgelegt werden, wohin im Anwenderprogramm gesprungen werden soll. Um den ganzen Anwenderspeicher abzudecken, kommen als Operand alle Schritt-Adressen von Ø bis 2Ø47 in Frage.

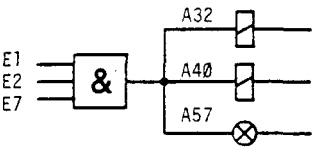
Durch 2-zeilige Sprungbefehle bei den leistungsfähigen CPUs (ab Intelligenzstufe (1H)) wird der Bereich für Schrittadressen auf 8191 (8 K) erweitert.

D 4 Weitere Definitionen

D 4.1 Die Verknüpfungszeile

Beispiel nach Kontakt- bzw. Logikplan:





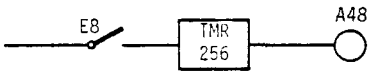
Programm

```
OUT  51
STH   1
ANH   2
ANH   7
OUT  32
OUT  4Ø
OUT  57
STH   9
```

} Verknüpfungszeile

Eine Verknüpfungszeile ist ein Ausschnitt aus einem Programm und besteht aus mehreren Befehlszeilen. Sie ist eine in sich abgeschlossene Verknüpfung von Elementen. Sie beginnt normalerweise mit einem Startbefehl (STH oder STL). Die Verknüpfung wird als erfolgreich bezeichnet, wenn das Endergebnis = 1 ist, d.h. wenn in unserem Beispiel die Ausgänge A32, A4Ø und A57 aktiviert (eingeschaltet) werden.

Beispiel mit einem Zeitglied:



Programm

```
STH    8
STR  256
 ØØ   5Ø
STH  256
OUT   48
```

} 1. Verknüpfungszeile

} 2. Verknüpfungszeile

Wie das obenstehende Programm zeigt, besteht obige Funktion bereits aus 2 Verknüpfungszeilen. In der 1. Verknüpfungszeile wird der Eingang E 8 abgefragt und aufgrund seines Signalzustandes das Zeitglied gesetzt (vergleichbar dem Steuerkreis eines Zeitrelais). In der 2. Verknüpfungszeile wird der Signalzustand des Zeitgliedes auf den Ausgang A48 übertragen (vergleichbar mit dem Lastkreis eines Zeitrelais).

D 4.2 Der Verknüpfungsspeicher = Accumulator = ACCU

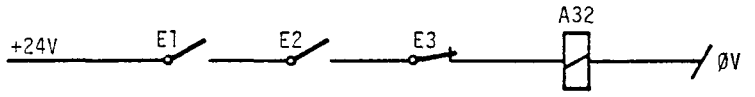
Dieser Speicher sitzt ebenfalls in der CPU und besteht aus einer einzigen Speicherstelle (1 Bit), welche die logischen Zustände Ø oder 1 annehmen kann.

Um eine vollständige Verknüpfung bis zu einem Aktionsbefehl abzuarbeiten, muss das jeweilige Verknüpfungs-Zwischenergebnis im ACCU festgehalten werden. Am Ende der Verknüpfung liegt das Endergebnis im ACCU (Ø oder 1) vor. Aufgrund dieses Resultates wird das entsprechende Element (z.B. ein Ausgang) dann nicht aktiviert (ACCU = Ø) bzw. aktiviert (ACCU = 1).

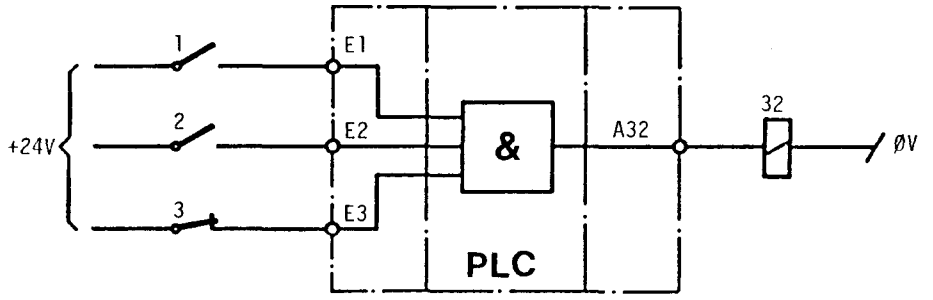
Durch das im ACCU gespeicherte Ergebnis können nachfolgende Anweisungen auch übersprungen werden (siehe z.B. Sprungbefehle).

D 4.3 Schliesser/Oeffner bzw. High/Low

Eine Kontaktverknüpfung wie im untenstehenden Schema gezeigt, wird als Verdrahtung genau nach Schema ausgeführt:

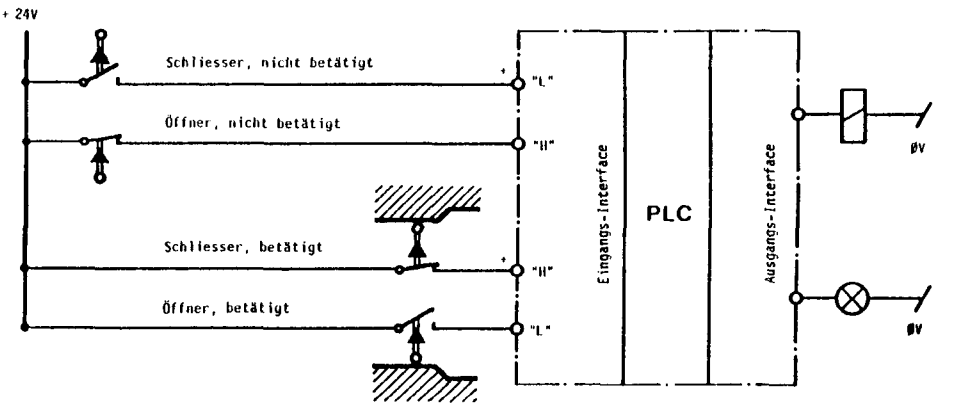


Wird dieses Problem mit einer PLC gelöst, so wird jeder Kontakt auf einen Eingang geführt. Die Verknüpfung erfolgt dann im Prozessor der PLC und nicht mit der Verdrahtung. Eine PLC-gerechte Darstellung obiger Verknüpfung wird in der folgenden Figur gezeigt.



Eine PLC kann nicht unterscheiden, ob an ihren Eingängen Schliesser oder Oeffner angeschlossen sind. Sie unterscheidet aber, ob am Eingang ein High-Signal (H) oder ein Low-Signal (L) ansteht. Diese Signalpegel sind für jedes Eingangsinterface im Teil Hardware genau definiert. Für einen 24V-Eingang im Quellbetrieb (Normalfall) kann vereinfachend gesagt werden:

- Liegt am Eingang +24V an ----------> so ergibt sich Signalzustand "H" = High
- Liegt am Eingang ØV an ----------> so ergibt sich Signalzustand "L" = Low



Zur Programmierung der PLC kann aber dennoch die verbreitete Darstellung mit Kontaktsymbolen verwendet werden, sofern die nachfolgenden Regeln beachtet werden. Für Quellbetrieb, d.h. Kontakt an +24V gilt:

- Hilft ein Kontakt in geschlossenem Zustand mit, dass eine Verknüpfung erfolgreich wird, so muss der betreffende Kontakt nach High abgefragt bzw. verknüpft werden (Normalfall).

- Hilft ein Kontakt in offenem Zustand mit, dass eine Verknüpfung erfolgreich wird, so muss dieser Kontakt nach Low abgefragt bzw. verknüpft werden.

Gleiche Ueberlegungen gelten ausser für mechanische Kontakte auch für Näherungsschalter und Lichtschranken.

Man ist nun versucht, der Einheitlichkeit halber, generell nur Schliesser einzusetzen. Dies ist auch für ca. 90% der Fälle richtig. Bei einigen besonderen Anwendungen ist jedoch der Einsatz von Oeffnern aus Sicherheitsgründen unerlässlich. Vereinfacht kann gesagt werden:

- Das Starten eines Vorganges muss immer durch "An Spannung legen" vorgenommen werden, während das Stillsetzen immer durch Abschalten der Spannung einzuleiten ist.

Um den vorangegangenen Ueberlegungen Rechnung zu tragen, sind die Abfrage- und Verknüpfungsbefehle immer paarweise vorhanden:

STH	STL
ANH	ANL
ORH	ORL
(WIH)	(WIL)
(JIO)	(JIZ)

Dies bedeutet, dass wir die Eingangssignale jeweils nach "H" oder "L" abfragen können. Stimmt die Art der Abfrage mit dem wirklichen Signalzustand am Eingang überein, so wird dies als logisch "1" im ACCU verarbeitet.

Signalzustände gelten nicht nur für Eingänge, sondern für alle adressierbaren Elemente:

Eingänge	"H":	+24V steht an (Standard, Quellbetrieb)
	"L":	0V steht an (Standard, Quellbetrieb)
Ausgänge	"H":	Ausgang gesetzt = Transistor leitend
	"L":	Ausgang nicht gesetzt = Transistor sperrt
Merker/Haftspeicher	"H":	Gesetzt
	"L":	Nicht gesetzt bzw. rückgesetzt
Timer	"H":	Timer wurde gesetzt und läuft ab
	"L":	Timer wurde nicht gesetzt oder ist abgelaufen
Zähler	"H":	Zähler wurde gesetzt und Register-Inhalt > 0
	"L":	Zähler wurde nicht gesetzt oder Register-Inhalt = 0

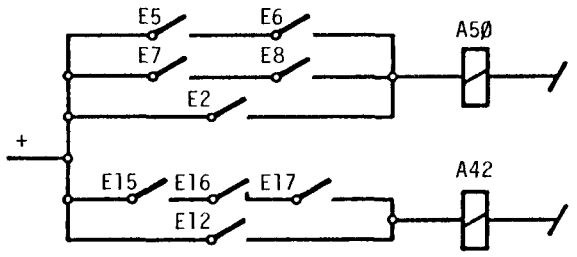
D 5 Programmierungsarten

Eine speicherprogrammierbare Steuerung ergibt ein Höchstmass an Flexibilität in der Auslegung und Anpassung des Programmes. Aehnlich aber wie bei der traditionellen Steuerung enthebt die PLC den Steuerungsbauer jedoch nicht der wichtigen Aufgabe vorgängig eine klare Beschreibung der Steuerungsaufgabe vorzunehmen. Je nach Art des Prozesses und auch der Vorliebe des Steuerungsbauers wird diese Beschreibung auf verschiedene Weise erfolgen.

Durch den vielseitigen Befehlssatz der SAIA°PLC sind die unterschiedlichsten Steuerungsbeschreibungen als Vorlage zu einem PLC-Programm geeignet.

D 5.1 Programmierung nach Kontaktplan (Stromlaufplan)

Vorlage:



Programm (Anweisungsliste):

```
ADDR   NC   MNC   OPRD
 10    01   STH     5
 11    03   ANH     6
 12    05   ORH     7
 13    03   ANH     8
 14    05   ORH     2
 15    10   OUT    50
 16    01   STH    15
 17    03   ANH    16
 18    03   ANH    17
 19    05   ORH    12
 20    10   OUT    42
 21    20   JMP    10
```

Merkmale:

----> Benötigte Befehle STH, STL, ANH, ANL, ORH, ORL, OUT.

- Die Verknüpfung wird durch Kontaktsymbole dargestellt. Die Ausgänge übernehmen die Ergebnisse der jeweils vorangegangenen UND/ODER-Verknüpfungen.
- Alle Programmteile werden ständig zyklisch durchlaufen. Wir nennen das ein reines Umlaufprogramm, weil es keine Warte- und keine bedingten Sprungbefehle enthält.

Vorteile:

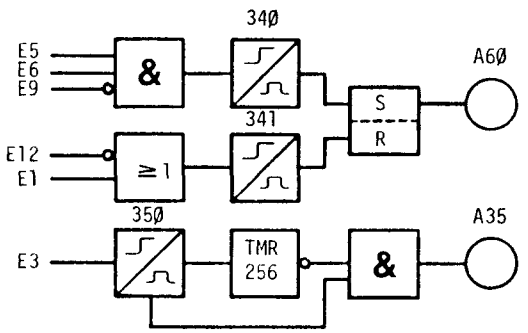
- Einfache Programmerstellung mit nur 7 Befehlen für Anhänger des Kontaktplans.
- Gut geeignet für Ueberwachungsaufgaben.
- Verschiedene Verknüpfungszeilen können praktisch beliebig aneinander gereiht werden.

Nachteile:

- Schlecht geeignet für sequentielle Prozesse (Ablaufsteuerungen), da alle jeweils nicht aktiven Teile verriegelt werden müssen.
- Komplexere Steuerungen werden im Kontaktplan unübersichtlich und schwerfällig.
- Beschränkter Befehlssatz.
- Bei grossen Programmen lange Reaktionszeit.

D 5.2 Programmierung nach Logikplan (Signallaufplan)

Vorlage:



Programm (Anweisungsliste):

ADDR	NC	MNC	OPRD	
30	01	STH	5	
31	03	ANH	6	
32	04	ANL	9	
33	09	DYN	340	
34	11	SEO	60	RS
35	02	STL	12	
36	05	ORH	1	
37	09	DYN	341	
38	12	REO	60	
39	01	STH	3	
40	09	DYN	350	
41	14	STR	256	
42	00	00	20	Timer
43	02	STL	256	
44	03	ANH	350	
45	10	OUT	35	
46	20	JMP	30	

Merkmale:

-----> Zusätzlich verfügbare Befehle SEO, REO, COO, STR, SCR, NEG, DYN usw.

- Die Verknüpfung erfolgt über funktionsorientierte Logiksymbole, wobei der Signalfluss in ähnlicher Weise verfolgt werden kann wie beim Kontaktplan.
- Alle Programmteile werden ständig zyklisch durchlaufen (reines Umlaufprogramm).

Vorteile:

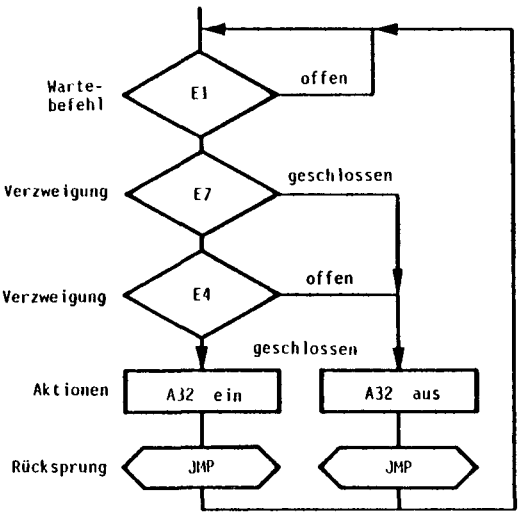
- Die Ausgänge können speichernd oder nicht speichernd gesetzt werden.
- Gut geeignet für Ueberwachungsaufgaben und reine Logikprogramme.
- Da ein grosser Teil des Befehlssatzes verwendet werden kann, bleiben komplexere Probleme übersichtlich.
- Verschiedene Verknüpfungszeilen können mit einer gewissen Freiheit aneinandergereiht werden.

Nachteile:

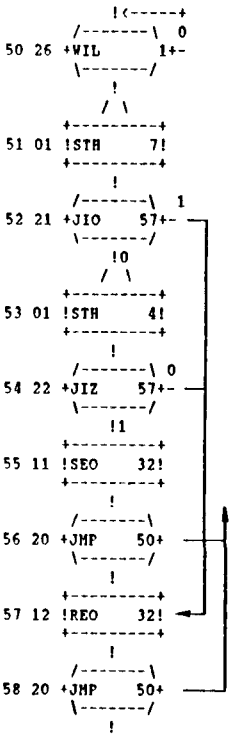
- Schlecht geeignet für sequentielle Prozesse (Ablaufsteuerungen), da alle jeweils nicht aktiven Teile verriegelt werden müssen.
- Bei grossen Programmen lange Reaktionszeiten.

D 5.3 Programmierung nach Flussdiagramm (Ablaufplan)

Vorlage:



Programm in Flussidagramm:



Programm (Anweisungsliste):

ADDR	NC	MNC	OPRD
50	26	WIL	1
51	01	STH	7
52	21	JIO	57
53	01	STH	4
54	22	JIZ	57
55	11	SEO	32
56	20	JMP	50
57	12	REO	32
58	20	JMP	50

Merkmale:

------> Besondere Befehle WIH, WIL, JIO, JIZ

- Es können Warteschleifen gebildet werden, welche den Prozessor solange in dieser Schleife halten, bis die Weiterlaufbedingungen erfüllt sind.
- Es sind Verzweigungen mit bedingten Sprungbefehlen möglich.
- Diese Programmierungsart ist gut geeignet für sequentielle Prozesse (Ablaufsteuerung).

Vorteile:

- Einfache Programmerstellung direkt nach der Ablaufbeschreibung des sequentiellen Prozesses.
- Diese Programmierungsart wird auch von den Prozessleuten gut verstanden.
- Infolge der Warteschleifen läuft das Programm nur so schnell ab, wie der Prozess selber. Daraus ergeben sich einfache Inbetriebnahme und rasche Fehlerdiagnose.
- Die Reaktionszeiten sind äusserst kurz. Sie werden praktisch nur durch die Verzögerung des Eingangs-Interfaces bestimmt.

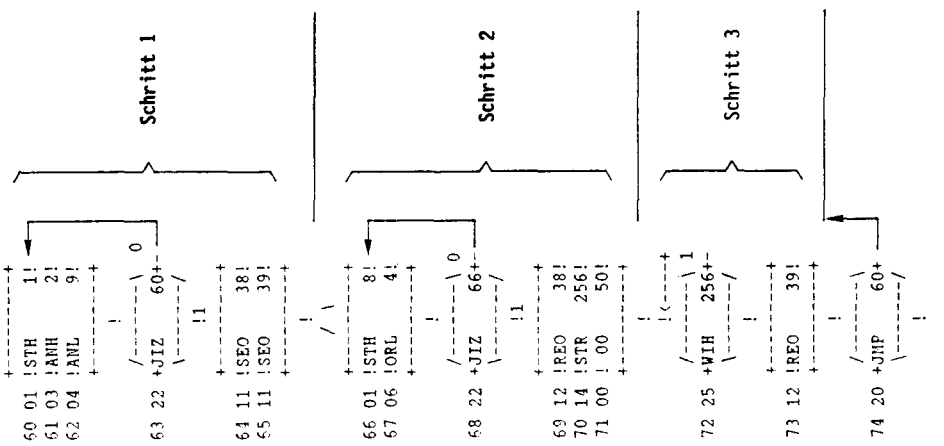
Nachteile:

- Ueberwachungsfunktionen müssen getrennt in Parallelprogrammen untergebracht werden.

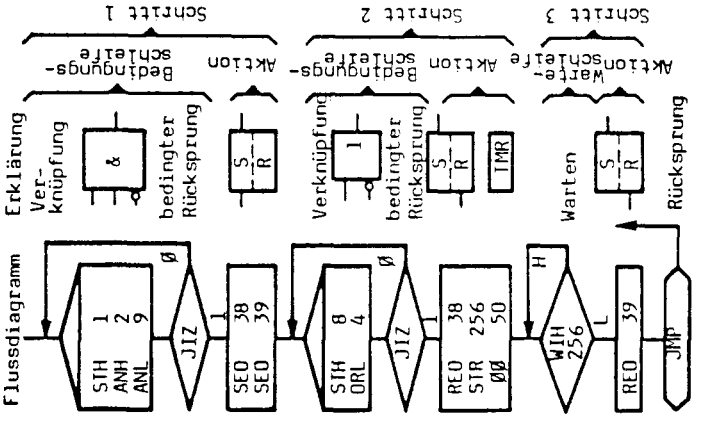
D 5.4 Kombinierte Programmierung nach Flussdiagramm/Logikplan bzw. nach Graftec/Funktionsplan

Die universelle Programmiersprache der SAIA°PLC erlaubt es, durch kombinierte Anwendung der Befehle praxisbezogene Schrittprogramme zu erstellen.

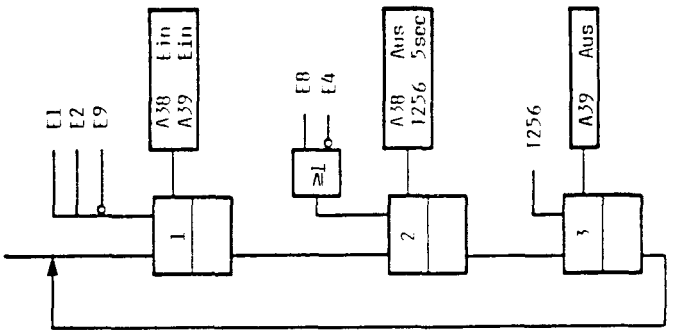
Dokumentation nach Flussdiagramm



Vorlage nach Flussdiagramm



Vorlage nach DIN - Funktionsplan bzw. nach GRAFCET

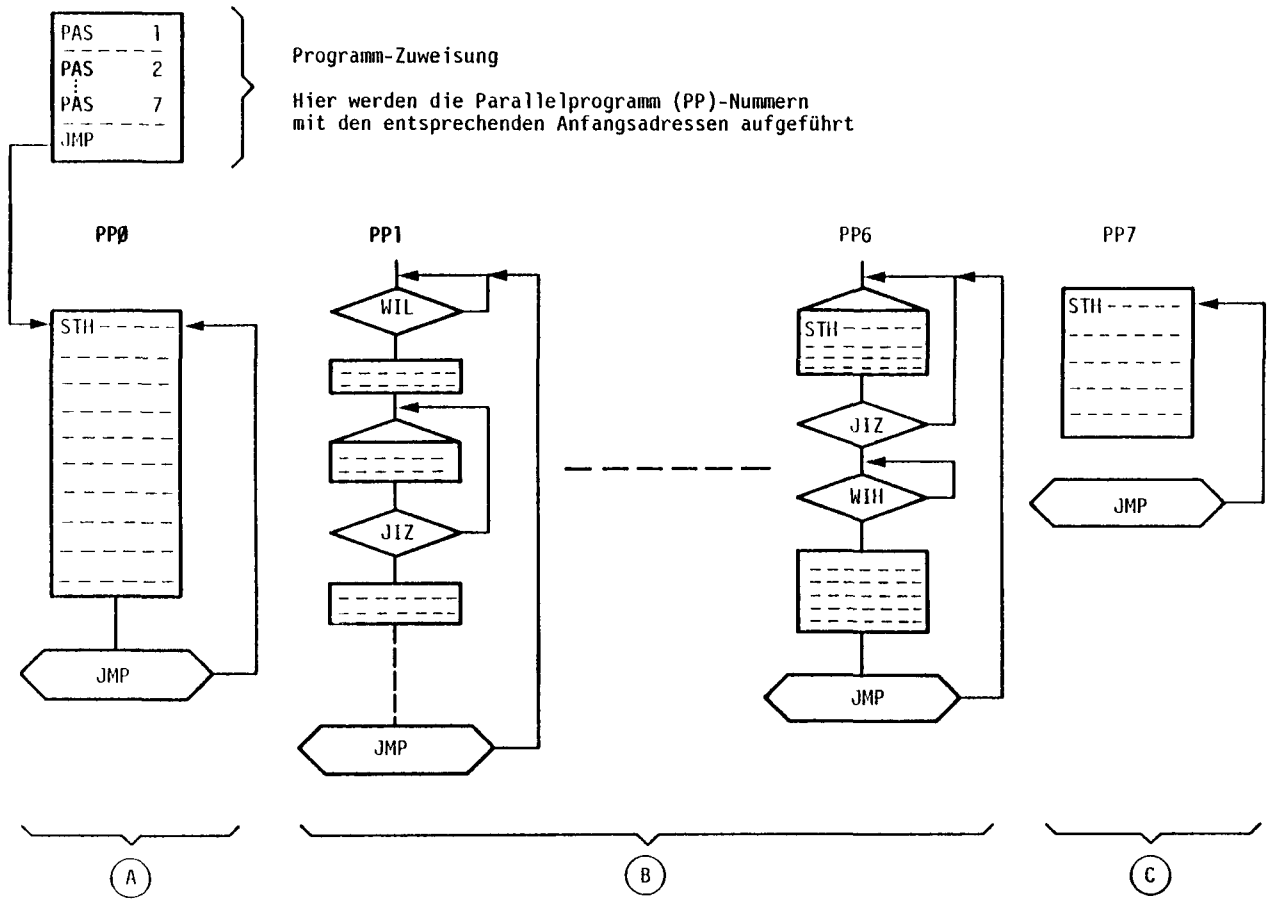


Merkmale: Durch den Einsatz von Logikbefehlen wird die Flexibilität weiter vergrössert.

D 5.5 Programmierung mit Parallelprogrammen

Optimale Bedingungen werden durch den Einsatz von Parallelprogrammen erreicht. Es können bis zu 16 voneinander unabhängige, asynchron ablaufende Parallelprogramme eingesetzt werden. Dies bedeutet, dass verschiedene sequentielle Abläufe einer Maschine (z.B. eines Montageautomaten) in unterschiedlichen Programmen nach Ablaufplan untergebracht werden, die schrittweise mit dem Prozessfortschritt durchlaufen werden.
Ueberwachungen und dauernd aktive Funktionen werden in einem ebenfalls parallelen Umlaufprogramm untergebracht.

Programmstruktur:



Ⓐ: PPØ (Parallelprogramm Ø) ist ein reines Umlaufprogramm (ohne Warteschleifen) mit Ueberwachungen und dauernd aktiven Funktionen.

Ⓑ: PP1 bis (z.B.) PP6 sind parallele Ablaufprogramme im Flussdiagramm von verschiedenen asynchron ablaufenden sequentiellen Funktionen.

Ⓒ: PP7 ist ein kurzes, reines Umlaufprogramm, welches einige wenige Funktionen enthält, die äusserst kurze Reaktionszeiten bedingen.

Merkmale:

-----> Befehl PAS für Programm-Zuweisung.
Übrige Merkmale siehe Programmstruktur.

Vorteile:

- Sequentielle Abläufe (mit Warteschleifen) und dauernd aktive Funktionen (Ueberwachungen, reine Logik usw.) können sauber getrennt in der ihnen gemässen Art programmiert werden.

- Es werden nur so viele Parallelprogramme zugewiesen wie benötigt werden (max. 16).

- Durch entsprechenden Software-Interrupt können sequentielle Programme jederzeit stillgesetzt oder wieder von vorne gestartet werden (Befehl PAS 18).

Nachteile:

- Um schnelle Reaktionszeiten für die dauernd aktiven Funktionen beizubehalten, ist es vorteilhaft, diese in einem separaten, kurzen Umlaufprogramm zu behandeln.

D 5.6 Programmierung mit Unterprogrammen (Subroutinen)

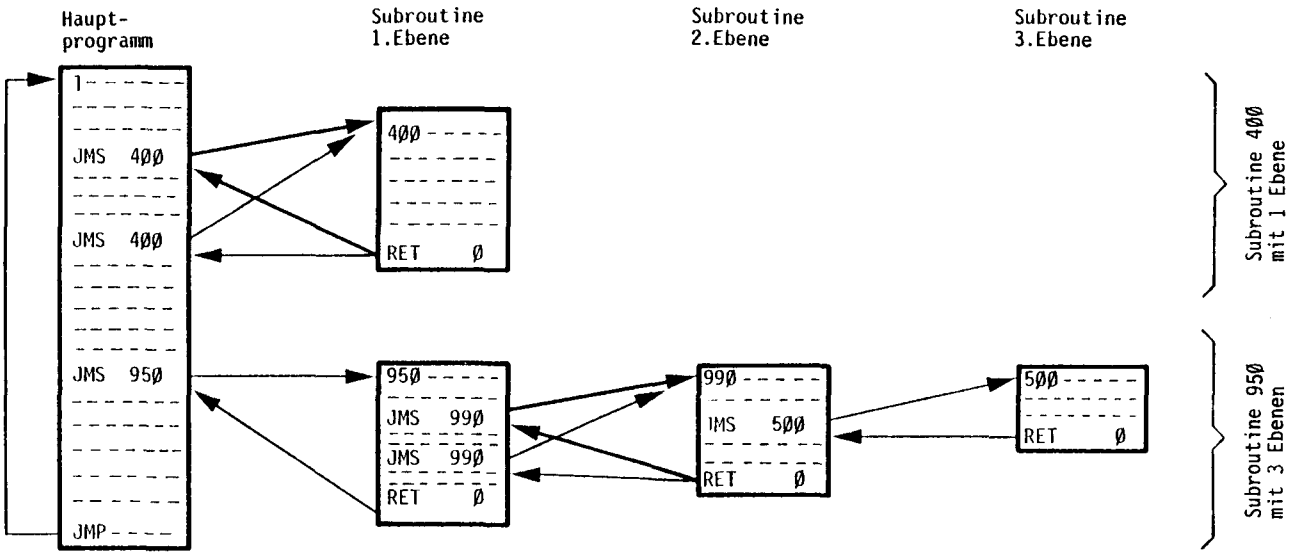
-----> Benötigte Befehle JMS und RET.

Programmteile, die oft wiederholt werden müssen, können mit Vorteil als Subroutinen geschrieben werden.

Subroutinen können in allen vorgängig beschriebenen Programmtechniken, d.h. nach Kontaktplan, Logikplan, Flussdiagramm oder unter Einsatz von Parallelprogrammen verwendet werden.

Mit Subroutinen wird viel Speicherplatz und Programmierarbeit gespart. Die Programme werden unter Verwendung von Subroutinen auch übersichtlich in Funktionsblöcke aufgeteilt (strukturierte Programmierung).

Es können beliebig viele Subroutinen angesprungen werden, wobei Verschachtelungen bis zu 3 Ebenen zulässig sind. Dabei ist lediglich darauf zu achten, dass von verschiedenen PPs nicht gleichzeitig in die gleichen Subroutinen gesprungen wird.



D 5.7 Adress-Indexierung (Reihenbehandlung)

-----> Befehle SEI, INI, DEI

Wenn mehrere Elemente (Eingänge, Ausgänge, Merker oder Timer) in gleicher Art behandelt werden müssen (z.B. in Ueberwachungsschaltungen oder bei Schieberegistern), so ergibt das üblicherweise lange Programme von geringer Verknüpfungstiefe.
Sollten z.B. durch Tastendruck alle 235 Haftspeicher zurückgesetzt werden, so würden dazu 236 Programmzeilen benötigt. Mit Adressindexierung kann die gleiche Aufgabe mit nur 5 Programmzeilen programmiert werden.

Ohne Indexierung:

```
STH     1
REO   765
REO   766
REO   767
REO   768
REO   769
REO   77Ø
   .
   .
REO   997
REO   998
REO   999
JMP
```

Mit Indexierung:

```
SEI    Ø
STH    1
REO 1765
INI  234
JIO
JMP
```

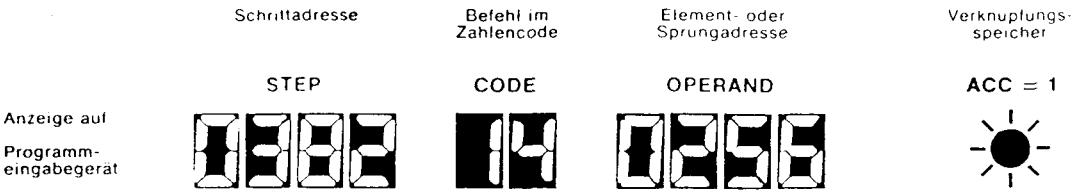
Die Indexierschleife wird 1 + 234 mal durchlaufen, wobei die Elementadresse (versehen mit 1ØØØ) jedesmal um 1 erhöht wird.

Notizen:

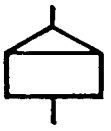
TEIL E BEFEHLSSATZ UND PROGRAMMIERUNG

TEIL E BEFEHLSATZ UND PROGRAMMIERUNG

E 1 Abfrage- und Verknüpfungsbefehle



	Zahlen-code	Mnemo-code	Befehl englisch		Beschreibung	
Logikbefehle	Ø1	STH	Start High	\	Beginn einer Verknüpfung	/ High
	Ø2	STL	Start Low	/	mit Element abgefragt auf	\ Low
	Ø3	ANH	AND High	\	UND-Verknüpfung zwischen ACCU	/ High
	Ø4	ANL	AND Low	/	und Element abgefragt auf	\ Low
	Ø5	ORH	OR High	\	ODER-Verknüpfung zwischen ACCU	/ High
	Ø6	ORL	OR Low	/	und nächstem Element abgefragt auf	\ Low
	Ø7	XOR	Exclusive OR		Exklusiv-ODER-Verknüpfung des Elementes	
	Ø8	NEG	Negate Accu		Negiere Accu (Verknüpfungsergebnis)	
	Ø9	DYN	Dynamic Control		Dynamisierung der Verknüpfung (Accu wird nur im 1. Zyklus beeinflusst)	



STH
STL Start- und Abfragebefehl

STH: Start High ---> Start mit Elementabfrage nach High
STL: Start Low ---> Start mit Elementabfrage nach Low

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
STH	Ø1	Elementadresse	Ø...999 (i)
STL	Ø2	Elementadresse	Ø...999 (i)

(i) = indexierbar

Jede Verknüpfungszeile beginnt mit einem Startbefehl

Vorangegangene Verknüpfungsergebnisse werden mit dem Startbefehl gelöscht. Gleichzeitig wird der Signalzustand "H" oder "L" des adressierten Elementes E, A, T, Z, M abgefragt und das entsprechende Resultat im ACCU abgelegt.

STH: Start der Verknüpfung mit Elementabfrage nach High
wenn Signal = H ---> ACCU = 1
wenn Signal = L ---> ACCU = Ø

STL: Start der Verknüpfung mit Elementabfrage nach Low
wenn Signal = L ---> ACCU = 1
wenn Signal = H ---> ACCU = Ø

Wahrheitstabelle:

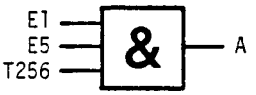
Befehl	Signalzustand des Elementes	ACCU-Inhalt nach Befehlsausführung
STH	H L	1 Ø
STL	L H	1 Ø

Allgemein: Hat das Element denselben Status nach dem es abgefragt wird, so ist das Resultat im ACCU = 1.

Kapitel F
alle Beispiele

ANH
ANL

UND-Verknüpfung



ANH: AND High ---> UND-Verknüpfung des ACCU mit abgefragtem Element nach HIGH
ANL: AND Low ---> UND-Verknüpfung des ACCU mit abgefragtem Element nach LOW

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
ANH	Ø3	Elementadresse	Ø...999 (i)
ANL	Ø4	Elementadresse	Ø...999 (i)

(i) = indexierbar

Mit dem UND-Befehl wird der Signalzustand des adressierten Elements abgefragt und das logische Abfrage-Resultat mit dem Inhalt des ACCUs UND-verknüpft.

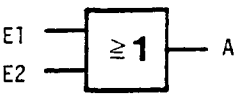
Verknüpfungstabelle:

Befehl	Signalzustand des Elementes	ACCU-Inhalt vor Befehlsausführung	ACCU-Inhalt nach Befehlsausführung
ANH	H H	1 Ø	1 Ø
	L L	1 Ø	Ø Ø
ANL	L L	1 Ø	1 Ø
	H H	1 Ø	Ø Ø

Kapitel F
Beispiele: 1, 2 und 4

ORH
ORL

ODER-Verknüpfung



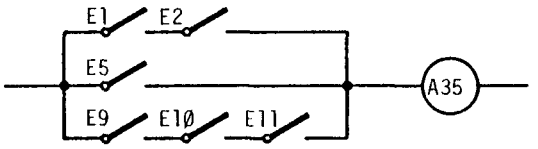
ORH: OR High ---> ODER-Verknüpfung des ACCU mit nach HIGH abgefragtem Element
ORL: OR Low ---> ODER-Verknüpfung des ACCU mit nach LOW abgefragtem Element

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
ORH	Ø5	Elementadresse	Ø...999 (i)
ORL	Ø6	Elementadresse	Ø...999 (i)

(i) = indexierbar

Mit dem ODER-Befehl werden einzelne Elemente oder Teilverknüpfungen parallel geschaltet. Die Abfrage H/L bezieht sich auf das erste Element der Teilverknüpfung.




```
STH  1   } 1. Teilverknüpfung
ANH  2   }
ORH  5     2. Teilverknüpfung
ORH  9   }
ANH  1Ø  } 3. Teilverknüpfung
ANH  11  }
OUT  35
```

Die Gesamtverknüpfung beginnt mit einem Start (STH oder STL). Jede weitere parallele Teilverknüpfung beginnt mit einem ODER (ORH oder ORL).

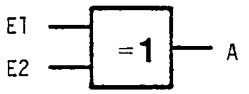
Wird eine parallele Teilverknüpfung als erfolgreich erkannt (ACCU = 1), so haben die logischen Zustände der nachfolgenden Teilverknüpfungen keinen Einfluss mehr auf das Resultat dieser Gesamtverknüpfung.
Aus obigem Beispiel geht auch hervor, dass der ODER-Befehl Priorität gegenüber UND besitzt.

Verknüpfungstabelle:

Befehl	Ergebnis der Teilverknüpfung		ACCU nach Befehlsausführung
	erste	nächste	
OR	1 1	1 Ø	1 1
	Ø Ø	1 Ø	1 Ø

Kapitel F
Beispiele: 1, 2

XOR Exklusiv ODER-Verknüpfung



XOR: Exclusive OR ---> Exklusiv ODER-Verknüpfung des ACCU mit adressiertem Element

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
XOR	Ø7	Adresse des zu vergleichenden Elementes	Ø...999 (i)

(i) = indexierbar

Mit dem XOR-Befehl können die Signalzustände zweier Elemente miteinander verglichen werden. Sind beide gleich, so ist der ACCU-Inhalt Ø; bei Ungleichheit 1. Genauer gesagt erfolgt der Vergleich des adressierten Elementes mit dem ACCU-Inhalt gemäss Verknüpfungstabelle.

Verknüpfungstabelle:

Befehl	Signalzustand des Elementes	ACCU-Inhalt vor Befehlsausführung	ACCU-Inhalt nach Befehlsausführung
XOR	H	1	Ø
	H	Ø	1
	L	1	1
	L	Ø	Ø

Kapitel F
Beispiel: 3

NEG Negiere ACCU

NEG: Negate ACCU ---> Umkehrung des ACCU-Inhaltes

Befehlsformat:

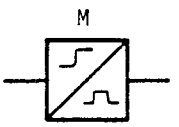
Befehlscode		Operand
Mnemo-code	Zahlen-code	
NEG	Ø8	Ø

Wahrheitstabelle:

Befehl	ACCU-Inhalt	
	vor Befehlsausführung	nach Befehlsausführung
NEG	1	Ø
	Ø	1

Mit NEG wird der Inhalt des ACCU umgekehrt.

Kapitel F
Beispiel: 3



DYN Dynamisierung

DYN: Dynamic control ---> Flankentriggerung oder Dynamisierung eines Signals bzw. einer Verknüpfung

Befehlsformat:

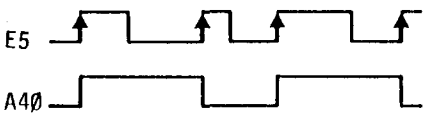
Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
DYN	Ø9	Adresse des Flanken-Merkers	288...999 (i)

(i) = indexierbar

Der DYN-Befehl bewirkt eine dynamische Ansteuerung des nachfolgenden Elementes bei erfolgreicher, vorangegangener Verknüpfung. Von einem Dauersignal wird nur die ansteigende Signalflanke ausgewertet. Anders gesagt: wechselt der Inhalt des ACCUs von Ø nach 1, so bleibt nach DYN die 1 nur während dieses Programmdurchlaufes erhalten. Bei den folgenden Durchläufen ist der ACCU an dieser Stelle Ø.

Jeder DYN-Befehl verlangt im Operanden einen Merker zur Abspeicherung des ersten Programmdurchlaufes. Beim nächsten Programmdurchlauf werden als Folge dieses gesetzten Merkers die Aktionen der betreffenden Befehlszeile nicht mehr ausgeführt (ACCU bleibt auch bei positivem Verknüpfungsergebnis = Ø). Somit ist lediglich die positive Flanke wirksam.
Die Funktion des DYN-Befehls kann am einfachsten anhand eines Schrittschalters mit entsprechendem Ersatzprogramm gezeigt werden:

Schrittschalter



mit DYN-Befehl:

```
STH      5
DYN      5ØØ
COO      4Ø
```

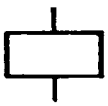
Ersatz-Programm ohne DYN:

```
STH      5
ANL      5ØØ
SEO      5ØØ
COO      4Ø
STL      5
REO      5ØØ
```

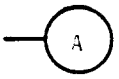
Kapitel F
Beispiele: 6, 8, 9

E 2 Schaltbefehle

	Schrittadresse	Befehl im Zahlencode	Element- oder Sprungadresse	Verknüpfungs-speicher
	STEP	CODE	OPERAND	ACC = 1
Anzeige auf SAIA® PC-Programm-eingabegerät	0002	14	0256	☀

		Zahlen-code	Mnemo-code	Befehl englisch	Beschreibung
	**Schaltbefehle**	1Ø	OUT	Set Output with Status of Accu	Setze Ausgang oder Merker mit Inhalt des Accu
		11	SEO	Set Output	Setze Ausgang oder Merker speichernd
		12	REO	Reset Output	Setze Ausgang oder Merker zurück
		13	COO	Complement Output	Frage den Ausgang oder Merker ab und setze ihn umgekehrt

OUT Ausgang setzen mit Inhalt des ACCU



OUT: Set output with status of ACCU
---> Ausgang oder Merker setzen entsprechend dem Inhalt des ACCUs.

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
OUT	1Ø	Adresse des Ausganges oder Merkers	Ø...255 (i) 288...999 (i)

(i) = indexierbar

Mit dem OUT-Befehl wird das (veränderliche) Verknüpfungsergebnis direkt auf den Ausgang bzw. Merker übertragen.

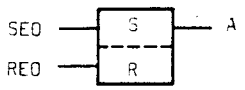
Wahrheitstabelle:

Befehl	ACCU-Inhalt	Ausgang oder Merker
OUT	1 Ø	H L

Kapitel F
Beispiele: 1, 2, 3, 5

SEO
REO

Ausgang speichernd setzen bzw. rücksetzen



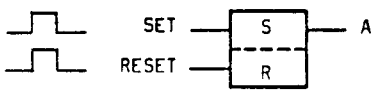
SEO: Set output ---> Ausgang speichernd setzen
REO: Reset output ---> Ausgang speichernd rücksetzen

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
SEO	11	Adresse des Ausganges oder Merkers	Ø...255* (i) 288...999 (i)
REO	12	Adresse des Ausganges oder Merkers	Ø...255* (i) 288...999 (i)

(i) = indexierbar

* Funktion dieser Befehle auf Timer und Zähler siehe STR/SCR.



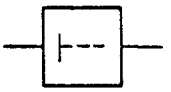
Die Befehle SEO und REO entsprechen in ihrer Funktion derjenigen eines RS-Flip-Flop.

Ein mit einem SEO-Befehl gesetzter Ausgang oder Merker bleibt solange gesetzt (H), bis er durch einen REO-Befehl wieder zurückgesetzt wird. Voraussetzung ist in beiden Fällen, dass die vorangehende Verknüpfung erfolgreich war (ACCU = 1).

Werden SEO und REO in Umlaufprogrammen (Logikplan) verwendet, so ist auf die Setz- bzw. Rücksetzpriorität zu achten.

Kapitel F
Beispiele: 4,5,10,11,12

COO Komplementieren des Ausganges



COO: Complement output ---> Frage den Ausgang ab und setze ihn umgekehrt

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
COO	13	Adresse des Ausganges oder Merkers	Ø...255 (i) 288...999 (i)

(i) = indexierbar

Der adressierte Ausgang oder Merker wird auf seinen Signalzustand (H/L) abgefragt und komplementär (umgekehrt) gesetzt. COO wird nur ausgeführt, wenn ACCU = 1.

Kapitel F
Beispiel: 6

E 3 Zeit- und Zählbefehle

E 3.1 Transfer-Befehle und Arithmetik-Befehle

	Mnemo-Code	Zahlen-Code	Operand	Erläuterungen
1. Zeile	STR/ SCR	14/ 15	256...319	Adresse des Registers
2. Zeile Transfer-Befehle **E ---> C/T**		16 17 18	Höchste Adresse von 8 aufeinander folgenden Elementen	2 x 4 Bit BCD mal 1 2 x 4 Bit BCD mal 1Ø 2 x 4 Bit BCD mal 1ØØ
T/C ---> E		19 2Ø 21 22 23 24 25 26	Höchste Adresse der Elementenreihe	Einlesebefehl für 5x4 Bit BCD Ausgabebefehl für 5x4 Bit BCD Ausgabebefehl für 8 Bit binär Ausgabebefehl für 12 Bit binär Ausgabebefehl für 16 Bit binär Einlesebefehl für 8 Bit binär Einlesebefehl für 12 Bit binär Einlesebefehl für 16 Bit binär
Arithmetik-Befehle ⊕⊖⊗÷		27 28 29 3Ø	xxx xxx xxx xxx	Addition Subtrakt. Multipl. Division mit einer Konstanten (Ø...255) oder mit dem Inhalt eines T/C (256...319)*
Transfer-Befehle C ⟶ IR C ⟶ C		31	iii	iii = Ø: Zähler wird mit dem Wert der Indexregisters geladen iii = 256...319*: Zähler wird mit dem Wert des entsprechenden T/C geladen

*) Je nach CPU stehen 64, 228 bzw. 256 Register zur Verfügung.
siehe Übersicht der Register Seite III .

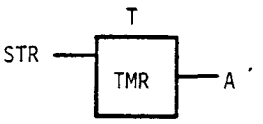
E 3.2 Zeitglieder und Zähler

Für diese Funktionen stehen zusammen 32 adressierbare Register zur Verfügung, die als Timer (Zeitglieder) oder Zähler programmiert werden können. Die gleiche Registeradresse kann für verschiedene Funktionen (Timer oder Zähler) beliebig oft im Programm verwendet werden, sofern die vorangehende Funktion nicht mehr benötigt wird.

Es handelt sich um zweizeilige Befehle, wobei in der 2. Zeile jeweils der feste interne oder veränderliche externe Zeit- oder Zählwert programmiert wird.

Alle Zählerregister können zur Speicherung und zum Transfer von Daten, sowie für arithmetische Funktionen verwendet werden.

Mit dem Befehl DTC (siehe weiter unten) lassen sich die Register-Inhalte auf einfachste Weise auf dem Programmiergerät oder einem Operand-Display 4-stellig anzeigen.



STR Setze Zeitglied

STR: Set Timer ---> Setze Zeitglied auf vorgewählten Wert und starte es

Befehlsformat (zweizeiliger Befehl):

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
STR	14	Adresse des Zeitgliedes	256...287 (i)	◀ 1. Zeile
---	ØØ	Zeitwert in 1/1Ø sec	Ø ...2Ø47	◀ 2. Zeile

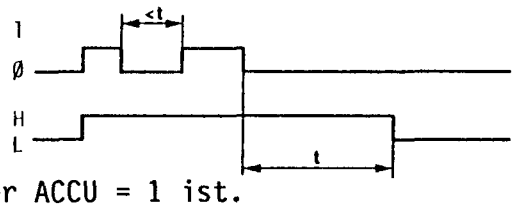
(i) = indexierbar

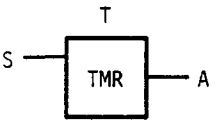
Die Zeitglieder können mit einem entsprechenden Zeitwert gesetzt werden. Durch das Setzen ändert sich der logische Zustand von L nach H und die Zeit beginnt sofort abzulaufen.

Wird während des Zeitablaufes der Timer erneut gesetzt (STR), so beginnt dieser wieder mit der entsprechenden Vollzeit. Die Grundfunktion entspricht somit einer Abfallverzögerung.

Setzbefehl (STR) bei ACCU

Zeitglied (A), Signalzustand





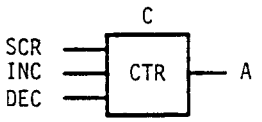
STR wird nur aufgeführt, wenn der ACCU = 1 ist.

Kapitel F
Beispiele: 7, 9, 1Ø, 12

Wird der Befehl REO auf einen Timer angewendet (z.B. REO 26Ø), so kann ein laufender Timer angehalten werden, bis er mit SEO wieder freigegeben wird. Es läuft dann die Restzeit ab.

Hinweise:
- Die Zeitbasis ist ab Werk auf Ø,1s eingestellt. Um eine feinere Zeitaufteilung zu erreichen, kann der Anwender die entsprechende Hardware-Brücke öffnen, womit die Zeitbasis auf Ø,Ø1s verkleinert wird.
- Die relative Zeitgenauigkeit (Wiederholgenauigkeit) beträgt +Ø/-1 Digit. Das ergibt z.B. bei 3s im Ø,1s-Takt 2.9Ø bis 3.ØØs Die absolute Zeitgenauigkeit hängt vom Systemquarz ab.
- Mit der geänderten Zeitbasis von Ø,Ø1s wird der Prozessor nicht etwa schneller, sondern die Zykluszeit wird dadurch um ca. 2Ø% verlängert. Hingegen wird die Zeitgenauigkeit verbessert 3s ⟶ 2,99 bis 3,ØØs.
- Durch Einhängen der Brücke NVOL werden alle Zeit- und Zähler-Register nullspannungssicher.

SCR — Setze Zähler



SCR: Set Counter ---> Setze Zähler auf vorgewählten Wert

Befehlsformat (zweizeiliger Befehl):

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
SCR	15	Zähler-adresse	256...319* (i)	1. Zeile
---	ØØ	Zähl-wert	Ø...2Ø47	2. Zeile

(i) = indexierbar

Die Zähler können mit einem entsprechenden Zählwert gesetzt werden. Durch das Setzen ändert sich der Signalzustand von L nach H. Beim Abwärtszählen wechselt beim Uebergang vom Zählerinhalt ØØØ1 auf ØØØØ auch sein Signalzustand wieder zurück auf L. Der Zähler kann keine negativen Werte annehmen. Diese müssten allenfalls mit einem zweiten Zähler erfasst werden.

Zählwert (Register-Inhalt)	Logischer Zustand des Zählers
ØØØØ	L
≥ ØØØ1	H

SCR wird nur ausgeführt wenn der ACCU = 1 ist.

Kapitel F
Beispiele: 8, 18

*) Je nach CPU stehen 64, 228 bzw. 256 Register zur Verfügung. Siehe Übersicht der Register Seite III.

Funktionserweiterungen bei Zeitgliedern und Zählern

Mit der 2. Zeile der Anweisungen STR und SCR wird der Zeit- bzw. Zählwert festgelegt. Ausser dem Code ØØ in dieser 2. Zeile können mit höheren Codes Bereichserweiterungen bzw. externe Werteingabe realisiert werden.

Bereichserweiterung

Mit dem Code ØØ beträgt der max. Wert, der im Operand verarbeitet werden kann 2Ø47 bzw. 2Ø4,7 s. Grössere Werte können gemäss Tabelle mit den Codes Ø1 bis 15 vorgewählt werden:

Code der 2. Zeile	Gesamtwert = Wert im Operand +
ØØ	+ Ø
Ø1	+ 2Ø48
Ø2	+ 4Ø96
Ø3	+ 6144
Ø4	+ 8192
Ø5	+ 1Ø24Ø
Ø6	+ 12288
Ø7	+ 14336
Ø8	+ 16382
Ø9	+ 18432
1Ø	+ 2Ø48Ø
11	+ 22528
12	+ 24576
13	+ 26624
14	+ 28672
15	+ 3Ø72Ø

Für Zähler gelten die angegebenen Werte direkt.

Für Timer mit: - Zeitbasis $^1/_{10}$ s (x 0,1s)
(Hardwarebrücke eingehängt, auf Lieferungzustand)

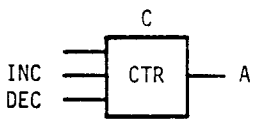
- Zeitbasis $^1/_{100}$ s (x 0,01s)
(Hardwarebrücke geöffnet)

Aufgabe: Es soll eine Zeit von 1Ø min = 6ØØs programmiert werden.

Lösung: Der Code heisst Ø2, weil 6ØØØ zwischen 4Ø96 und 6144 liegt. Der zugehörige Operand 6ØØØ - 4Ø96 = 19Ø4. Muss öfters mit längeren Zeiten gearbeitet werden, so empfiehlt sich, die Kombination eines Timers mit einem Zähler.

INC
DEC

Aufzählen, abzählen



INC: Increment Counter ---> Zählerstand + 1
DEC: Decrement Counter ---> Zählerstand - 1

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
INC	17	Adresse des Zähler-Registers	256 ... 319 * (i)
DEC	18	Adresse des Zähler-Registers	256 ... 319 * (i)

(i) = indexierbar

INC und DEC werden nur ausgeführt, wenn der ACCU = 1 ist.

*) Je nach CPU stehen 64,228 bzw. 256 Register zur Verfügung.
Siehe Übersicht der Register Seite III

Kapitel F
Beispiele: 8 und 18

E 3.3 Transfer-Befehle von Binär-Werten und BCD-Werten (Übersicht)

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
SCR/STR	15/14	Adresse des Registers T/C	256 ... 319 * (i)	(1. Zeile)
---	CODE	Höchste Adresse von N aufeinanderfolgenden Elementen	Element (E, A, Merker)	(2. Zeile)

(i) = indexierbar

CODE	N Elemente	Beschreibung der Instruktion
16	8 Elemente	Einlesen von 2 x 4 Bit BCD x 1
17	8 Elemente	Einlesen von 2 x 4 Bit BCD x 10
18	8 Elemente	Einlesen von 2 x 4 Bit BCD x 100
19	20 Elemente	Einlesen von 5 x 4 Bit BCD
20	20 Elemente	Ausgabe von 5 x 4 Bit BCD
21	8 Elemente	Ausgabe von 8 Bit binär
22	12 Elemente	Ausgabe von 12 Bit binär
23	16 Elemente	Ausgabe von 16 Bit binär
24	8 Elemente	Einlesen von 8 Bit binär
25	12 Elemente	Einlesen von 12 Bit binär
26	16 Elemente	Einlesen von 16 Bit binär

*) Je nach CPU stehen 64,228 bzw. 256 Register zur Verfügung.
Siehe Übersicht der Register Seite III

E 3.4 Externe Werteingabe im BCD-Format (2x4 Bit)

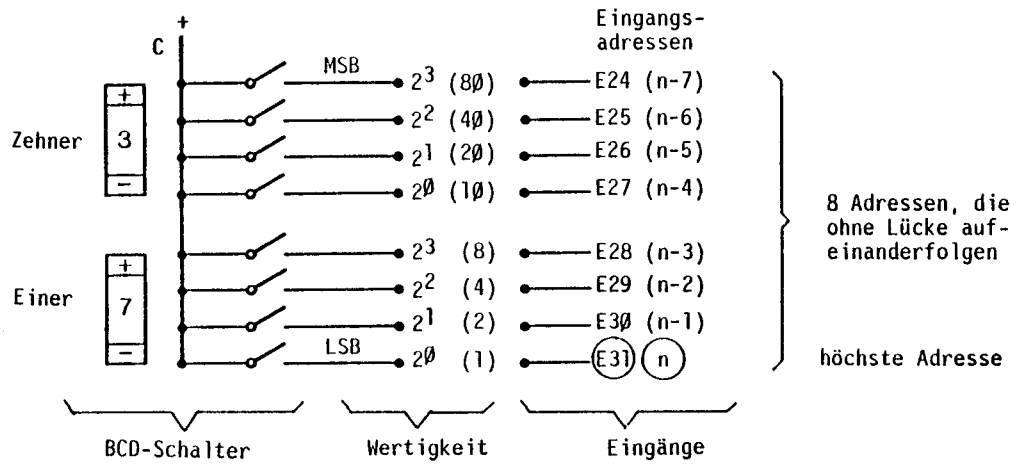
Zeit- oder Zählwerte können auch extern, in Form eines 2-stelligen BCD-Wertes (ØØ...99) eingegeben werden. Dieser Wert wird mit dem Faktor 1, 1Ø oder 1ØØ multipliziert, je nach Code 16, 17 oder 18 in der 2. Programmzeile.

Befehlsformat:

Multiplikator	Code der 2. Zeile	Operand
1 x 1Ø x 1ØØ x	16 17 18	Höchste Adresse (n) von 8 aufeinanderfolgenden Eingängen oder Merkern

Der 2-stellige BCD-Wert, bestehend aus 8 aufeinanderfolgenden Adressen (= 1 Byte), kann von den Eingängen, den Merkern eventuell auch von den Ausgängen eingelesen werden.

Beispiel für 2-stelligen BCD-Schalter, angeschlossen an 8 Eingängen:



Diese Anordnung ergibt als

		Code	Operand	
- Zähler für	1... 99	16	31	2. Zeile
Zähler für	1Ø... 99Ø	17	31	
Zähler für	1ØØ...99ØØ	18	31	
- Timer * für	Ø,1...9,9s	16	31	
Timer * für	1...99s	17	31	
Timer * für	1Ø...99Ø s	18	31	

*) Für Standardtakt Ø,1 sec.

Kapitel F
Beispiele: 9 und 18

E 3.5 Erweiterte Befehle

STR SCR	Erweiterte Zähler- und Timer-Befehle für Datentransfer und Arithmetik

E 3.5.1 Datentransfer und Übertrag

a) Einlesebefehl von Elementen (E, A, M) als Wert von Timern (T) bzw. Zählern (C). Im Operand der 2. Zeile steht die höchste Adresse der Elementreihe (LSB).

Codes	(2. Zeile)
19	5x4 Bit BCD (5 Dekaden)
24	8 Bit binär
25	12 Bit binär
26	16 Bit binär

Beispiel:

STR (14)	256
24	431

24 = 8 Bit binär
431 = 8 Merker, 424...431

b) Ausgabebefehle aus Registern von T bzw. C auf Elemente (A, M). Im Operand der 2. Zeile steht die höchste Adresse der Elementreihe (LSB).

Codes	(2. Zeile)
20	5x4 Bit BCD (5 Dekaden)
21	8 Bit binär
22	12 Bit binär
23	16 Bit binär

Beispiel:

SCR (15)	300
21	75

21 = 8 Bit, binär
75 = 8 Ausgänge, 68...75

c) Laden eines Timers (T) bzw. Zählers (C) mit dem Inhalt des Indexregisters (IR) bzw. mit dem Wert eines anderen T/C.

SCR (15)	ccc
31	iii

31 = CODE

iii = 0 Der Zähler ccc wird mit dem Inhalt des Indexregisters (des entsprechenden Parallelprogrammes) geladen.

iii = 256...287 Der Zähler ccc wird mit dem Inhalt des adressierten T/C iii geladen.

iii = 288...319* Der Zähler ccc wird mit dem Inhalt des adressierten C iii geladen.

E 3.5.2 Arithmetische Operationen

Die Operationen werden zwischen zwei Zählregistern ausgeführt, wobei im Register der 1. Zeile (ccc) das Resultat der Operation deponiert wird. Wird die Kapazitätsgrenze des Registers (65'535) überschritten, so wird der ACCU des Bitprozessors auf 0 gesetzt. In der 2. Zeile kann für einen Wert ≤ 255 auch eine Konstante eingegeben werden.

SCR (15) oder STR (14)

SCR (15)	ccc
CODE	xxx

ccc = Adresse des 1. T/C
xxx = Adresse des 2. T/C oder Wert einer Konstanten ≤255
CODE = 27...30, siehe nächste Seite

Hinweis: Alle Elementadressen können indexiert behandelt werden. Die Befehle STR und SCR werden nur ausgeführt, wenn ACCU = 1.

*) Je nach CPU stehen 64, 228 bzw. 256 Register zur Verfügung. Siehe Übersicht der Register Seite III.

E 3.6 CODES für Arithmetik-Operationen

SCR (15)	ccc
CODE	xxx

27 Addition des Inhaltes von ccc und des Inhaltes von xxx mit Speicherung des Resultates in ccc. Wird 65'535 überschritten, so wird ACCU = Ø.

Inhalt ccc + Inhalt xxx ---> Inhalt ccc

28 Subtraktion des Inhaltes von xxx vom Inhalt ccc mit Speicherung des Resultates in ccc. Ein negatives Resultat setzt ACCU = Ø.

Inhalt ccc - Inhalt xxx ---> Inhalt ccc

29 Multiplikation des Inhaltes von ccc mit Inhalt xxx mit Speicherung des Resultates in ccc. Wird 65'535 überschritten, so wird ACCU = Ø.

Inhalt ccc x Inhalt xxx ---> Inhalt ccc

3Ø Division des Inhaltes von ccc durch Inhalt xxx mit Speicherung des Resultates in ccc. Sollte der Inhalt von xxx = Ø sein, so wird auch ACCU = Ø, wobei der Inhalt in ccc unverändert bleibt. Ein allfälliger Rest geht bei der Division verloren.

Inhalt ccc ÷ Inhalt xxx ---> Inhalt ccc

Beispiele:

- Vor der Addition — C256 = 3Ø, C26Ø = 54

  Operation

SCR(15)	256
27	26Ø

  Nachher — C256 = 84, C26Ø = 54, ACCU = 1

- Vor der Subtraktion — C258 = 124, C274 = 146 [1]

  Operation

SCR(15)	258
28	274

  Nachher — C258 = 65'514 [1], C274 = 146, ACCU = Ø

  [1] 124 - 146 = -22
  ---> 65'536 - 22 = 65'514

- Vor der Multiplikation — C26Ø = 12, C282 = 6

  Operation

SCR(15)	26Ø
29	282

  Nachher — C26Ø = 72, C282 = 6, ACCU = 1

- Vor der Division — C31Ø = 1942, Konstante = 23

  Operation

SCR(15)	31Ø
3Ø	23

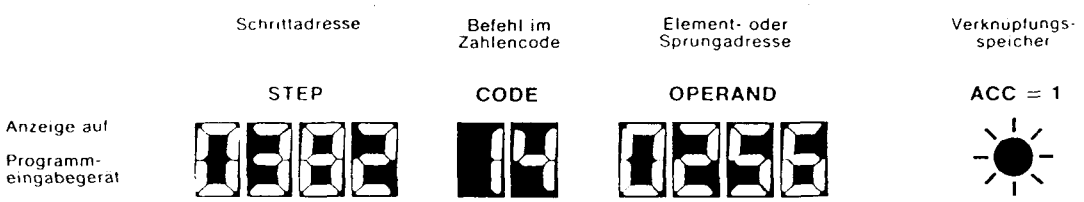
  Nachher — C31Ø = 84, ACCU = 1, der Rest (1Ø) wird nicht gespeichert.

E 3.7 Zusammenfassung aller Befehle für Zeit- und Zählregister

	Mnemo-Code	Zahlen-Code	Operand	Erläuterungen
1. Zeile	STR SCR	14 15	256...319 [1]	Adresse des Registers
2. Zeile	Konstanten in Timer/Zähler	ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7 Ø8 Ø9 1Ø 11 12 13 14 15	xxxx xxxx	Wert im Operand + Ø . + 2Ø48 . + 4Ø96 . + 6144 . + 8192 . + 1Ø24Ø . + 12288 . + 14336 . + 16348 . + 18432 . + 2Ø48Ø . + 22528 . + 24576 . + 26624 . + 28672 Wert im Operand + 3Ø72Ø
	Datentransfer	16 17 18	Höchste Adresse von 8 aufeinanderfolgenden Elementen	2 x 4 Bit BCD x 1 2 x 4 Bit BCD x 1Ø 2 x 4 Bit BCD x 1ØØ
		19 2Ø 21 22 23 24 25 26	Höchste Adresse der Elementenreihe	Einlesebefehl für 5x4 Bit BCD Ausgabebefehl für 5x4 Bit BCD Ausgabebefehl für 8 Bit binär Ausgabebefehl für 12 Bit binär Ausgabebefehl für 16 Bit binär Einlesebefehl für 8 Bit binär Einlesebefehl für 12 Bit binär Einlesebefehl für 16 Bit binär
	Arithmetik	27 28 29 3Ø	xxx xxx xxx xxx	Addition Subtraktion Multiplikation Division } mit einer Konstanten (Ø...255) oder mit dem Inhalt eines T/C (256...319) [1]
	Übertrag IR→C / C→C	31	iii	iii = Ø Zähler wird mit dem Wert des Indexregisters geladen iii = 256...319 [1] Zähler der 1. Zeile wird mit dem Wert des T/C der 2. Zeile geladen

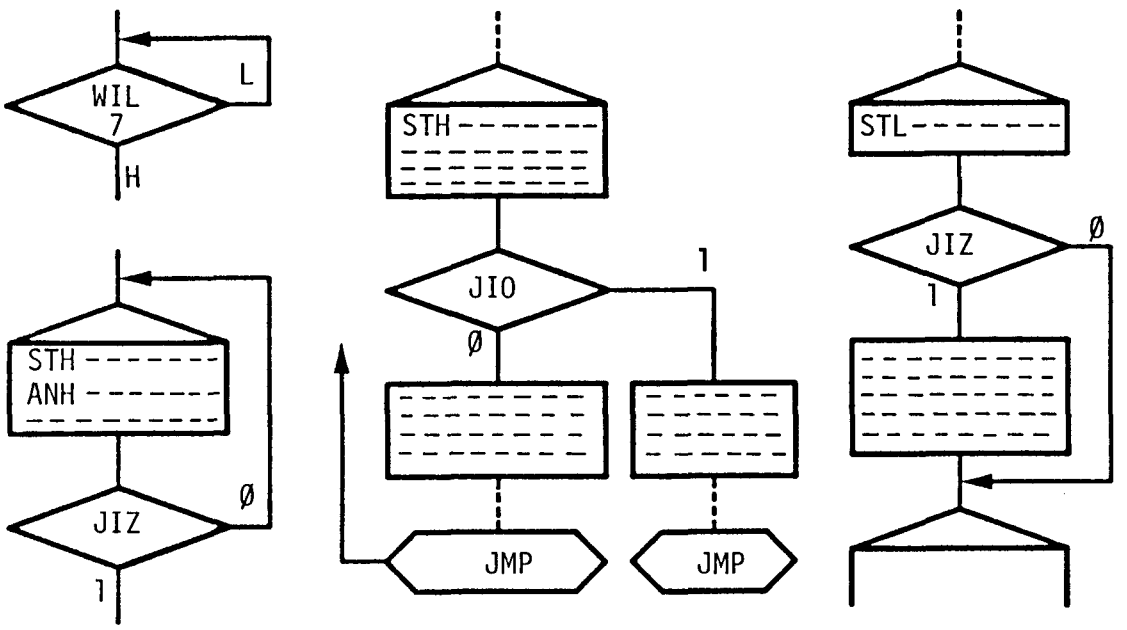
1) Übersicht des Zähl- und Zeitregisters siehe Seite III.

E 4 Sprung- und Wartebefehle



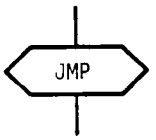
		Zahlen-code	Mnemo-code	Befehl englisch	Beschreibung
	Sprungbefehle	2Ø	JMP	Unconditional Jump	Unbedingter Sprung auf Schrittadresse
		21	JIO	Jump if Accu is One	Springe, wenn { Accu = 1 / Accu = Ø } auf Schrittadresse
		22	JIZ	Jump if Accu is Zero	
		23	JMS	Jump to Subroutine	Springe in Unterprogramm
		24	RET	Return from Subrout.	Rücksprung vom Unterprogramm
	Wartebefehle	25	WIH	Wait if High	Warte, solange Element { High / Low
		26	WIL	Wait if Low	

Mit Sprung- und Wartebefehlen können nicht nur Warteschleifen (gemäss Flussdiagramm), sondern auch Verzweigungen bzw. Uebersprünge (wie sie auch im Logikplan sinnvoll sein können) realisiert werden. Mit besonderen Befehlen wird in Subroutinen gesprungen.



Warteschleifen Verzweigung Uebersprünge

JMP Unbedingter Sprung



JMP: Jump ---> unbedingter Programmsprung (unabhängig vom ACCU).

Befehlsformat:

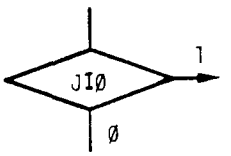
Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
JMP	2Ø	Schritt-Adresse	1...2Ø47	
Für einen Sprung zu einer Schrittadresse > 2Ø47, ist eine zweite Befehlszeile zu verwenden:				
JMP	2Ø	Immer Ø	ØØØØ	1. Zeile
- - -	ØØ	Schritt-Adresse	Ø...8191	2. Zeile

Jedes Programm wird mit einem unbedingten Rücksprung abgeschlossen.
JMP setzt den ACCU = 1.

Alle Beispiele

JIO
JIZ

Bedingter Sprung



JIO: Jump if ACCU is One ---> Springe, wenn ACCU = 1
JIZ: Jump if ACCU is Zero ---> Springe, wenn ACCU = Ø

Befehlsformat:

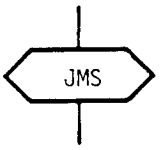
Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
JIO	21	Schritt-Adresse	1...2Ø47	
JIZ	22	Schritt-Adresse	1...2Ø47	
Für einen bedingten Sprung zu einer höheren Schrittadresse als 2047, ist eine zweite Befehlszeile zu verwenden :				
JIO	21	Immer Ø	ØØØØ	1. Zeile
- - -	ØØ	Schritt-Adresse	Ø...8191	2. Zeile
JIZ	22	Immer Ø	ØØØØ	1. Zeile
- - -	ØØ	Schritt-Adresse	Ø...8191	2. Zeile

Abhängig vom ACCU, d.h. von einer vorangehenden Abfrage oder Verknüpfung werden die Sprünge nur bedingt ausgeführt. Ist die Sprungbedingung nicht erfüllt, so läuft das Programm gradlinig weiter. Ob gesprungen wird oder nicht, in beiden Fällen setzen die Sprungbefehle den ACCU = 1.

Kapitel F
Beispiele: 11, 15, 16

JMS
RET

Sprung in und Rücksprung aus Unterprogramm (Subroutine)



JMS: Jump to Subroutine ---> Spring ins Unterprogramm
RET: Return from Subroutine ---> Rücksprung vom Unterprogramm = Ende des Unterprogrammes.

Beide Befehle werden unbedingt, d.h. abhängig vom ACCU-Status ausgeführt.

Befehlsformat:

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
JMS	23	Anfangsadresse des Unter-programmes	1...2Ø47	nur eine Zeile
Für Unterprogramme, die bei einer höheren Adresse als 2Ø47 beginnen, ist der folgende zweizeilige Befehl zu verwenden:				
JMS	23	Immer Ø	ØØØØ	1. Zeile
_ _ _	ØØ	Anfangsadresse des Unterprogrammes	Ø...8191	2. Zeile
RET	24	Immer Ø	ØØØØ	

Im vorangehenden Abschnitt D 5.6 ist das Arbeiten mit Unterprogrammen beschrieben.

Beim Sprung ins Unterprogramm wird die nächstfolgende Schrittadresse im Hauptprogramm gespeichert. Der Rücksprung mit dem Befehl RET erfolgt damit automatisch an die richtige Adresse. Aus diesem Grund steht im Operand von RET lediglich Ø.

Von einem 1. Unterprogramm kann in ein 2. und von diesem auch in ein 3. Unterprogramm gesprungen werden (Verschachtelung bis zu 3 Ebenen). Die Rücksprünge erfolgen in umgekehrter Reihenfolge zurück bis zum Hauptprogramm.

JMS und RET setzen den ACCU = 1.

Operand ØØØØ ist reserviert für Sprünge von Zeilen, die 2Ø47 überschreiten.

Kapitel F
Beispiele: 12, 18

JMP, JIO, JIZ, JMS Erweiterte Funktionen für Adressen 2048 . . . 8191

Sollen Sprünge mit Zieladressen in der zweiten Hälfte des Anwenderspeichers, d.h. ab Schrittadresse 2Ø48 bis 8191 programmiert werden, so sind diese Sprungbefehle zweizeilig auszuführen.

a) Sprungbefehle mit Operanden 1 bis 2Ø47
(Operand ØØØØ ist nicht gestattet, siehe b).

Beispiel: Bedingter Sprung mit Zieladresse 1845

entweder

```
JIO(21)  1845
```

einzeilig wie bisher

oder

```
JIO(21)     Ø
   ØØ    1845
```

zweizeilig, wobei in der ersten Zeile der Operand Ø steht.

b) Sprungbefehle mit Operanden 2Ø48 bis 8191

Beispiel: Sprung in die Subroutine 328Ø

Bei der Programmierung:

```
5Ø1   JMS(23)     Ø    [E]
5Ø2      ØØ    328Ø    [E]
```

Bei der Kontrolle:

```
5Ø1   JMS(23)     Ø    [+]
5Ø2      Ø1    1232    [C]   ---> convert
5Ø2      EE    328Ø    [+]
```

Der Wert Ø1 1232, wie er im Anwenderspeicher steht, entspricht der Sprungadresse 328Ø. Ø1 steht für das Vielfache von 2Ø48 und 1232 ist der Rest (1 x 2Ø48 + 1232 = 328Ø).

Mit der Taste [C] wird die wirkliche Sprungadresse angezeigt, wobei im CODE die Zeichen EE stehen (gilt für Programmiergerät ..PØ5).

Bei einem Sprungbefehl mit dem Operanden Ø wird automatisch die zweite Zeile für die Zieladresse gelesen. Ein Sprung auf die Zieladresse Ø besteht daher immer aus zwei Zeilen:

```
JMP(2Ø)    Ø
   ØØ      Ø
```

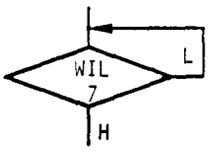
c) Ein praktisches Beispiel anhand eines Blinkers in einer Subroutine.

Hauptprogramm

```
5ØØ   26  WIL     1
5Ø1   23  JMS     Ø
5Ø2       ØØ   35ØØ  ⇒
5Ø3   2Ø  JMP   5ØØ
```

Subroutine

```
⇒ 35ØØ  Ø2  STL   256
  35Ø1  14  STR   256
  35Ø2      ØØ      2
  35Ø3  13  COO    34
  35Ø4  24  RET     Ø  ⇒
```



WIH
WIL Wartebefehle

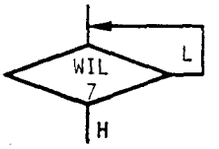
WIH: Wait if High ---> Warte, solange Element Signalzustand "H" führt
WIL: Wait if Low ---> Warte, solange Element Signalzustand "L" führt

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
WIH	25	Adresse des Elementes	Ø...999 (i)
WIL	26	Adresse des Elementes	Ø...999 (i)

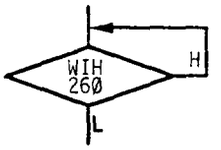
(i) = indexierbar

Der Wait-Befehl wird ausschliesslich im Flussdiagramm verwendet. Der Prozessor wartet solange in der Warteschleife, bis das adressierte Element den zur Fortsetzung erforderlichen Signalzustand angenommen hat.



z.B. warten, solange ein Eingang "L" ist.

z.B. warten, solange die Zeit abläuft,
d.h. das Element (Timer) "H" ist.

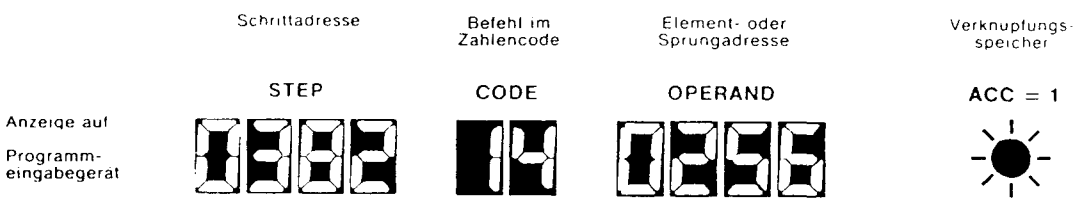


Mit "Wait" kann ähnlich wie mit "Start" ein Programmteil eröffnet werden, da in beiden Fällen ein Element auf seinen Signalzustand abgefragt wird.
WIH und WIL setzen den ACCU = 1, d.h. nachfolgend programmierte Aktionen werden nach Verlassen der Warteschleife ausgeführt.

Kapitel F
Beispiele: 1Ø, 12, 16

E 5 Hilfsbefehle

Indexierung(siehe E6), Zusätzliche PAS-Instruktionen (siehe E7), Anzeigebefehle (siehe E8)



	Zahlen code	Mnemo-code	Befehl english	Beschreibung
Hilfsbefehle (Kapitel E5)	00	NOP	No Operation	Keine Operation
	19	SEA	Set Accu	Setze Accu = 1
Indexierung (Kapitel E6)	16	SEI	Set Index	Setze Indexregister auf vorgewählten Wert
	27	INI	Increment Index	Erhöhe das Indexregister um 1
	28	DEI	Decrement Index	Erniedrige das Indexregister um 1
PAS-Instruktionen (Kapitel E7)	29	PAS	0...15	Zuweisung des parallelprogramms
	29	PAS	18	Begrenzung der Parallelprogramme
	29	PAS	30...38	Check-Sum des System oder Anwenderspeichers
Anzeigebefehle (Kapitel E8)	30	DOP	Display Operand	Anzeige eines Operanden
	31	DTC	Display Timer or counter	Anzeige des Zeitstandes oder Zählerstandes

NOP Keine Operation

NOP: No Operation ---> Keine Operation

Befehlsformat:

Befehlscode		Operand
Mnemo-code	Zahlen-code	
NOP	ØØ	Ø

Dieser Befehl wird vom Prozessor wohl abgearbeitet, löst aber keinerlei Funktionen aus. Er dient dazu, Reserveplätze für Programm-Ergänzungen zu schaffen und Programm-Lücken zu füllen.

Programmzeilen, die nach einer Aenderung überflüssig werden, können mit NOP überschrieben (gelöscht) werden. Mehrere NOP erhält man durch mehrfaches Betätigen der Taste "Enter" am Programmeingabegerät.

SEA ACCU = 1 setzen

SEA: Set Accumulator = 1 ---> ACCU = 1 setzen

Befehlsformat:

Befehlscode		Operand
Mnemo-code	Zahlen-code	
SEA	19	Ø

Mit SEA wird der ACCU unbedingt auf 1 gesetzt. SEA wird daher vor Befehlen verwendet, die nur ausgeführt werden bei ACCU = 1, z.B. vor DTC.

Kapitel F
Beispiel 8

E 6 Indexierung

SEI
INI
DEI

Elementadressen - Indexierung (Reihenbehandlung)

SEI: Set Index ---> Setze das Indexregister auf den Anfangswert gemäss Operand

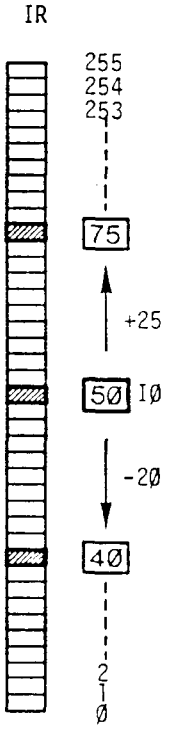
INI: Increment Index ---> Erhöhe das Indexregister um 1 bis zum oberen Endwert gemäss Operand

DEI: Decrement Index ---> Erniedrige das Indexregister um 1 bis zum unteren Endwert gemäss Operand

Oft muss eine Reihe von Eingängen, Ausgängen, Merkern, Timern oder Zählern in gleicher Art behandelt werden (als Beispiel diene das Rücksetzen aller Haftspeicher gemäss D 5.7). In diesen Fällen können lange und aufwendige Programme mit Hilfe der Adress-Indexierung drastisch verkürzt werden.

Als Hilfsmittel dient das Indexregister IR, das eine Art Zählregister ist mit einer Kapazität von Ø bis 255*. Die 3 Befehle erlauben das Setzen bzw. Verändern des Registerinhaltes bis zum gewünschten Grenzwert.

Beispiel:



INI 75 Erhöht das Indexregister um 1, bis der obere Endwert 75 erreicht ist. Der ACCU ist dann = Ø.

SEI 5Ø Setzt das Indexregister auf den Anfangswert IØ = 5Ø. Der ACCU wird auf 1 gesetzt.

DEI 4Ø Erniedrigt das Indexregister um 1, bis der untere Endwert 4Ø erreicht ist. Der ACCU ist dann = Ø.

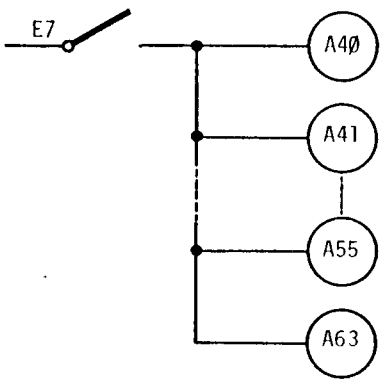
---

*) Jedes Parallelprogramm besitzt sein eigenes Indexregister, sodass total 16 Register zur Verfügung stehen. Diese Register verlieren ihren Inhalt bei Spannungsausfall.

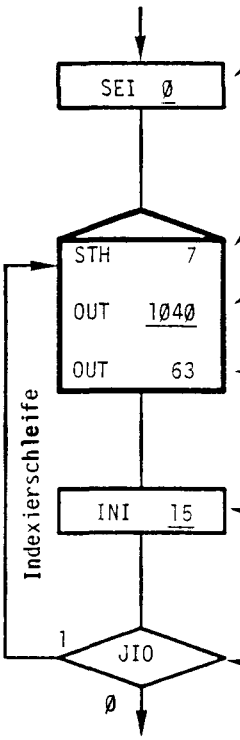
Die CPU der PCA232 besitzt Indexregister mit einer Kapazität von 1K (Ø...1Ø23). Siehe Beispiel d) am Ende dieses Kapitels.

Alle Element-Operanden, welche in der Beschreibung Teil E mit einem "(i)" bezeichnet sind, können indexiert behandelt werden. Der Prozessor zählt dabei zur im Operanden angegebenen Elementadresse den jeweils aktuellen Stand des Indexregisters hinzu.

Ein Beispiel soll dies erläutern:



Durch Aktivierung von E7 sollen die Ausgänge A4Ø bis A55 und zusätzlich der Ausgang A63 aktiviert werden.



Das Indexregister wird auf den Anfangswert Ø gesetzt.

Abfrage des Einganges E7.

Die Indexierfunktion ab Ausgang A4Ø wird durch Hinzufügen der Zahl 1ØØØ bewirkt. Die Adresse 4Ø wird im ersten Schleifendurchgang nicht verändert, da das Indexregister vorerst Ø ist (SEI Ø). Bei den folgenden Durchläufen wird der Registerinhalt durch den Befehl INI jeweils um 1 erhöht, sodass nacheinander die Ausgänge A41, 42, 43 bis 55 gesetzt werden.

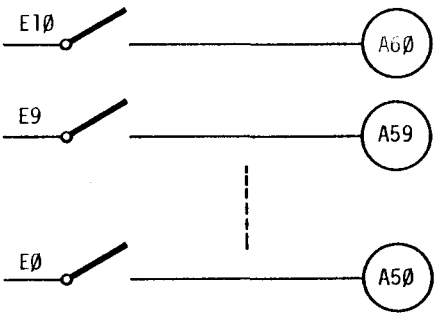
Ausgang A63 ist direkt adressiert (ohne 1ØØØ), sodass das Indexregister für ihn keine Wirkung hat.

INI bewirkt die Erhöhung des Registers jeweils um 1 pro Durchgang bis zum Wert 15. Solange das Register <15 ist, hat der ACCU nach Abarbeiten von INI 15 den Wert 1.

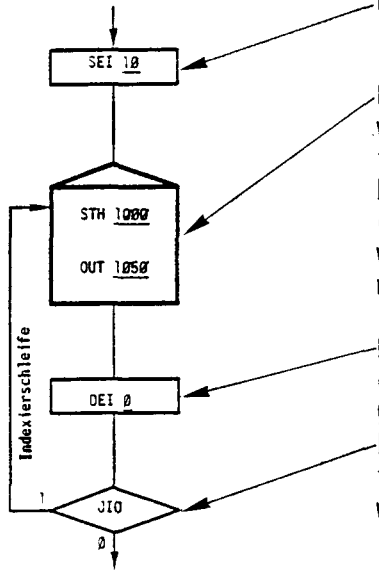
Der bedingte Sprung wird also solange ausgeführt, bis die Schleife Ø bis 15 mal, d.h. 16 mal durchlaufen ist und alle 16 Ausgänge von A4Ø bis A55 behandelt sind.

Kapitel F
Beispiele: 15, 16, 17

Zweites Beispiel:



Die Eingänge E1Ø bis EØ sollen auf die Ausgänge A6Ø bis A5Ø wirken. Die Behandlung soll in absteigender Reihenfolge mittels DEI erfolgen.



Das Indexregister wird auf den Anfangswert 1Ø (1Ø - Ø = 1Ø bzw. 6Ø - 5Ø) gesetzt.

Eingänge wie auch Ausgänge sollen indexiert werden, was durch Hinzufügen der Zahl 1ØØØ erfolgt. Es werden die tiefsten Adressen der Reihen eingetragen, da ja anfänglich der Registerinhalt von 1Ø dazugezählt wird. Somit wird zuerst E1Ø mit A6Ø behandelt, dann E9 mit A59 bis EØ mit A5Ø.

DEI bewirkt bei jedem Durchlauf die Erniedrigung des Registerwertes um 1 bis der angegebene Wert Ø erreicht wird.
Bei Registerwert Ø = DEI Ø wird der ACCU ebenfalls Ø, sodass die Indexierschleife verlassen wird.

Befehlsformate:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
SEI	16	Anfangswert des Index-registers (IØ)	IØ = Ø...255
INI	27	Oberer Endwert für erhöhende Indexierung	1...255 [1]
DEI	28	Unterer Endwert für erniedrigende Indexierung	Ø...254 [2]

1) Wird durch INI der max. Wert von 255 erreicht, hat das Indexregister nach dem nächstfolgenden INI den Wert Ø (...254, 255, Ø, 1...)

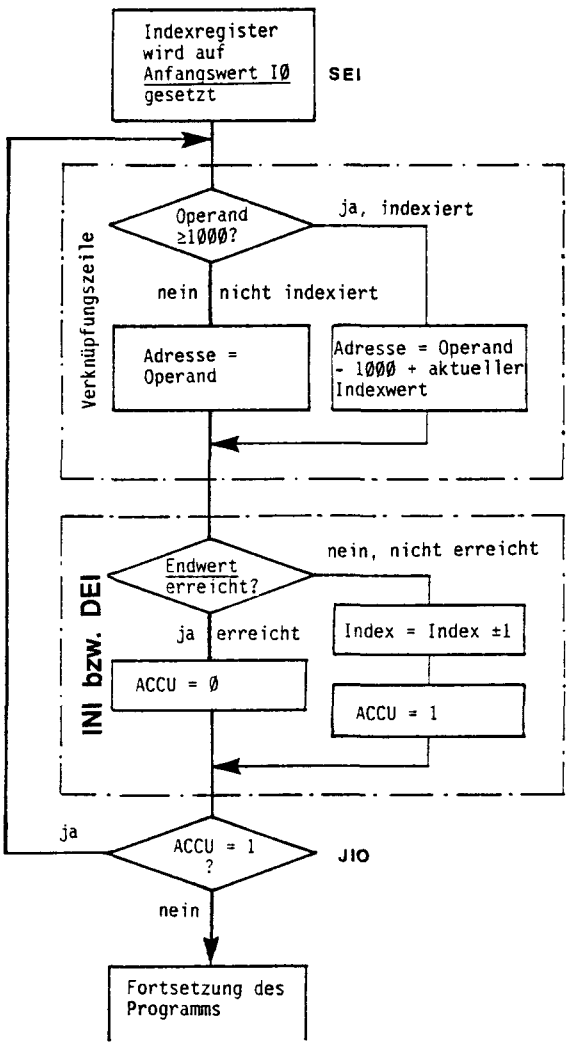
2) Wird durch DEI der Wert Ø erreicht, hat das Indexregister nach dem nächstfolgenden DEI den Wert 255 (...1, Ø, 255, 254).

Wahrheitstabelle:

Befehl	Einfluss auf ACCU
SEI	Befehl setzt ACCU = 1
INI	Wenn Registerwert ≠ Operand ---> ACCU = 1 Wenn Registerwert = Operand ---> ACCU = Ø
DEI	Wenn Registerwert ≠ Operand ---> ACCU = 1 Wenn Registerwert = Operand ---> ACCU = Ø

Zusammenfassung der Indexierung

Indexierschema:



a) Indexierung mit INI

- Das Register wird mit SEI im allgemeinen auf den Wert Ø gesetzt.
- Von den zu indexierenden Elementen werden die tiefsten Adressen (+1ØØØ) eingesetzt.
- Der obere Endwert für INI ergibt sich aus der Differenz von höchster und tiefster Adresse, welche indexiert behandelt werden muss.

b) Indexierung mit DEI

- Das Register wird mit SEI im allgemeinen auf den Wert X gesetzt. X = Differenz zwischen höchster und tiefster Adresse, welche indexiert behandelt werden sollen.
- Von den zu indexierenden Elementen werden die tiefsten Adressen (+1ØØØ) eingesetzt.
- Für den unteren Endwert für DEI wird im allgemeinen Ø eingesetzt.

Weiter Programmierungsmöglichkeiten mit der Indexier-Technik

In diesem Abschnitt ist nur die sogenannte Reihenbehandlung mittels Indexierung aufgeführt. Das Indexregister kann aber auch in dem Sinn verwendet werden, dass z.B. Subroutinen je nach Stand des Indexregisters auf andere Elemente wirken.

Beispiel:

Durch Schliessen von E1 blinkt A45, mit E2 blinkt A5Ø.

Hauptprogram

```
********************* HAUPT-PROGRAMM
ADDR  NC  MNC  OPRD
 700  01  STH     1
 701  22  JIZ   704 ->
 702  16  SEI     0
 703  23  JMS   710 =>
 ----------------
 704  01  STH     2
 705  04  ANL     1
 706  22  JIZ   700 ->
 707  16  SEI     5
 708  23  JMS   710 =>
 709  20  JMP   700 ->
```

Subroutine

```
==================== SUBROUTINE
ADDR  NC  MNC  OPRD
 710  02  STL   256
 711  14  STR   256
 712  00  00      3
 713  13  COO  1045
 714  24  RET     0 ->
```

SEI, INI, DEI Zusätzliche Funktionen

SEI(16) iii Setzen des Indexregisters

a) iii = Ø...255
   Das Indexregister wird mit dem Wert iii geladen.

b) iii = 256...319* (für PCA232: 256...511)
   Das Indexregister wird mit dem Wert des adressierten T/C geladen.

INI(27) iii Indexregister um 1 erhöhen (inkrementieren)
DEI(28) iii Indexregister um 1 erniedrigen (dekrementieren)

a) iii = Ø...255
   Das Indexregister wird bis zum vorgegebenen Wert iii in- bzw. dekrementiert.

b) iii = 256...319* (für PCA232: 256...511)
   Der vorgegebene Wert befindet sich im adressierten T/C.

Anmerkungen:
- Alle PCA verfügen über 16 Indexregister, je eines pro Parallelprogramm.
- Die max. Zählkapazität dieser Register beträgt 255. Dieser Wert darf, sofern er von einem Zählregister zugewiesen werden soll, nicht überschritten werden. Die maximale Kapazität des IR bei der PCA232 ist 1Ø23 (siehe Beispiel d).

Allgemeine Beispiele:

a) C267 = 1Ø2
   Nach dem Befehl SEI(16) 267 beträgt der Wert des Indexregisters ebenfalls 1Ø2.

b) C256 = 44
   Nach dem Befehl INI(27) 256 beträgt der Grenzwert, bis zu welchem inkrementiert wird, ebenfalls 44.

c) C26Ø = 1ØØ, IR = 4
   Nach dem Befehl SEI(16) 1256 beträgt der Wert des Indexregisters 1ØØ (doppelte Indexierung).

d) Sollen bei der PCA232 Werte > 255 ins Indexregister geladen werden, so kann dies indirekt über einen Zähler erfolgen.

```
z.B.        SCR  28Ø  }  Der Wert 8ØØ wird
             ØØ  8ØØ  }  ins Indexregister
            SEI  28Ø  }  geladen.
```

*) Registerstruktur siehe Seite III.

E 7 PAS-Instruktionen

PAS Ø -----> PAS 15 Zuweisung der Parallelprogramme (PP)

PAS: Program Assignment ---> Zuweisung des Parallelprogrammes

Befehlsformat (zweizeiliger Befehl):

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
PAS	29	Nummer des Programmes, fortlaufend von	Ø...15	1. Zeile
---	ØØ	Anfangsadresse des Programmes	Ø...819Ø	2. Zeile

Sollen mehrere Programme (max. 16) parallel ablaufen, so muss dies der CPU gleich am Anfang des Programmes "mitgeteilt" werden. Dies erfolgt durch die Zuweisung der Anfangsadressen aller abzuarbeitenden Parallelprogramme mit dem zweizeiligen Befehl PAS.

Die Auflistung der Parallelprogramme im Assignierungsteil soll lückenlos (ohne Auslassung von Nummern) ab Programm Nr. 1 in aufsteigender Reihenfolge geschehen. Dieser Assignierungsteil wird im allgemeinen nur einmal unmittelbar nach dem Einschalten der PCA durchlaufen.

PPs können mit dem Befehl PAS Ø...15 auch "umassigniert" werden. In einem beliebigen Programmteil kann z.B. PP3 von der Anfangsadresse 3ØØ auf Anfangsadresse 4ØØ umassigniert werden.

Das Ø. Parallelprogramm muss nicht angewiesen werden. Es wird nach der Zuweisung aller PP direkt angesprungen.

Der Aufbau und die Struktur der PP geht aus Abschnitt D 5.5 hervor. Die Abarbeitung der einzelnen PP erfolgt in einer Art "Time Sharing". Der Prozessor wechselt von einem PP zum nächsten nach genau vorgegebenen Bedingungen.

Nötige Bedingungen für PP-Wechsel

Folgende Instruktionen bewirken einen PP-Wechsel:

- WIH, WIL (sofern die Wartebedingung erfüllt ist)
- JMP, JIO, JIZ, JMS, RET
- und jeder zweite bzw. dritte STH- oder STL-Befehl

PAS 0...15 wird unabhängig vom ACCU immer ausgeführt und verändert den ACCU nicht.

Kapitel F
Beispiele: 13, 18

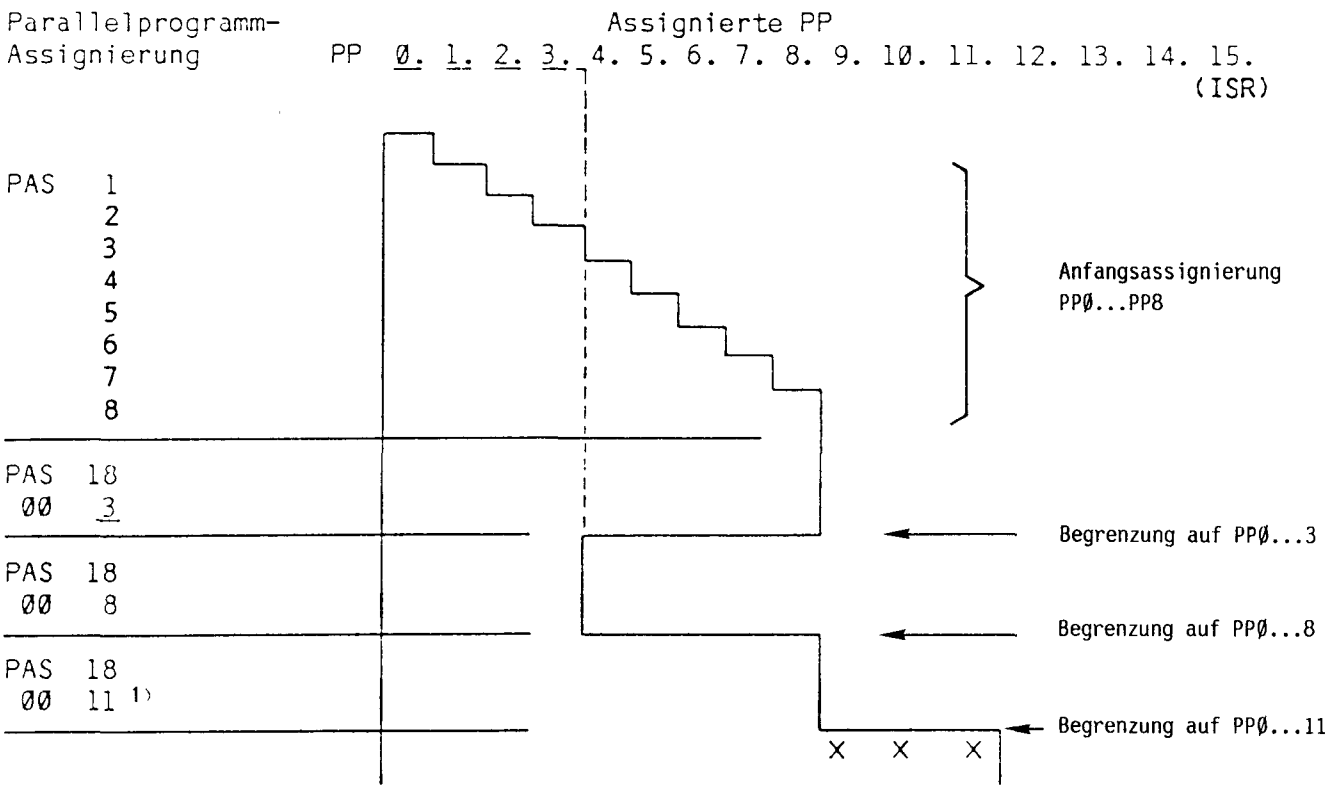
PAS 18 Begrenzung der assignierten Parallelprogramme

Alle SAIA°PLC erlauben es, bis zu 16 Parallelprogramme (PP) zu assignieren und parallel ablaufen zu lassen. Wurde ein PP nicht benötigt, so musste dies bisher durch Umassignierung auf eine leere Sprungschlaufe bewirkt werden. Dieses Vorgehen ergab aber keine Zeiteinsparung bei der Abarbeitung der verbleibenden PP.

Mit dem Befehl PAS 18 kann die Anzahl der aktivierten PP von oben her begrenzt werden. Nach der PP-Assignierung durch PAS Ø...max. 15 kann im Anwenderprogramm an beliebiger Stelle und beliebig oft die Anzahl der aktiven PP von oben her begrenzt werden mit dem Befehl PAS 18.

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
PAS	29	Immer 18	---	1. Zeile
---	ØØ	Begrenzung der PP von oben her	PP1...15	2. Zeile

Der Befehl PAS 18 wird immer ausgeführt (unabhängig vom ACCU) und verändert den ACCU nicht.



1) Wird durch PAS 18 eine höhere Zahl als die ursprünglich assignierten PP angegeben, so resultiert keine Fehlfunktion. Die leeren PP 9...11 benötigen jedoch Bearbeitungszeit im Systemprogramm.

PAS 3Ø
PAS 31...38

"Check-Sum" des System- und Anwenderprogrammes

Die "Check-Sum"-Funktion dient zur Prüfsummenbildung der Speicherinhalte des Systemprogrammes (PAS 3Ø) oder des Anwenderprogrammes (PAS 31...38). Damit kann überprüft werden, ob in den Speichern Inhaltsveränderungen stattgefunden haben.

Nach Ausführung des Befehls wird:
ACCU = 1 wenn die Bezugsgrösse mit der Prüfsumme übereinstimmt,
ACCU = Ø wenn die Bezugsgrösse mit der Prüfsumme nicht übereinstimmt.

Die Befehle PAS 3Ø...38 werden immer ausgeführt, unabhängig vom ACCU. Bei einer Inhaltsveränderung der Speicher kann der Anwender die ihm dafür notwendig erscheinenden Massnahmen programmieren: Auslösung eines Alarmes, Rücksetzen des Watchdog usw.

Prüfsummenbildung des Systemprogrammes

Mnemo-code	Zahlen-code	Operand	
PAS	29	Immer 3Ø	1. Zeile
---	ØØ	Immer Ø	2. Zeile

Prüfsummenbildung des Anwenderprogrammes

Befehlscode		Operand		
Mnemo-code	Zahlen-code	Beschreibung	Bereich	
PAS	29	Programm-teil 1.K...8.K *	31...38 *	1. Zeile
---	XX	Bezugsgrösse	xxxx	2. Zeile

Die Bezugsgrösse für das Anwenderprogramm erhält man durch Abarbeiten des entsprechenden PAS-Befehls in der Betriebsart STEP. Die PCA zeigt diese Bezugsgrösse auf dem Display des Programmiergerätes einige Sekunden lang an. In der Betriebsart PROG kann anschliessend der entsprechende Wert in der 2. Zeile eingegeben werden.

Achtung: Die Ausführungszeit für diese Befehle ist relativ lang:
PAS 3Ø ≈ 28,Øms, PAS 31...38 ≈ 8,3ms (PCA232: PAS 3Ø ≈ 13,6ms,
PAS 31...38 ≈ 9,5ms)

"Check-Sum" sollte daher nur ausgeführt werden, wenn der zu überwachende Ablauf dies zulässt, z.B. beim Einschalten der SPS, am Ende eines Ablaufzyklus usw.

---

*) Die Prüfsumme des Anwenderprogrammes wird für jedes "K" Programm separat durchgeführt (siehe Beispiel auf der folgenden Seite).

Es wird empfohlen, diesen Befehl erst in das Anwenderprogramm einzufügen, wenn dieses fertig entwickelt und getestet ist. Jede Programmänderung, egal ob es sich um eine Erweiterung oder Verkürzung handelt, führt zu einer Änderung der Prüfsumme und erfordert eine Änderung der Bezugsgrösse.

Beispiel: Beim Einschalten soll ein 2K-Anwenderprogramm überwacht werden.

```
20   PAS    31   } "Check-Sum"
21    Ø9   825   } 1. K
22   JIZ    35
23   PAS    32   } "Check-Sum"
24    Ø7  154Ø   } 2. K
25   JIZ    35

3Ø   COO   255   } Arbeitsprogramm mit
31   JMP    3Ø   } WD-Ueberwachung

35   SEO    15   } Alarmausgang setzen
36   JMP    35   } ausserhalb des Ar-
                 } beitsprogrammes
```

Vorgehen:
- Nach Programmeingabe und Prüfung Betriebsart STEP wählen
- [A] 23 [+] eintippen
  -> Die Bezugsgrösse für 2.K (PAS 32) erscheint für ca. 2Øs
- Bezugsgrösse in Betriebsart PROG eingeben:
  [A] 24 Bezugsgrösse [+]
- Gleiches Vorgehen für PAS 31

E 8 Anzeigebefehle

DOP Anzeige eines Operanden

DOP: Display Operand ---> Anzeige der im Operanden stehenden Zahl

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
DOP	3Ø	Beliebige anzuzeigende Zahl	Ø...2Ø47

DOP ist ein Hilfsbefehl, der vor allem zur Inbetriebnahme und Fehlerdiagnose eingesetzt wird. Ein Programm kann so gestaltet werden, dass beim Auftreten eines bestimmten Prozess-Zustandes oder -Fehlers eine Identifikations-Zahl angezeigt wird. Diese Anzeige erfolgt im "RUN"-Betrieb und zwar im Operand-Feld des Eingabegerätes oder auf einem Operand-Display.
Die Anzeige bleibt jeweils 1s stehen. Soll sie längere Zeit anstehen, so ist der Befehl DOP mindestens einmal pro Sekunde abzuarbeiten (vorwiegend in einem Umlaufprogramm).
DOP wird nur ausgeführt, wenn eine Verknüpfung nicht erfolgreich war (ACCU = Ø).

Kapitel F
Beispiel 14

DTC Anzeige des Zeit- oder Zählerstandes

DTC: Display Timer or Counter ---> Anzeige des Timer- oder Zählerstandes

Befehlsformat:

Befehlscode		Operand	
Mnemo-code	Zahlen-code	Beschreibung	Bereich
DTC	31	Adresse des Zeitgliedes oder Zählers	256 ... 319 * (i)

(i) = indexierbar

DTC ist ebenfalls ein wertvoller Hilfsbefehl zur Inbetriebnahme und Fehlerdiagnose.
Mit DTC kann auf dem Eingabegerät oder dem Operand-Display der PCA im RUN-Betrieb der Zeitablauf eines Zeitgliedes oder der Zählerstand eines Zählers angezeigt werden (Maximalanzeige 9999).
Die Anzeige bleibt jeweils 1s stehen. Soll sie längere Zeit anstehen, so ist der Befehl DTC mindestens einmal pro Sekunde abzuarbeiten (vorwiegend in einem Umlaufprogramm).
DTC wird nur ausgeführt, wenn ACCU = 1 ist.

*) Registerstruktur siehe Seite III.

Kapitel F
Beispiele: 8, 18

TEIL F PROGRAMMIERBEISPIELE

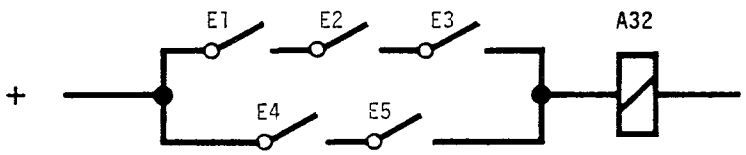
TEIL F PROGRAMMIERBEISPIELE

Nachdem im Teil E die Befehle Stufe 1H definiert worden sind, werden sie in der folgenden Beispielsammlung in Zusammenhang gebracht. Es handelt sich daher vorwiegend um typische Beispiele, um die Wirkungsweise und die Handhabung der einzelnen Befehle zu verdeutlichen. Das letzte Beispiel ist ein praxisbezogenes Gesamtprogramm.
Die E/A-Adressierung wurde so gewählt, dass alle Beispiele einzeln auf einer PCA1 bzw. PCA2 unter Zuhilfenahme eines Eingangssimuliergerätes PCA2.S1Ø nachvollzogen werden können. Die Adressierung ist daher wie folgt:

E = Ø...31
A = 32...63

Beispiel 1: UND/ODER-Verknüpfung nach Kontaktplan

1) Parallele Zweige



Schritt (STEP)	Anweisung Beschreibung	im Mnemo-code (numerisch)	wirkend auf Element oder Schritt	bzw. OPERAND
1Ø	Start mit Abfrage	STH (Ø1)	E1	1
11	UND-Verknüpfung	ANH (Ø3)	E2	2
12	UND-Verknüpfung	ANH (Ø3)	E3	3
13	ODER-Verknüpfung	ORH (Ø5)	E4	4
14	UND-Verknüpfung	ANH (Ø3)	E5	5
15	Resultat übertragen	OUT (1Ø)	A32	32
16	Rücksprung z. Anfang	JMP (2Ø)	STEP 1Ø	1Ø

SCHRITT WAS WO

1	2	3	4

STEP

1	2

CODE

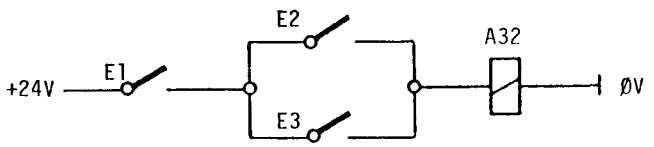
1	2	3	4

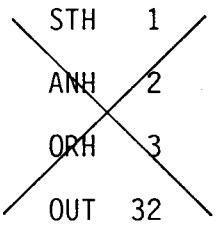
OPERAND

2) ODER hat Priorität vor UND

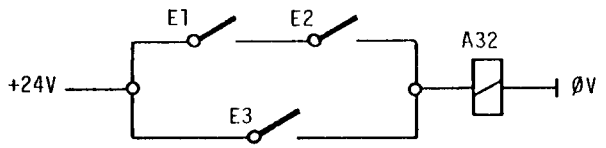
Bei der Kombination von ODER- und UND-Funktionen muss auf eine Besonderheit hingewiesen werden, die an diesem Beispiel erläutert werden soll.

Es sei die folgende Kontaktanordnung zu programmieren:



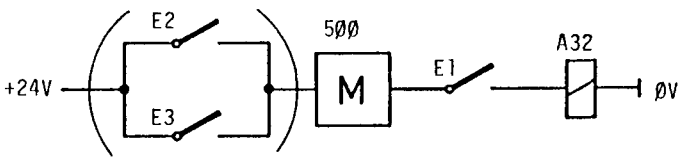


Der Schaltplan darf nicht so programmiert werden, da nach Definition die ODER-Funktion einer Parallelschaltung des folgenden Verknüpfungszweiges entspricht. Es würde sonst dem folgenden Schema entsprechen:



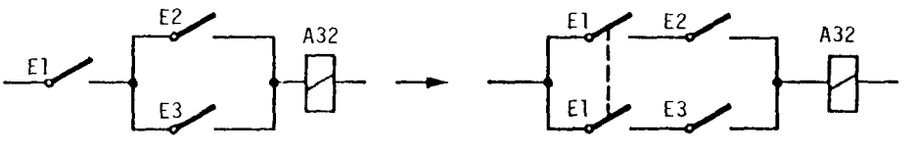
Die Schaltung kann auf 2 Arten programmiert werden:

- Durch Zwischenspeichern des ODER-Resultates




```
ADDR  NC   MNC   OPRD
  40  01   STH      2   E2 ABFRAGEN NACH H
  41  05   ORH      3   PARALLELSCHALTEN VON E3, ABGEFRAGT NACH H
  42  10   OUT    500   ZWISCHEN-ERGEBNIS IN EINEM MERKER ABSPEICHERN
--------------------
  43  01   STH    500   NEUE VERKNUEPFUNGSZEILE MIT ABFRAGE DES MERKERS
  44  03   ANH      1   UND - VERKNUEPFUNG MIT E1, ABGEFRAGT NACH H
  45  10   OUT     32   UEBERNAHME DES RESULTATES AUF A32
  46  20   JMP     40-->
```

- Durch Umzeichnen der Schaltung in eine direkt programmierbare Parallelschaltung mit durchgehenden parallelen Zweigen.



Der zweite E1-Kontakt dient nur der Programmierung. In Wirklichkeit ist E1 nur einmal vorhanden, im Programm wird er aber zweimal abgefragt.

```
ADDR NC   MNC   OPRD
  50 01   STH      1   START DES 1. VERKNUEPFUNGSZWEIGES
  51 03   ANH      2
  52 05   ORH      1   PARALLELSCHALTUNG UND START DES 2. VERKN.ZWEIGES
  53 03   ANH      3
  54 10   OUT     32
  55 20   JMP     50->
```

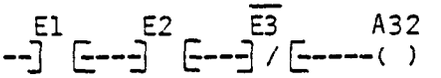
Beispiel 2: UND/ODER nach Logikplan

1) UND-Funktion

Logikplan:

Der Logikplan erlaubt die eindeutige Darstellung der logischen Verknüpfungen und der Signalzustände.

Kontaktplan:

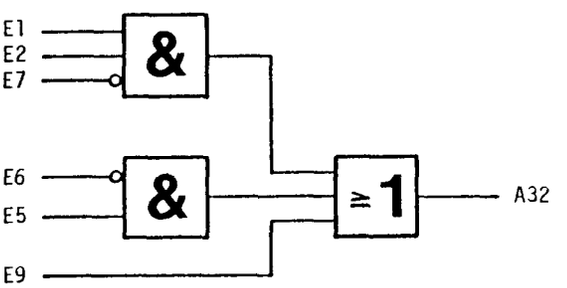


Die Darstellung der obenstehenden Funktion im Kontaktplan ist problematisch. Aus diesem Grund wird in der Praxis oft zu den symbolischen Kontakten Zuflucht genommen. Dabei ist die Verknüpfung erfüllt (A32 aktiv), wenn E1 und E2 = H und E3 = L ist.

```
ADDR NC   MNC   OPRD
  60 01   STH      1   ABFRAGE E1 NACH H
  61 03   ANH      2   UND - VERKNUEPFUNG ERFOLGREICH (ACCU=1)WENN E2 = H
  62 04   ANL      3   UND - VERKNUEPFUNG ERFOLGREICH (ACCU=1)WENN E3 = L
  63 10   OUT     32
  64 20   JMP     60->
```

Bei der Programmierung nach Logikplan werden die nichtinvertierten Elemente mit STH, ANH, ORH abgefragt und verknüpft, die invertierten mit STL, ANL, ORL.

2) UND/ODER kombiniert

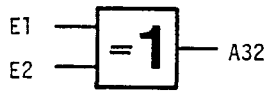



```
ADDR NC   MNC   OPRD
  70 01   STH      1
  71 03   ANH      2
  72 04   ANL      7
  73 06   ORL      6   NEUER, PARALLELER VERKNUEPFUNGSZWEIG
  74 03   ANH      5
  75 05   ORH      9   NEUER, PARALLELER VERKNUEPFUNGSZWEIG
  76 10   OUT     32
  77 20   JMP     70->
```

Beispiel 3: EXOR-Verknüpfungen

1) Vergleich auf ungleiche logische Zustände

Es sind zwei Eingänge zu vergleichen. Haben beide Eingänge den gleichen logischen Zustand, so muss der Ausgang = L sein. Sind sie ungleich, muss der Ausgang = H (aktiviert) sein.

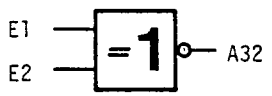



```
ADDR NC   MNC   OPRD
  80 01   STH      1   ABFRAGE E1
  81 07   XOR      2   EXOR-VERKNUEPFUNG MIT E2
  82 10   OUT     32
  83 20   JMP     80->
```

2) Vergleich auf gleiche logische Zustände

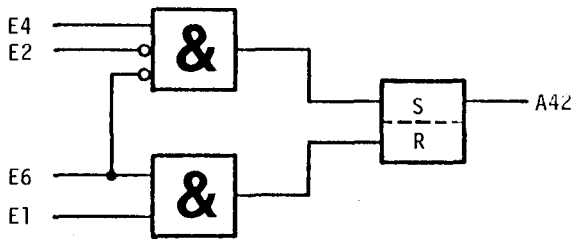
Wie Beispiel 3.1 aber

wenn Eingänge gleich ----> Ausgang = H (aktiviert)
wenn Eingänge ungleich ----> Ausgang = L




```
ADDR NC   MNC   OPRD
  85 01   STH      1
  86 07   XOR      2   EXOR-VERKNUEPFUNG
  87 08   NEG      0   UMKEHRUNG DES ACCU-INHALTES
  88 10   OUT     32
  89 20   JMP     85->
```

Beispiel 4: Verknüpfung im Funktionsplan



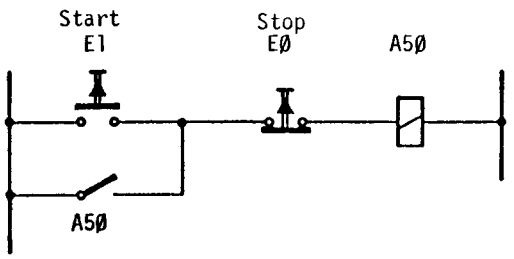

```
ADDR  NC   MNC   OPRD
  90  01   STH      4   ABFRAGE E4
  91  04   ANL      2   UND - VERKNUEPFUNG, ABGEFRAGT NACH L
  92  04   ANL      6   UND - VERKNUEPFUNG, ABGEFRAGT NACH L
  93  11   SEO     42   WENN VERKN. ERFOLGREICH, SETZE AUSGANG 42
--------------------
  94  01   STH      6   NEUE VERKNUEPFUNG, ABFRAGE E6
  95  03   ANH      1   UND - VERKNUEPFUNG
  96  12   REO     42   WENN VERKN. ERFOLGREICH, SETZE AUSG. 42 ZURUECK
  97  20   JMP     90-->
```

Wären beide Verknüpfungen erfolgreich, würde bei jedem Programmdurchlauf bei Schrittadresse 93 der Ausgang gesetzt und bei Adresse 96 zurückgesetzt, was ein unzulässiges Flackern dieses Schaltkreises ergäbe. Mit der Verriegelung über E6 wird dieser Zustand verhindert.

Beispiel 5: Start/Stop-Schaltung mit Selbsthaltung

1) Nach Kontaktplan

Aus der Schützenschalttechnik ist das folgende klassiche Beispiel bekannt:

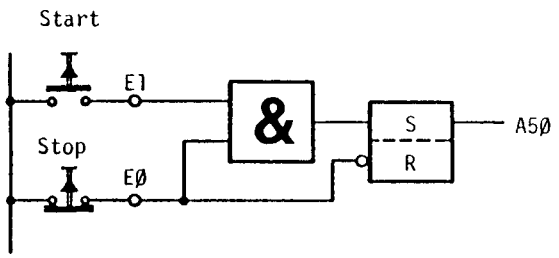


Programm:

```
STH      1
ORH      5Ø      } Start
OUT      4ØØ
---------------
STH      4ØØ
ANH      Ø       } Stop
OUT      5Ø
```

Sofern es die Hardware erlaubt (keine Einschübe C3Ø oder Interfaces B9Ø), kann der Ausgang A5Ø direkt in die Verknüpfung einbezogen werden. Wie das Programm zeigt, wird der Oeffnungskontakt EØ nach "H" verknüpft, da nur bei geschlossenem EØ A5Ø aktiviert werden kann.
Diese Programmierungsart ist auch sicher gegen Drahtbruch. Bricht nämlich ein Draht in den Leitungen von EØ, E1 oder A5Ø, so wird A5Ø immer stillgesetzt.

2) Nach Funktionsplan

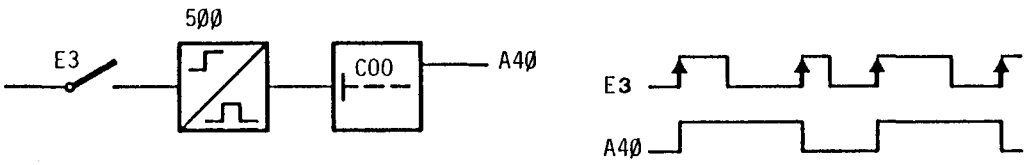


Programm:

```
STH      1
ANH      Ø       } Start
SEO      5Ø
---------------
STL      Ø       } Stop
REO      5Ø
```

Wie im Beispiel 4 ist auch hier der Setz-Befehl nur wirksam, wenn EØ = H ist. Werden beide Tasten gedrückt, so hat, wegen der UND-Verknüpfung, der Rücksetzbefehl Priorität.
Auch diese Programmierung ist in der vorliegenden Form drahtbruchsicher.

Beispiel 6: Impulsuntersetzer (Schrittschalter)

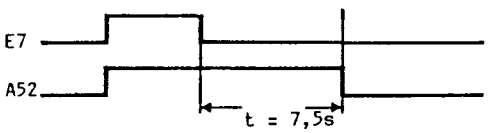



```
ADDR NC   MNC   OPRD
 100 01   STH      3 ┐ ABFRAGE E3
 101 09   DYN    500 │ FLANKENTRIGGERUNG, ABSPEICHERN IN MERKER 500
 102 13   COO     40 │ ABFRAGE VON A40 UND UMKEHRUNG SEINES ZUSTANDES
 103 20   JMP    100 ┘
```

Wäre in diesem Beispiel der DYN-Befehl weggelassen, so würde bei geschlossenem Schalter E3 bei jedem Programmdurchlauf der Ausgang 40 komplementiert, in dieser kleinen Schlaufe etwa 3000 mal pro Sekunde.

Mit DYN wird beim Schliessen von E3 nur der 1. Programmdurchlauf den Ausgang 40 komplementieren. Jeder weitere Durchlauf hat keine Wirkung mehr, bis sich der Signalzustand von E3 geändert hat und erneut eine Einschaltflanke entsteht (Schliessen des Kontakts).

Beispiel 7: Abfallverzögerung

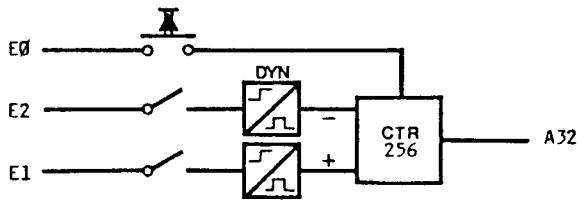


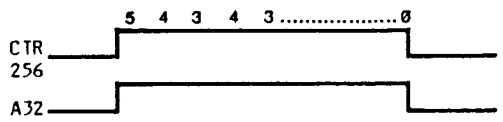

```
ADDR NC   MNC   OPRD
 104 01   STH      7  ABFRAGE E7
 105 14   STR    256  ZEITGLIED SETZEN          ) 2 - ZEILIGER
 106 00    00     75  ZEITEINGABE IN 1/10 SEC   )     BEFEHL
-------------------
 107 01   STH    256  ABFRAGE ZEITGLIED
 108 10   OUT     52  UEBERTRAGUNG AUF A52
 109 20   JMP    104 ->
```

Beim Schliessen von E7 wird das Zeitglied gesetzt und sein logischer Zustand wird "H". Die Zeit läuft erst ab, wenn E7 geöffnet wird.
(Genau genommen beginnt die Zeit sofort abzulaufen. Da aber bei geschlossenem E7 im nächsten Programmdurchlauf nach einigen 100 µs das Zeitglied wieder gesetzt wird, beginnt der Zeitablauf wieder von vorn, bis das Signal an E7 weggenommen, d.h. der Schalter E7 geöffnet wird).

Wird während dem Zeitablauf E7 wieder geschlossen, wird das Zeitglied nachgeschaltet (rückgesetzt und wieder gestartet).

Beispiel 8: Auf-/Abwärtszähler
(mit Anzeige des Zählerstandes auf dem Eingabegerät
oder auf einem Display-Modul)



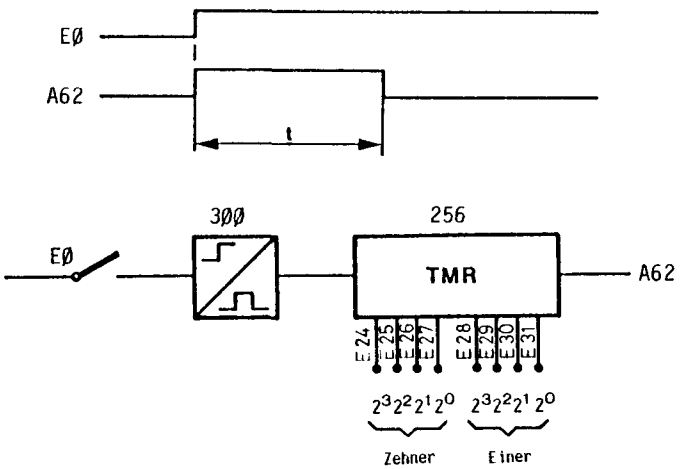



```
ADDR NC   MNC   OPRD
 110 01   STH      0  ABFRAGE E0
 111 15   SCR    256  ZAEHLER SETZEN                ) 2 - ZEILIGER
 112 00    00      5  ZAEHLWERT                     )       BEFEHL
--------------------
 113 01   STH      1  ABFRAGE E1
 114 09   DYN    500  FLANKENTRIGGERUNG, ABSPEICHERN IN MERKER 500
 115 17   INC    256  +1
--------------------
 116 01   STH      2  ABFRAGE E2
 117 09   DYN    501  FLANKENTRIGGERUNG, ABSPEICHERN IN MERKER 501
 118 18   DEC    256  -1
--------------------
 119 01   STH    256  ABFRAGE ZAEHLER (H SOLANGE INHALT >0 )
 120 10   OUT     32  UEBERNAHME DES ZAEHLERZUSTANDES AUF A32
--------------------
 121 19   SEA      0  ACCU = 1 SETZEN
 122 31   DTC    256  ANZEIGE DES ZAEHLERINHALTES AUF OPERANDFELD
 123 20   JMP    110 ->
```

Für DTC muss der ACCU = 1 sein, da ansonsten die Zählerstandsanzeige unterbleibt. Entweder wird (wie im Beispiel) SEA vor DTC gesetzt oder der DTC-Befehl wird an den Anfang der Umlaufschleife genommen, da nach Ausführung des JMP-Befehls der ACCU immer 1 ist.

Beispiel 9: Timer einschaltwischend mit externer Zeiteingabe im BCD-Code

Zeitbereich 1...99 sec. extern einstellbar mit zwei BCD-Schaltern. Eingänge E24...31 vom BCD-Schalter.



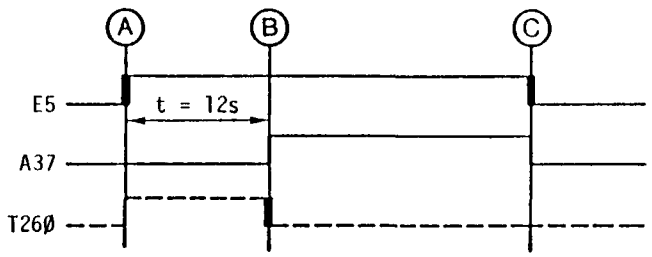
BCD-Tabelle

Binärsignale an 4 Eingängen (BCD-Schalter)				
E28	E29	E3Ø	E31	Dezimal-wert
$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	
L	L	L	L	Ø
L	L	L	H	1
L	L	H	L	2
L	L	H	H	3
L	H	L	L	4
L	H	L	H	5
L	H	H	L	6
L	H	H	H	7
H	L	L	L	8
H	L	L	H	9

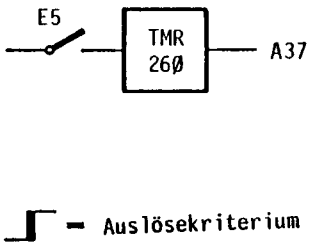
```
ADDR NC   MNC   OPRD
 130 01   STH     0   ABFRAGE EO
 131 09   DYN   300   FLANKENTRIGGERUNG,TIMER WIRD NUR IM 1.ZYKL.GESETZT
 132 14   STR   256   TIMER SETZEN
 133 17    17    31   EXTERNWERT * 10 * 1/10SEC. ADR. N AUF E31
------------------
 134 01   STH   256   ABFRAGE ZEITGLIED
 135 10   OUT    62   UEBERTRAGUNG AUF A62
 136 20   JMP   130->
```

Beispiel 1Ø: Einschaltverzögerung nach Flussdiagramm

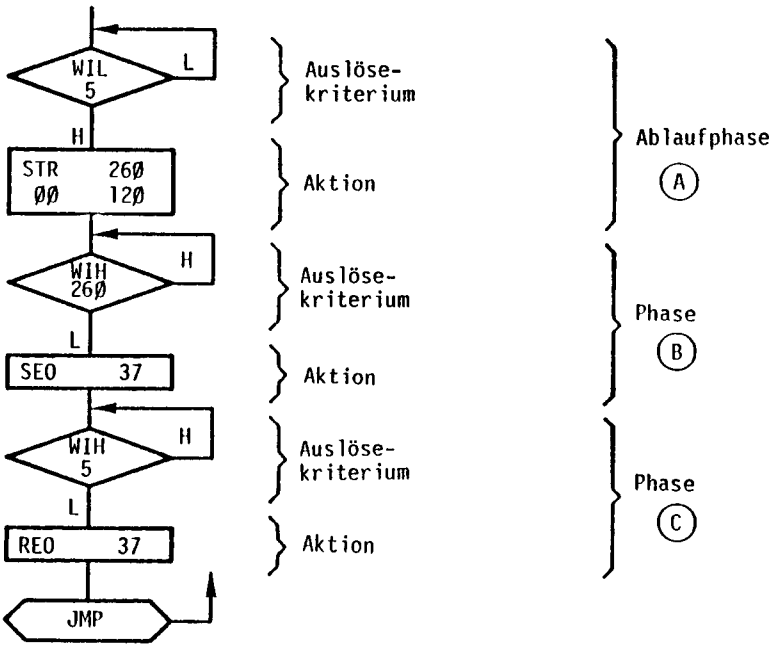
Ablaufdiagramm:



Kontaktplan:



Das sequentielle Zeitschaltdiagramm kann in verschiedene Ablaufphasen unterteilt werden. Zu jeder Phase gehört ein entsprechendes Auslösekriterium. Mit einer Warteschleife wird jeweils gewartet, bis die entsprechende Bedingung erfüllt ist, um die nachfolgende Funktion auszuführen.

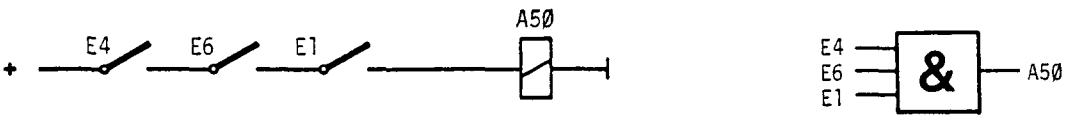


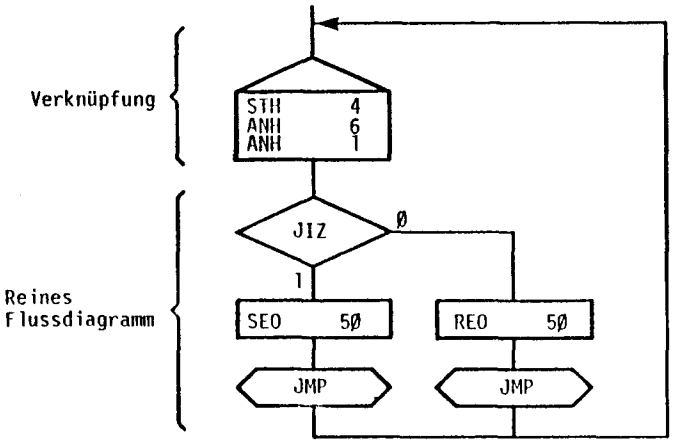

```
ADDR  NC   MNC   OPRD
140   26   WIL      5   AUSLOESE-KRITERIUM    ) ABLAUF-PHASE
141   14   STR    260   AKTION                ) A
142   00    00    120
143   25   WIH    260   AUSLOESE-KRITERIUM    ) PHASE
144   11   SEO     37   AKTION                ) B
145   25   WIH      5   AUSLOESE-KRITERIUM    ) PHASE
146   12   REO     37   AKTION                ) C
147   20   JMP    140 ->
```

Beispiel 11: UND-Verknüpfung im Flussdiagramm

Kombination von Verknüpfung und Flussdiagramm (ohne Warteschleifen)

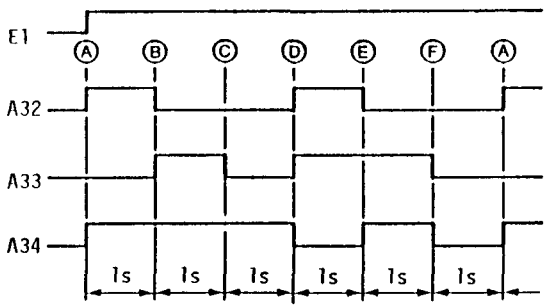
Vorlagen im Kontakt- bzw. Logikplan:

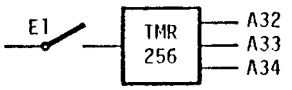


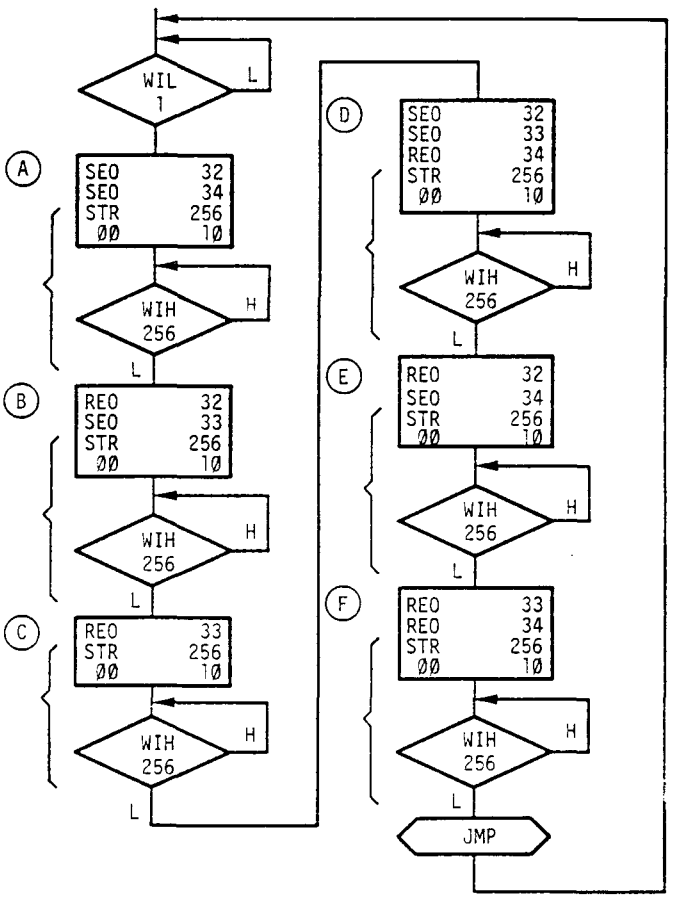



```
 ADDR NC   MNC   OPRD
→150 01   STH     4   ABFRAGE E4
 151 03   ANH     6   UND-VERKNUEPFUNG
 152 03   ANH     1   UND-VERKNUEPFUNG
┌153 22   JIZ   156-> WENN 0, SPRINGE AUF SCHRITTADRESSE 156
│154 11   SEO    50  WENN 1, SETZE AUSGANG 50
└155 20   JMP   150-> RUECKSPRUNG AUF PROGRAMMANFANG
 →156 12  REO    50  SETZE AUSGANG 50 ZURUECK
 157 20   JMP   150-> RUECKSPRUNG AUF PROGRAMMANFANG
```

Beispiel 12: Ablaufsteuerung ohne und mit Unterprogramm (Subroutine)

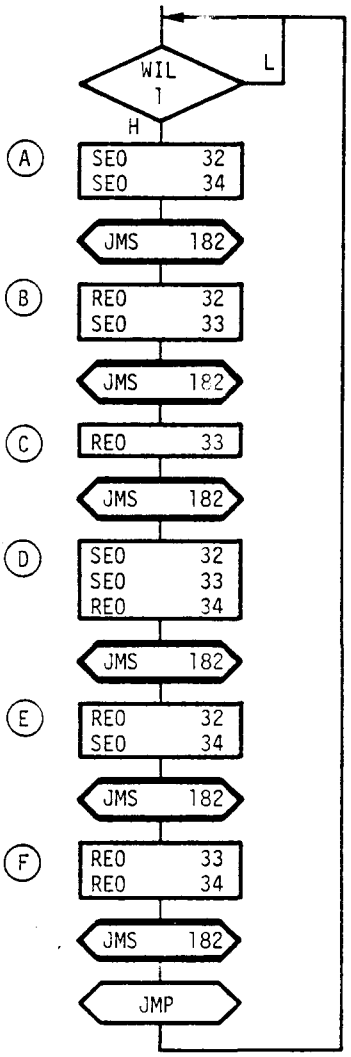






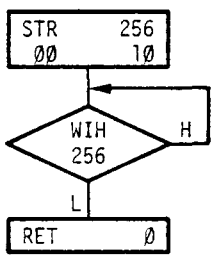
Das Flussdiagramm zeigt den Programmablauf ohne Unterprogramm. Die Programmteile, die mit der Klammer bezeichnet sind, wiederholen sich 6 mal und können somit vorteilhaft als Unterprogramm (Subroutine) geschrieben werden.

Hauptprogramm




```
******************** HAUPT-PROGRAMM
ADDR  NC   MNC   OPRD
 160  26   WIL     1   WARTE, WENN E1 OFFEN
 161  11   SEO    32
 162  11   SEO    34
 163  23   JMS   182=>  SPRINGE ZUR SUBR. 182
 164  12   REO    32
 165  11   SEO    33
 166  23   JMS   182=>  SPRINGE ZUR SUBR. 182
 167  12   REO    33
 168  23   JMS   182=>  SPRINGE ZUR SUBR. 182
 169  11   SEO    32
 170  11   SEO    33
 171  12   REO    34
 172  23   JMS   182=>  SPRINGE ZUR SUBR. 182
 173  12   REO    32
 174  11   SEO    34
 175  23   JMS   182=>  SPRINGE ZUR SUBR. 182
 176  12   REO    33
 177  12   REO    34
 178  23   JMS   182=>  SPRINGE ZUR SUBR. 182
 179  20   JMP   160->
```

Subroutine "182"



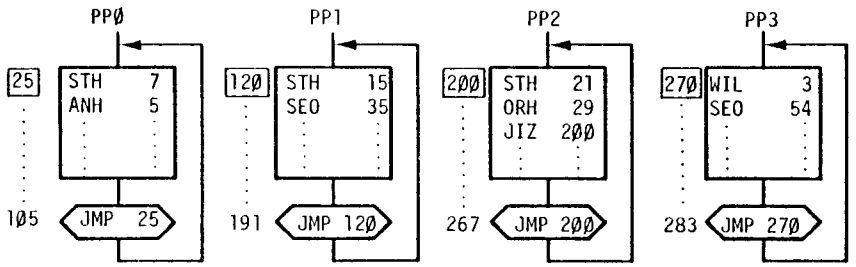

```
==================== SUBROUTINE 182 (1 SEK. WARTEN)
 182  14   STR   256
 183  00    00    10
 184  25   WIH   256
 185  24   RET     0->
```

Beispiel 13: Arbeiten mit Parallelprogrammen

Das Programm zur Steuerung einer Vorrichtung mit 2 Manipulatoren und einem Rundtisch sei in 4 Parallelprogramme aufgeteilt.

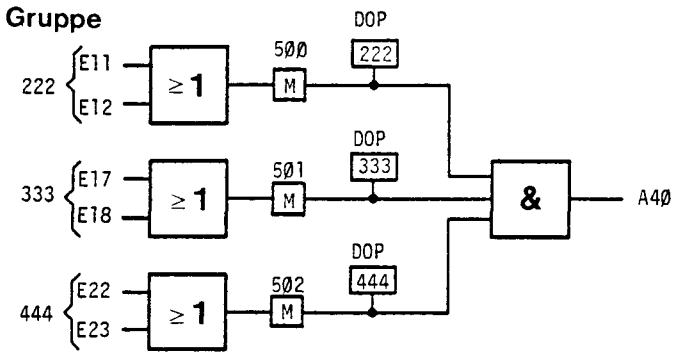
Parallelprogramm Ø	Ueberwachung	Anfangsschrittadresse 25
Parallelprogramm 1	Manipulator 1	Anfangsschrittadresse 12Ø
Parallelprogramm 2	Manipulator 2	Anfangsschrittadresse 2ØØ
Parallelprogramm 3	Rundtisch	Anfangsschrittadresse 27Ø

Parallelprogramm




```
******************** ASSIGNIERUNG DER PARALLEL-PROGRAMME
ADDR  NC   MNC   OPRD
   0  29   PAS      1   ZUWEISUNG PP1
   1  00    00    120
   2  29   PAS      2   ZUWEISUNG PP2
   3  00    00    200
   4  29   PAS      3   ZUWEISUNG PP3
   5  00    00    270
   6  20   JMP     25 ->  PPO WIRD DIREKT ANGESPRUNGEN

+++++++++++++++++++ PARALLEL-PROGRAMM 0 (UEBERWACHUNG)
 ┌->25  01   STH      7
 |   .   .    .       .
 |   .   .    .       .
 |   .   .    .       .
 └ 105  20   JMP     25 ->

+++++++++++++++++++ PARALLEL-PROGRAMM 1 (MANIPULATOR 1)
 ┌->120  01   STH     15
 |    .   .    .       .
 |    .   .    .       .
 |    .   .    .       .
 └ 191  20   JMP    120 ->

+++++++++++++++++++ PARALLEL-PROGRAMM 2 (MANIPULATOR 2)
 ┌->200  01   STH     21
 |    .   .    .       .
 |    .   .    .       .
 |    .   .    .       .
 └ 267  20   JMP    200 ->

+++++++++++++++++++ PARALLEL-PROGRAMM 3 (RUND-TISCH)
 ┌->270  01   STH      3
 |    .   .    .       .
 |    .   .    .       .
 |    .   .    .       .
 └ 283  20   JMP    270 ->
```

Beispiel 14: Ueberwachungsschaltung mit Fehleranzeige auf dem Eingabegerät



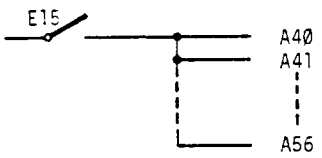
Die Eingänge der Gruppen 222, 333, 444 werden überwacht. Wenn eine Bedingung nicht erfüllt ist, löscht die Lampe an Ausgang 40. Auf dem Eingabegerät wird dann die fehlerhafte Gruppe angezeigt.

```
ADDR NC   MNC   OPRD
 300 01   STH     11
 301 05   ORH     12
 302 10   OUT    500
 303 30   DOP    222   ANZEIGE, WENN VERKN. NICHT ERFUELLT (ACCU=0)
--------------------
 304 01   STH     17
 305 05   ORH     18
 306 10   OUT    501
 307 30   DOP    333   ANZEIGE, WENN VERKN. NICHT ERFUELLT
--------------------
 308 01   STH     22
 309 05   ORH     23
 310 10   OUT    502
 311 30   DOP    444   ANZEIGE, WENN VERKN. NICHT ERFUELLT
--------------------
 312 01   STH    500
 313 03   ANH    501
 314 03   ANH    502
 315 10   OUT     40
 316 20   JMP    300->
```

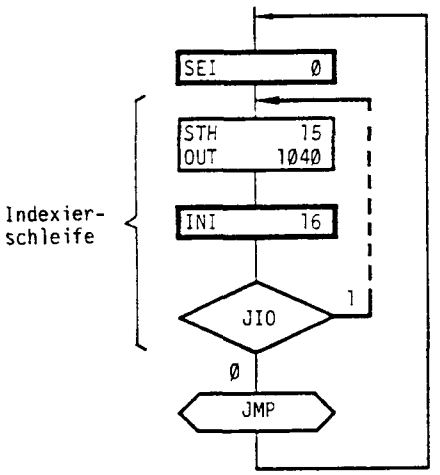
Beispiel 15: Reihenaktivierung

1. Beispiel mit Adress-Indexierung.

Die Ausgänge 4Ø...56 werden über Eingang 15 ein- bzw. ausgeschaltet.



Anzahl Indexschritte = 56 - 4Ø = 16



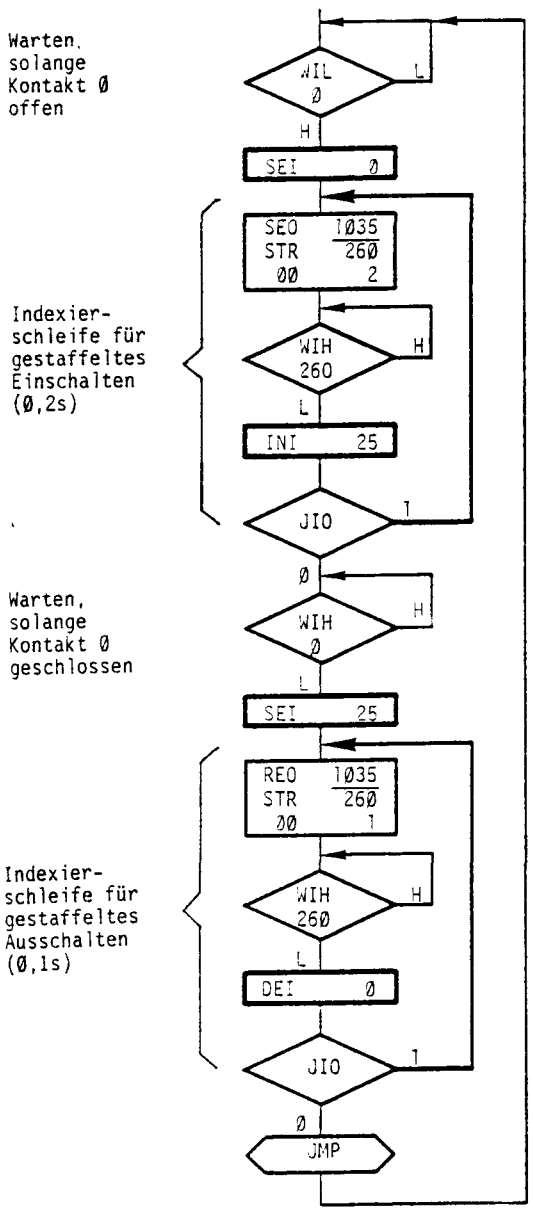

```
ADDR NC   MNC   OPRD
 350 16   SEI      0   SETZE INDEXREGISTER AUF ANFANGSWERT 0
 351 01   STH     15   ABFRAGE EINGANG 15
 352 10   OUT   1040   SETZEN DER INDEXIERTEN AUSG.(40 + 1000 = 1040)
 353 27   INI     16   INKREMENTIEREN DES INDEXREGISTERS BIS ENDWERT 16
 354 21   JIO    351-> WIEDERHOLEN DER INDEXIERUNG BIS ALLE AUSGAENGE
 355 20   JMP    350->                          A40...A56 BEHANDELT SIND
```

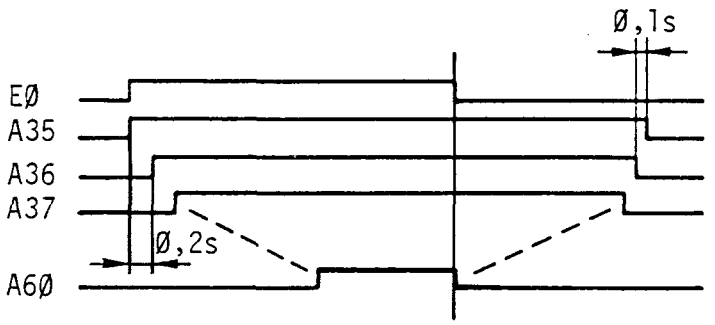
Beispiel 16: Lauflichter

2. Beispiel mit Adress-Indexierung.

Wenn Eingang Ø geschlossen wird, sollen nacheinander im Ø,2 sec.-Takt die Ausgänge 35...6Ø einschalten.

Wenn Eingang Ø wieder geöffnet wird, sollen nacheinander im Ø,1 sec.-Takt die Ausgänge 6Ø...35 ausschalten.

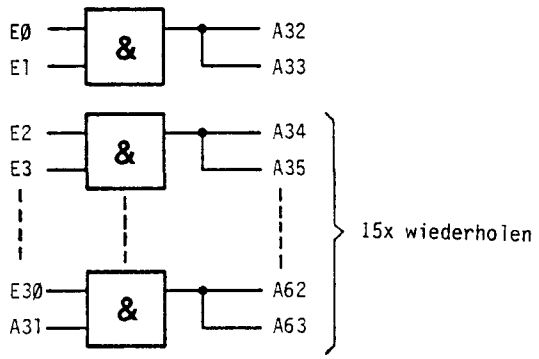





```
ADDR NC   MNC   OPRD
-------------------- EINSCHALTPHASE
 360 26   WIL      0
 361 16   SEI      0
 362 11   SEO   1035   35 + 1000 = 1035
 363 14   STR    260
 364 00    00      2
 365 25   WIH    260
 366 27   INI     25   60 -   35 =   25
 367 21   JIO    362->
-------------------- AUSSCHALTPHASE
 368 25   WIH      0
 369 16   SEI     25   60 -   35 =   25
 370 12   REO   1035   35 + 1000 = 1035
 371 14   STR    260
 372 00    00      1
 373 25   WIH    260
 374 28   DEI      0
 375 21   JIO    370->
 376 20   JMP    360->
```

Beispiel 17: Kleine Ueberwachungsschaltung

3. Beispiel mit Adress-Indexierung



Grundprogramm:

STH	Ø
ANH	1
OUT	32
OUT	33

Bei diesem Beispiel wird das ganze Grundprogramm indexiert.

```
 ADDR  NC   MNC   OPRD
   380 16   SEI      0
┌─►381 01   STH   1000
|  382 03   ANH   1001
|  383 10   OUT   1032
|  384 10   OUT   1033
|  385 27   INI     30   )2-MALIGES INKREMENTIEREN, DA INDEX JEWEILS UM
|  386 27   INI     30   )ZWEI SCHRITTE ERHOEHT WERDEN MUSS (62 - 32 = 30)
└─ 387 21   JIO    381->
   388 20   JMP    380->
```

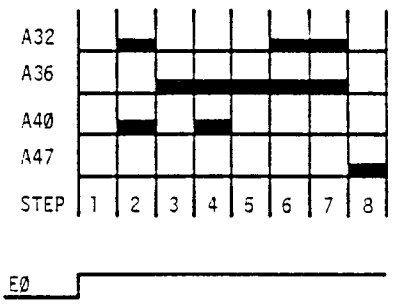
Beispiel 18: Umlaufender Programmschalter

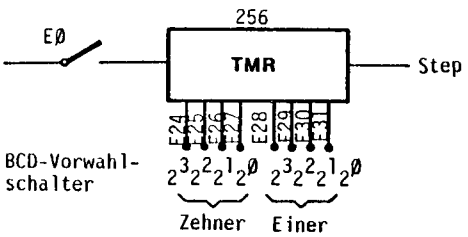
- mit veränderlicher Geschwindigkeit
- mit Hand-/Auto-Betrieb
- mit Schrittanzeige

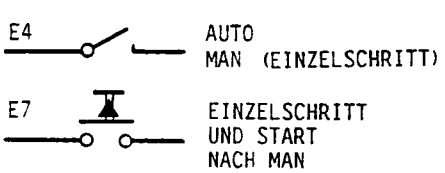
Aufgabe: Durch Schliessen von EØ soll der Programmschalter gemäss Diagramm ablaufen. Die Schrittzeit wird an den BCD-Schaltern E24...31 von Ø,1...9,9 sec. eingestellt. Ist der Schalter E4 geschlossen, so kann über die Taste E7 im Einzelschritt vorgegangen werden. Die Schrittnummer soll im Operanddisplay angezeigt werden.

Lösung: Das Ablaufprogramm ist in den Adressen 4Ø2...425 enthalten. Die periodische Abfrage auf Handbetrieb und das Abwarten der Schrittzeit sind in der Subroutine 43Ø untergebracht. Der Zähler 28Ø wird für die Schrittzählung verwendet. Damit die Anzeige dauernd vorhanden ist, wurde das kleine Parallelprogramm 45Ø assigniert.

Diagramm:







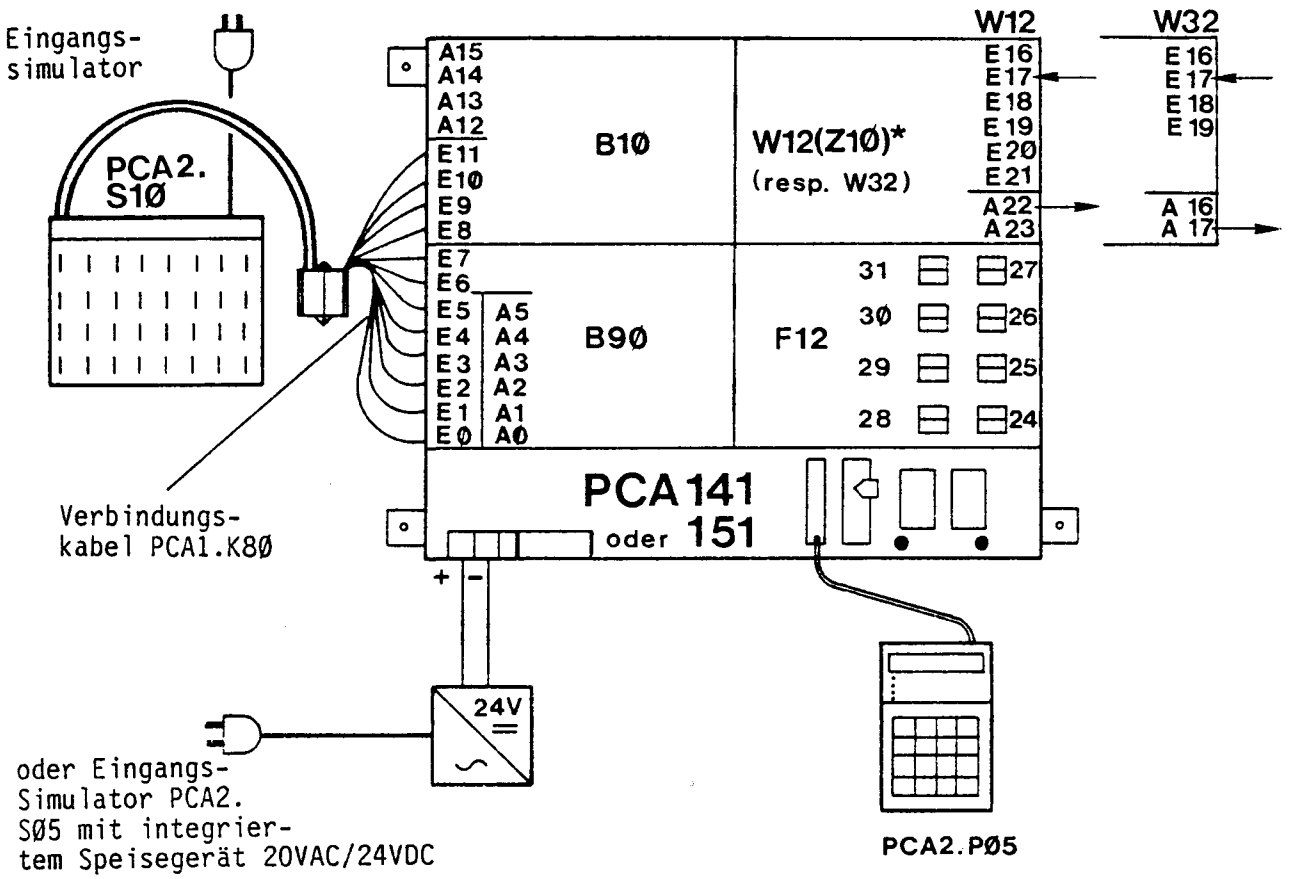

```
******************** ASSIGNIERUNG von :
 400 29  PAS     1   PARALLEL-PROGRAMM 1
 401 00   00   450
******************** HAUPTPROGRAMM PPO
 402 26  WIL     0   START PROGRAMMABLAUF
 403 15  SCR   280   STEP-ZAEHLER
 404 00   00     1
-------------------- STEP 1
 405 23  JMS   430=>  SPRINGE ZUR SUBR. 430
-------------------- STEP 2
 406 11  SEO    32
 407 11  SEO    40
 408 23  JMS   430=>
-------------------- STEP 3
 409 12  REO    32
 410 12  REO    40
 411 11  SEO    36
 412 23  JMS   430=>
-------------------- STEP 4
 413 11  SEO    40
 414 23  JMS   430=>
-------------------- STEP 5
 415 12  REO    40
 416 23  JMS   430=>
-------------------- STEP 6
 417 11  SEO    32
 418 23  JMS   430=>
-------------------- STEP 7
 419 23  JMS   430=>
-------------------- STEP 8
 420 12  REO    32
 421 12  REO    36
 422 11  SEO    47
 423 23  JMS   430=>
--------------------
 424 12  REO    47
 425 20  JMP   402->  START

 ==================== SUBROUTINE 430
 430 02  STL     4   MAN / AUTO
 431 22  JIZ   437->
-------------------- VARIABLER TIMER AUTO
 432 14  STR   256
 433 16   16    31
 434 25  WIH   256
 435 17  INC   280
 436 24  RET     0->
-------------------- MAN - BETRIEB
 437 26  WIL     7
 438 25  WIH     7
 439 17  INC   280
 440 24  RET     0->

 +++++++++++++++++++ PP1 : DISPLAY
 450 31  DTC   280
 451 20  JMP   450->
```

Programmier-Beispiele mit Analog-Modulen.

Die E/A-Adressierungen wurden so gewählt, dass alle Beispiele mit derselben Modul- und Geräte-Konfiguration nachvollzogen werden können.

Für alle Beispiele sind Programme sowohl für das 8-Bit Modul PCA1.W12 als auch für das 12-Bit Modul PCA1.W32 aufgeführt, welche alternativ eingesetzt werden können.

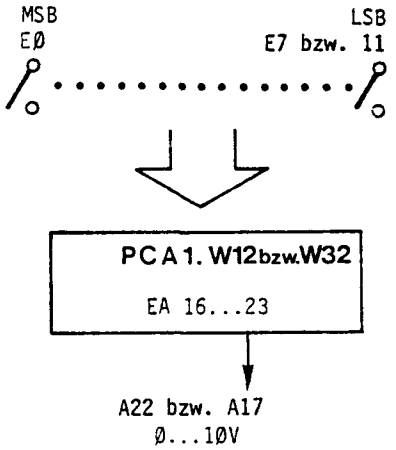
Diese Konfiguration sieht wie folgt aus:



*) Das Modul PCA1.W12 Z1Ø erlaubt Eingangsspannungen von Ø...1ØV (anstatt Ø...5V).

Beispiel 19: Analogspannung ausgeben ab 8 bzw. 12 Eingängen

Der Binärwert gebildet aus 8 bzw. 12 Eingängen EØ...E7 bzw. E11 ist am Analogausgang Kanal A22 bzw. A17 auszugeben.



Mit Modul PCA1.W12 (8 Bit bzw. 7 Bit)

```
4Ø    DTC   3Ø1    Binärwert anzeigen
      SCR   3Ø1  } Eingänge EØ...7 auf Zähler 3Ø1
       24    7   }
      SCR   3Ø1  } Inhalt von Zähler 3Ø1 auf
       21   23  } Ausgangskanal A22 analog
      SEO   23  } ausgeben
      ORH   22  }
      REO   23  }
      JMP   4Ø
```

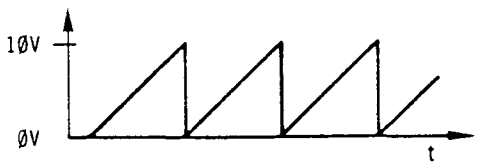
Mit Modul PCA1.W32 (12 Bit)

```
5Ø    DTC   3Ø1    Binärwert von C3Ø1 anzeigen
      SCR   3Ø1  } Uebertragung der
       25   11  } Eingänge auf C3Ø1
 1)   JMS   7ØØ    --> SR "Analogwert-
      JMP   5Ø     Ausgabe"
```

1) Unterprogramme siehe Handbuch "Hardware Baureihe PCA1" Modul PCA1.W32 oder W2..

Beispiel 2Ø: Sägezahnspannung ausgeben

Durch Inkrementieren eines Binärwertes von 8 bzw. 12 Bit soll eine Sägezahnspannung generiert werden.



Für W12 ist t durch den Timer 256 gegeben.
Für W32 ist t (für 4Ø96) Durchläufe) abhängig von der gewählten Subroutine und der Anzahl der assignierten Parallelprogramme.

Mit Modul PCA1.W12 (8 bzw. 7 Bit)

```
6Ø   SEI     7   }
  ->COO   15ØØ   }
   STH    15ØØ   } Binärzähler
  JIO       66   } inkrementieren
  DEI        Ø   }
  JIO       61   }
  SCR       3Ø1  } Binärwert auf
   24       5Ø7  } C3Ø1 laden
  SCR       3Ø1  }
   21       23   } Inhalt von C3Ø1 auf
   SEO      23   } Ausgangskanal A22
   ORH      22   } analog ausgeben
   REO      23   }
   STR     256   }
   ØØ        4   } Zeit von Ø,Ø4s warten
   WIH     256   }
  JMP       6Ø   }
```

Mit Modul PCA1.W32 (12 Bit)

```
8Ø   SEI     11  }
  ->COO   15ØØ   }
   STH    15ØØ   } Binärzähler
  JIO       86   } inkrementieren
  DEI        Ø   }
  JIO       81   }
  SCR       3Ø1  } Binärwert auf
   25       511  } C3Ø1 laden
 1) JMS     7ØØ    --> SR "Analogwert-Ausgabe"
  JMP       8Ø     (auch JMS 6ØØ oder 65Ø möglich)
```

---

1) Unterprogramme siehe Handbuch "Hardware Baureihe PCA1" Modul PCA1.W32 oder W2..

Beispiel 21 : BCD-Werte von Modul PCA1.F12 einlesen und analog ausgeben
von den Modulen PCA1.W12 (8 Bit) bzw. W32 (12 Bit).

Der 2-stellige BCD-Wert von A24 (von F12) soll eingelesen und alle 2s als Analogspannung über Kanal A22 bzw A17 des Analogmoduls ausgegeben werden. Der entsprechende Binärwert soll im Operand-Display angezeigt werden.

5	Ø

5,ØV ≙ Binärwert 128 (bzw. 2Ø48)
≙ 1Ø,ØV ≙ Binärwert 256 (bzw. 4Ø96)

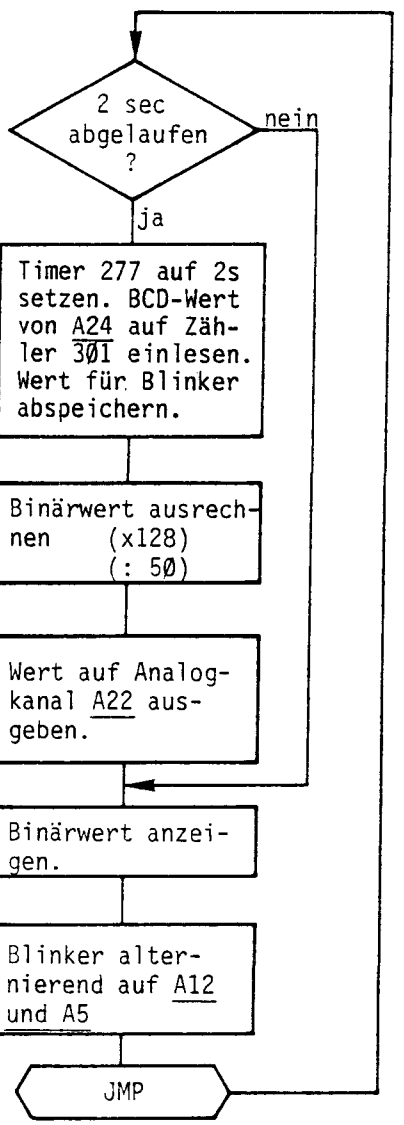
Die Ausgänge A5 und A12 sollen gemäss dem eingestellten BCD-Wert (mal 1/1Øs) alternierend blinken.

Programm
für PCA1.W12 (8 bzw. 7 Bit)

```
2ØØ  Ø1   STH    277
2Ø1  21   JIO    227

2Ø4  14   STR    277
2Ø5  ØØ     ØØ     2Ø
2Ø6  11   SEO      24
2Ø7  15   SCR    3Ø1
2Ø8  16     16     31
2Ø9  12   REO      24
21Ø  15   SCR    3Ø2
211  31     31   3Ø1

214  15   SCR    3Ø1
215  29     29   128
216  15   SCR    3Ø1
217  3Ø     3Ø     5Ø

22Ø  15   SCR    3Ø1
221  21     21     23
222  11   SEO      23
223  Ø5   ORH      22
224  12   REO      23

227  31   DTC    3Ø1

23Ø  Ø2   STL    278
231  14   STR    278
232  31     31   3Ø2
233  13   COO      12
234  Ø2   STL      12
235  1Ø   OUT       5
236  2Ø   JMP    2ØØ
```

*

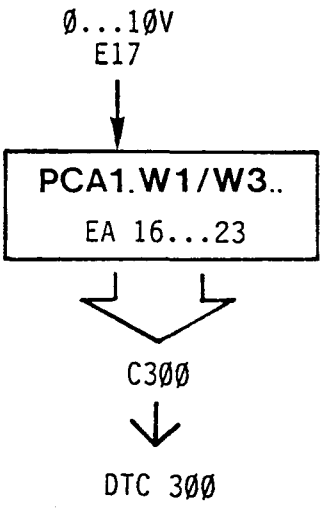
Flussdiagramm



*) Für PCA1.W32 (12 Bit)

```
214   15   SCR    3Ø1   } BCD-Wert x 41
215   29     29     41  } (1ØV ≙ 1ØØ ≙ 4Ø96 bzw. 41ØØ)
216   23   JMS    7ØØ  --> Unterprogramm siehe Hardwaremodul PCA1.W32 oder W2..
```

Beispiel 22: Einlesen einer Analogspannung in ein Zähler-Register und Anzeige des Binärwertes mit DTC



Mit Modul PCA1.W1.. (8 Bit)

```
9Ø   OUT   17   } Analogwert einlesen
     SCR   3ØØ   auf Zähler C3ØØ
      24    23   }
     DTC   3ØØ   Binärwert anzeigen
     JMP    9Ø
```

Mit Modul PCA1.W3.. (12 Bit)

```
1ØØ  JMS   5ØØ  ⇒Analogwert einlesen *)
                  auf Zähler C3ØØ *)
     DTC   3ØØ   Binärwert anzeigen
     JMP   1ØØ
```

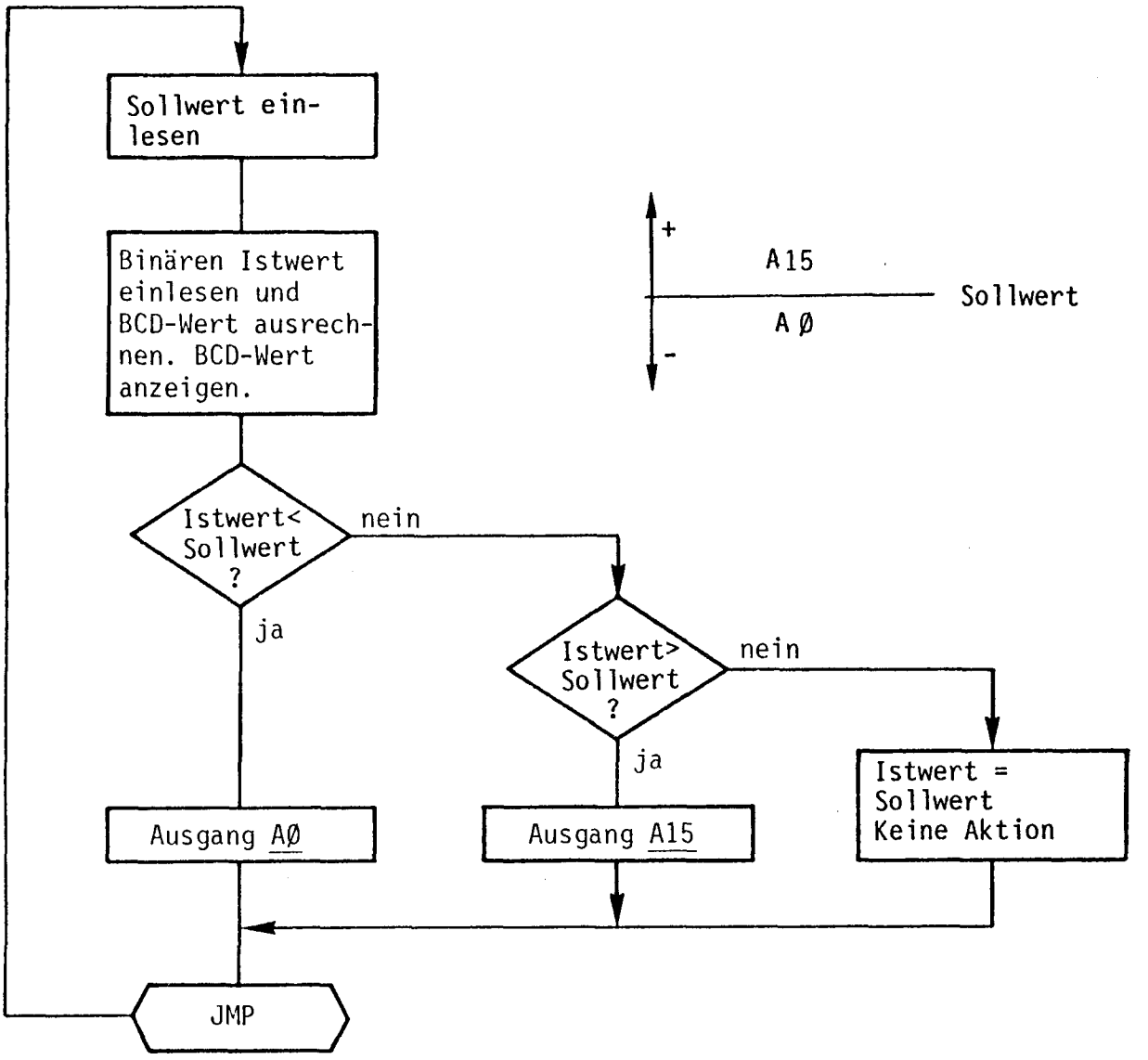
*) Unterprogramme siehe Handbuch "Hardware Baureihe PCA1 (Modul PCA1.W32)

Beispiel 23: Zweipunkt- und Dreipunktregler (mit Analogmodul PCA1.W1.. bzw. W3..)

Aufgabe 23a: Zweipunktregler

Ueber den 2-stelligen BCD-Schalter von Kanal A24 soll ein Sollwert vorgegeben werden. Der Istwert wird über Kanal 17 des Analogmoduls eingelesen. Befindet sich der Istwert unterhalb des vorgewählten Sollwertes, so wird Ausgang AØ, beim Ueberschreiten Ausgang A15 aktiviert. Der Istwert soll im Operand-Display als BCD-Wert (ØØ...99 ≙ Ø...9,9V) angezeigt werden.

Flussdiagramm



Lösung 23a:

```
400   11   SEO    24
401   15   SCR    310
402   16    16    31
403   12   REO    24

406   10   OUT    17
407   15   SCR    312
408   24    24    23

*411  15   SCR    312
412   29    29    50
413   15   SCR    312
414   30    30    128
415   31   DTC    312

418   15   SCR    310
419   28    28    312
420   22   JIZ    430   ;
421   01   STH    310
422   21   JIO    435   ;

425   12   REO     0
426   12   REO    15
427   20   JMP    400

430   12   REO     0
431   11   SEO    15
432   20   JMP    400

435   12   REO    15
436   11   SEO     0
437   20   JMP    400
```

Sollwert als BCD-Wert auf C310

Istwert als Binärwert auf C312

Istwert wandeln nach BCD: (x50, :128) und Display

Vergleich Istwert mit Sollwert

; Istwert > Sollwert

; Istwert < Sollwert

Istwert = Sollwert: keine Aktion

Istwert > Sollwert: Ausgang A15

Istwert < Sollwert: Ausgang A0

---

*) Für Modul PCA1.W3..

```
406   15   SCR    310
407   29    29    41

411   23   JMS    500   =>
412   15   SCR    312
413   31    31    300
414   31   DTC    312
```

BCD-Wert umrechnen in Binärwert
BCD * 41 (10V ≙ 100 ≙ 4096 ≙ 4100)

Istwert als Binärwert auf C300

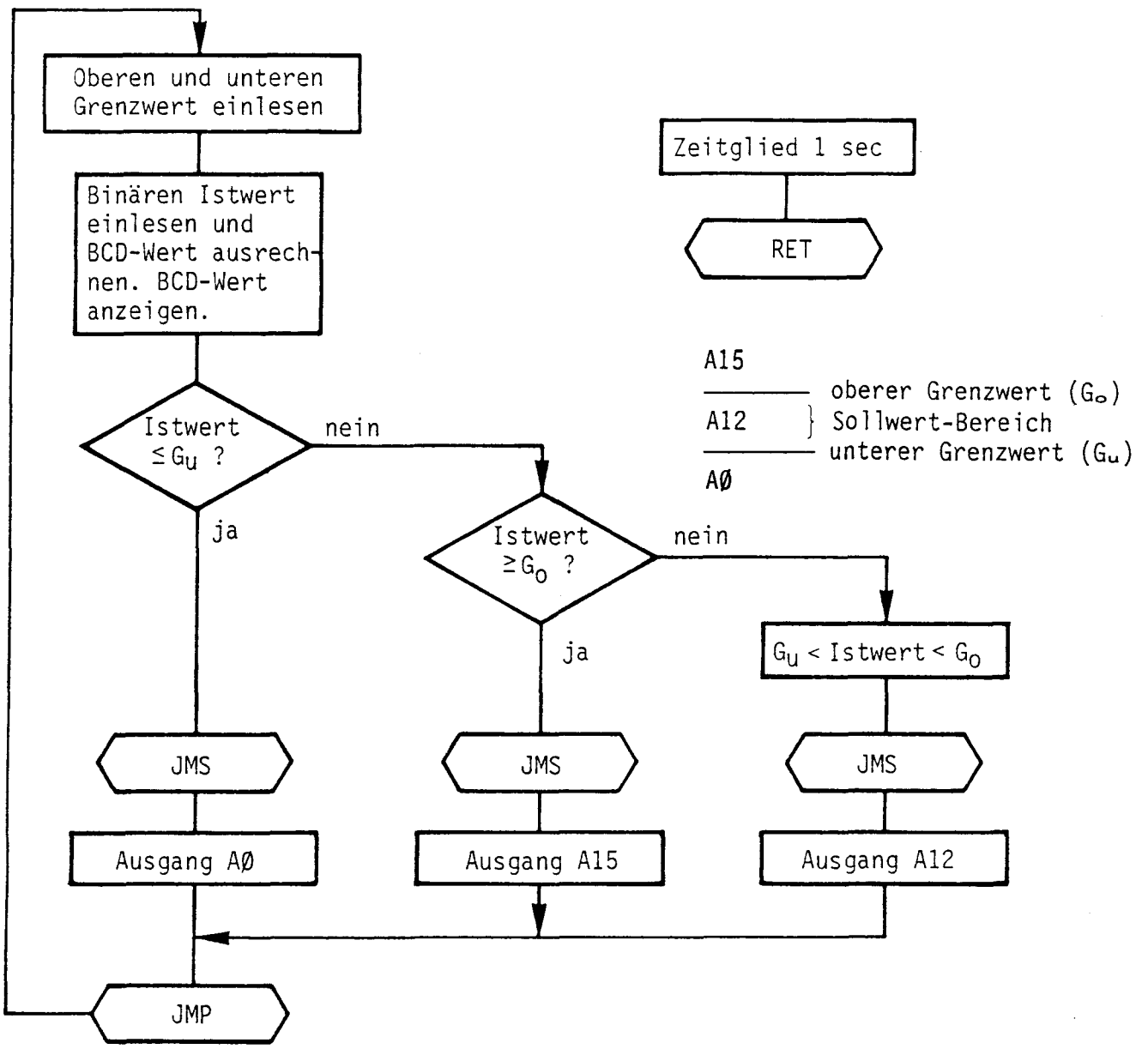
C300 auf C312 kopieren
C312 anzeigen, d.h. den Istwert

Zusatzaufgabe 23b: Dreipunktregler mit Zeithysterese

Ueber die 2-stelligen BCD-Schalter von A24 und A27 sollen der untere und der obere Grenzwert eines Dreipunktreglers eingegeben werden. Der Istwert wird über den Kanal 17 des Analogmoduls eingelesen. Erreicht dieser den unteren Grenzwert, so wird der Ausgang AØ, beim Erreichen des oberen Grenzwertes der Ausgang A15 aktiviert, sobald die als Hysterese programmierte Zeit von 1s abgelaufen ist. Befindet sich der Istwert innerhalb der beiden Grenzwerte, so ist Ausgang A12 aktiv. Der Istwert soll im Operand-Display als BCD-Wert angezeigt werden.

Flussdiagramm

Subroutine für Zeithysterese:



Lösung 23b

Für PCA1.W1.. (8 Bit)

```
300   11   SEO    24
301   15   SCR   310   }  Unterer Grenzwert als
302   16    16    31   }  BCD-Wert auf C310
303   12   REO    24

305   11   SEO    27
306   15   SCR   311   }  Oberer Grenzwert als
307   16    16    31   }  BCD-Wert auf C311
308   12   REO    27

310   10   OUT    17
311   15   SCR   312   }  Istwert als Binär-
312   24    24    23   }  Wert auf C312

**
314   15   SCR   312
315   29    29    50
316   15   SCR   312   }  Istwert wandeln
317   30    30   128   }  (x50; :128) und Display
318   31   DTC   312

321   15   SCR   310
322   28    28   312   }  Istwert ≤ unterer Grenzwert?
323   22   JIZ   331
324   23   JMS   350  ==========>  Zeithysterese
325   12   REO    15
326   12   REO    12
327  111   SEO     0   ;  Ausgang A0
328   20   JMP   300

331   15   SCR   312
332   28    28   311   }  Istwert ≥ oberer Grenzwert?
333   22   JIZ   340
334   23   JMS   350  ==========>  Zeithysterese
335   12   REO     0
336   12   REO    12
337   11   SEO    15
338   20   JMP   300   ;  Ausgang A15

340   23   JMS   350  ==========>  Zeithysterese
341   12   REO     0
342   12   REO    15
343   11   SEO    12   ;  Ausgang A12 (unterer Grenzwert < Istwert < oberer Grenzwert)
344   20   JMP   300
```

Subroutine "Zeithysterese":

```
350   14   STR   284
351   00    00    10
352   25   WIH   284 *
353   24   RET     0
```

*) Ist die Zeithysterese grösser als eine Sekunde, wird wegen des Wartebefehls der Istwert nicht ständig angezeigt.

**) Für das Modul PCA1.W3.. ist die gleiche Routine wie im vorangehenden Beispiel zu verwenden. Die Adressen 406...414 müssen an die Adressen 310...318 angepasst werden

Beispiel 24: Ein praktisches Beispiel anhand einer automatischen Bohrvorvorrichtung

Aufgabenstellung

Zu einer automatischen Bohrvorrichtung ist die Mechanik sowie das Schrittdiagramm vorgegeben. Ein Teil des Bedienfeldes ist ebenfalls vorgegeben. Der Ausbau dieses Komforts hängt jedoch von den Möglichkeiten der gewählten PLC ab.

Allgemeine Funktionsbeschreibung (siehe Zeichnung auf der folgenden Seite)

Runde Scheiben sollen automatisch zugeführt, mit einer exzentrischen Bohrung versehen und ausgestossen werden.

SCHRITT 1: Scheiben aus Magazin mittels Kolben A in Bohrposition schieben und festklemmen.
SCHRITT 2: Bohrmotor ein und Bohrer senken.
SCHRITT 3: Bohrer nach ca. 4s heben, um Späne aus Bohrloch zu entfernen.
SCHRITT 4: Erneut Bohrer senken bis Lochtiefe erreicht ist.
SCHRITT 5: Bohrer heben.
SCHRITT 6: Bohrmotor aus dem Einschiebkolben A zurück.
SCHRITT 7: Ausstosskolben C vor.
SCHRITT 8: Ausstosskolben C zurück.
Wiederbeginn bei Schritt 1.

Geber

Alle Endpositionen der Kolben werden durch Geber rückgemeldet. Für solche Abschaltfunktionen würde man aus Gründen der Drahtbruchsicherheit Oeffner einsetzen. Um die Funktionen aber etwas einfacher simulieren zu können, haben wir Schliesser gewählt.

Der minimale Magazinzustand wird ebenfalls mit einem Geber überwacht. Vor jedem Zyklus soll durch einen Geber der Bohrer auf Bruch überprüft und gegebenenfalls die Maschine stillgesetzt werden.

Bedienfeld

Zusätzlich zu den Funktionen "Start" und "Stop Zyklus" soll das Ablaufprogramm auch an den Programmanfang zurückgestellt werden können ("Reset Programm"). Um unbeabsichtigtes Rückstellen zu vermeiden, ist dazu eine Doppelbetätigung der Tasten "Stop" und "Reset" erforderlich.

Der Notstop erfolgt hardwareseitig über eine Pilz-Taste direkt auf das Hauptschütz (Vorschrift).

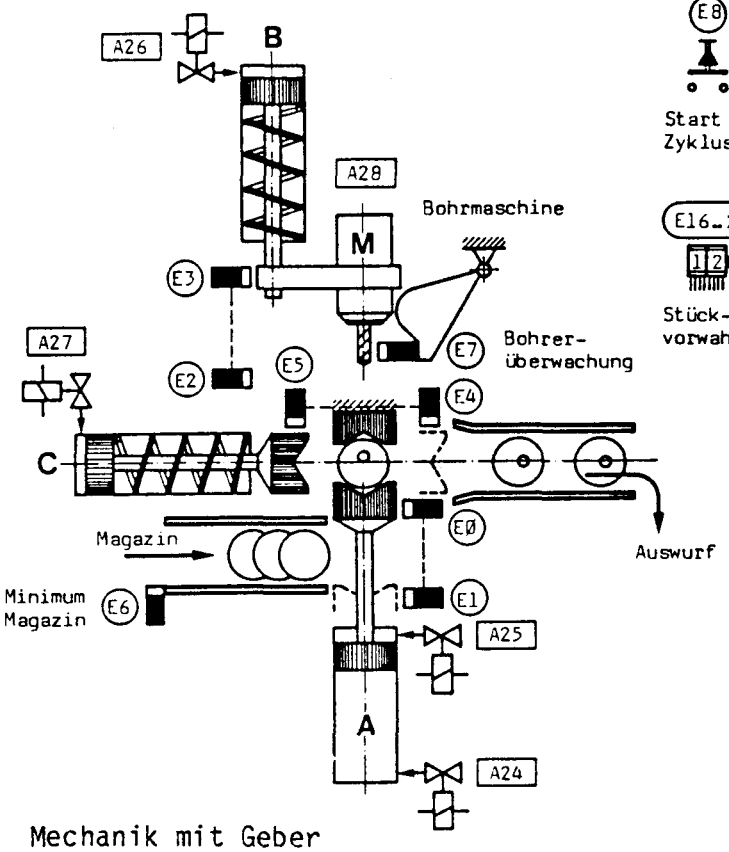
Die gewünschte Stückzahl (1ØØ...1Ø'ØØØ) pro Auftrag soll mit BCD-Schaltern vorgewählt werden können. Die Eingabe bzw. Wiedereingabe der vorgewählten Stückzahl soll über einen Impulstaster erfolgen.

Beim Einrichten und im Servicefall soll der ganze Ablauf auch einzelschrittweise durchlaufen werden können.

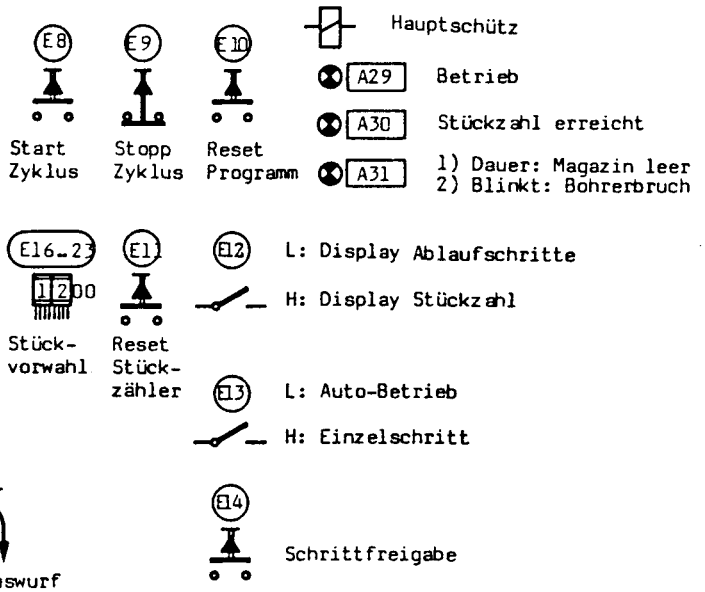
Es ist wünschenswert, nicht nur die verbleibende Stückzahl anzeigen zu können, sondern im Fehlerfall auch festzustellen, in welchem Schritt die Maschine stillsteht bzw. auf welche Funktion gewartet wird.

Gemäss Abbildung sollen die verschiedenen Funktionen mit Signallampen angezeigt werden.

Zeichnung im Ablaufschritt 2



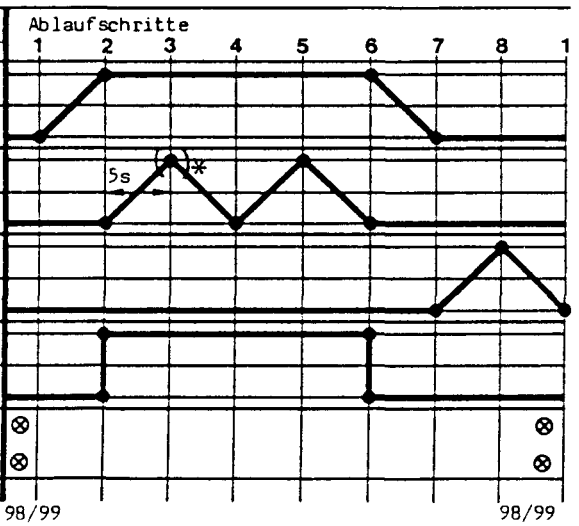
Mechanik mit Geber



Bedienfeld

Schrittdiagramm

Aktion	Zylinder		Ausgang	Signal
Stück einschieben	A	vor	A24	EØ
		rück	A25	E1
Bohrmasch. senken	B	unten	A26	E2
		oben	-	E3
Stück ausstossen	C	vor	A27	E4
		rück	-	E5
Motor Bohrmaschine	M	ein	A28	-
		aus	-	-
Bohrerkontrolle				E7
Magazinkontrolle				E6



*Alternativ 5 sec bohren oder Erreichen der Endstellung E2.

24.1 Dimensionierung der PLC

24.1.1 Funktionen

Zur Erfüllung der gestellten Funktionen werden ausser logischen Verknüpfungen auch Ablaufsteuerungen, Zeitfunktionen und Zählfunktionen gewünscht. Zusätzlich sind Display-Möglichkeiten von Zählern erforderlich. Diese Aufgabenbreite kann problemlos mit allen SAIA°PLCs erfüllt werden.

24.1.2 Anzahl E/A

Die kleine Anzahl E/A erlaubt den Einsatz einer PCA1. Die E/A werden auf dem hierfür vorgesehenen Blatt aufgelistet, wobei darauf zu achten ist, dass die E/A-Aufteilung im Raster von 8 oder allenfalls 4 E/A erfolgen kann.

Im vorliegenden Fall fehlt der Ausgang zur Anzeige der Magazin- bzw. Bohrerüberwachung. Lösung: Da diese Anzeigen relativ selten vorkommen werden, brauchen wir den gleichen Ausgang und unterscheiden durch Dauerlicht bzw. Blinklicht. Auf diese Weise finden wir Platz auf einer PCA151.

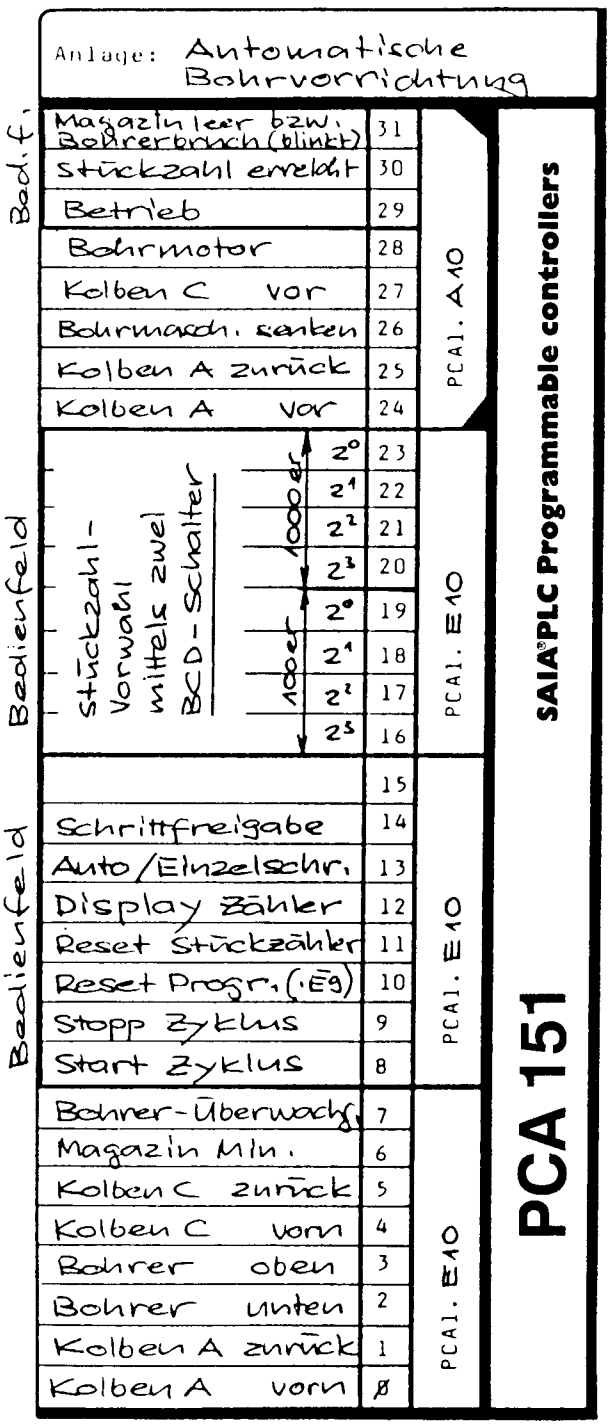
24.1.3 Art der E/A

Wir wählen nicht galvanisch getrennte 24V= E/A-Module mit Transistorausgängen und wählen die Magnetventile und Lampen dementsprechend.

24.1.4 Speicherkapazität

Bei der geringen Komplexität dieses Programmes dürfen wir mit einer Verknüpfungstiefe von ca. 5 rechnen. Wenn wir die 8 Eingänge des BCD-Schalters nur wie 1 E rechnen, ergibt sich:
24 E+A x Verknüpfungstiefe 5 = 12Ø Speicherplätze.
Diese sind problemlos auf 1K-Speicher unterzubringen.

Für den Bohrmotor 22ØV≈ benützen wir ein externes Interface mit Relais. Auf der Eingangsseite ergeben die 24V= Vorteile bei der Verwendung von Näherungsschaltern für die Geber.



E/A- Belegungsblatt

24.1.5 Anzeige

Mit dem Modul PCA1.D11 können die erforderlichen Anzeigen ohne Verlust von E/A realisiert werden.

24.2 Programmerstellung

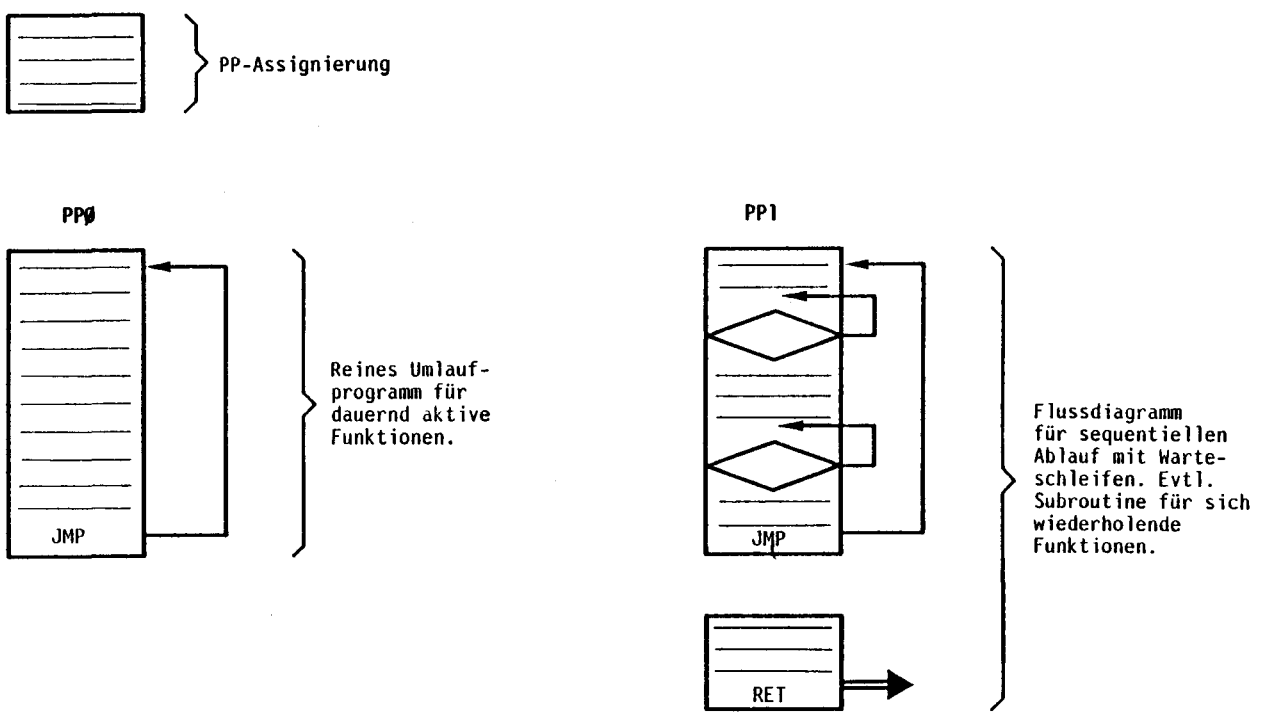
Geübten PLC-Anwendern wird die Prosabeschreibung zusammen mit den Figuren bereits genügen, um direkt zur Erstellung des Programmes bzw. des Flussdiagrammes zu schreiten.

Einsteigern seien folgende Schritte empfohlen:

24.2.1 Programmstruktur

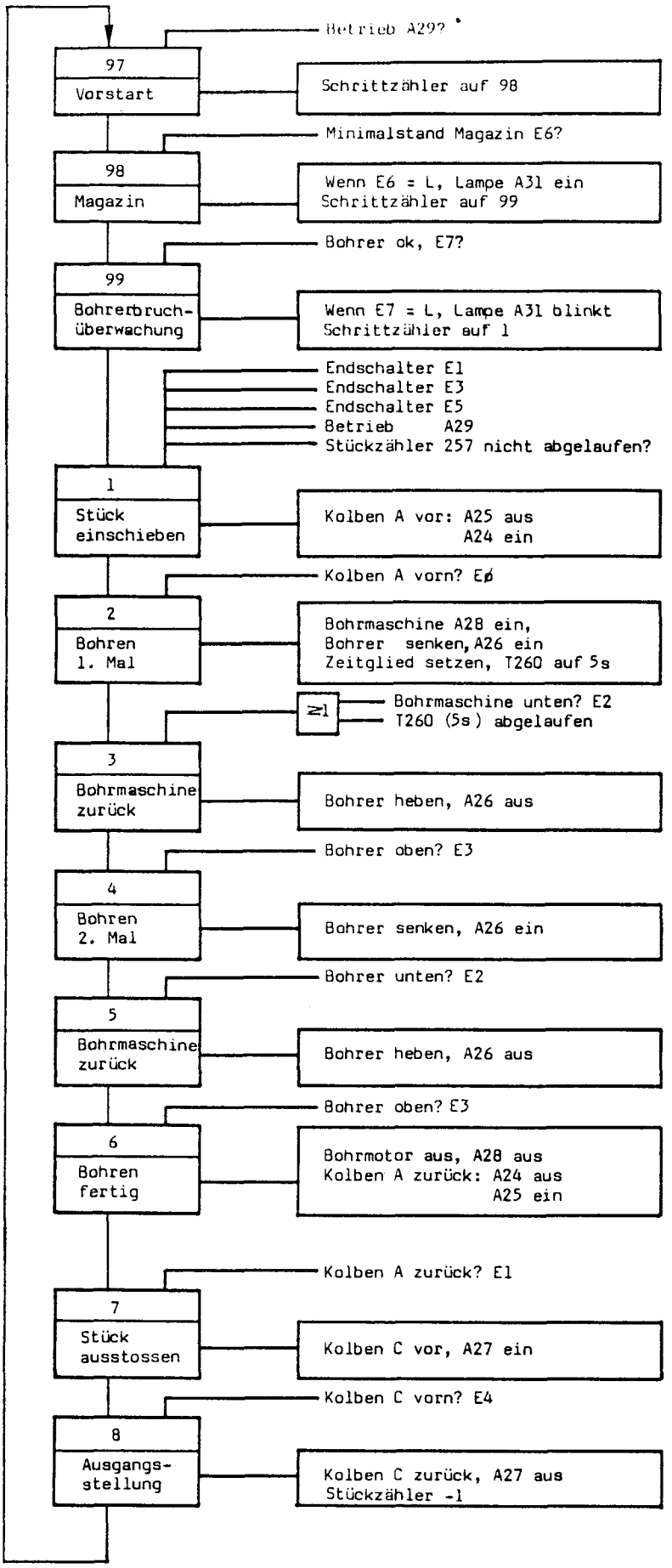
Es handelt sich um einen sequentiellen Ablauf, der sich sehr gut zur Programmierung nach Flussdiagramm eignet. Für dauernd aktive Funktionen, wie Start/Stop, Displays etc. wählen wir ein Umlaufprogramm als Parallelprogramm.

Daraus ergibt sich folgende Struktur:



24.2.2 Schrittablaufplan nach DIN

Falls schon in dieser Darstellungsart gearbeitet wurde, kann die Prosabeschreibung und das Schrittdiagramm zuerst in dieser Form dargestellt werden. Dabei wird man bemerken, dass vorgängig zum Ablaufschritt 1 noch einige Vorschritte zur jeweiligen Ueberprüfung verschiedener Funktionen nötig sind. In diese Vorschritte werden nur Funktionen aufgenommen, welche vor jedem Zyklusdurchlauf durchgeführt werden müssen. Dauernd aktive Funktionen gehören ins Umlaufprogramm PPØ.


Betrieb A29?
97
Vorstart
Schrittzähler auf 98
Minimalstand Magazin E6?
98
Magazin
Wenn E6 = L, Lampe A31 ein
Schrittzähler auf 99
Bohrer ok, E7?
99
Bohrerbruch-
überwachung
Wenn E7 = L, Lampe A31 blinkt
Schrittzähler auf 1
Endschalter E1
Endschalter E3
Endschalter E5
Betrieb A29
Stückzähler 257 nicht abgelaufen?
1
Stück
einschieben
Kolben A vor: A25 aus
A24 ein
Kolben A vorn? Eø
2
Bohren
1. Mal
Bohrmaschine A28 ein,
Bohrer senken, A26 ein
Zeitglied setzen, T260 auf 5s
≥1
Bohrmaschine unten? E2
T260 (5s) abgelaufen
3
Bohrmaschine
zurück
Bohrer heben, A26 aus
Bohrer oben? E3
4
Bohren
2. Mal
Bohrer senken, A26 ein
Bohrer unten? E2
5
Bohrmaschine
zurück
Bohrer heben, A26 aus
Bohrer oben? E3
6
Bohren
fertig
Bohrmotor aus, A28 aus
Kolben A zurück: A24 aus
A25 ein
Kolben A zurück? E1
7
Stück
ausstossen
Kolben C vor, A27 ein
Kolben C vorn? E4
8
Ausgangs-
stellung
Kolben C zurück, A27 aus
Stückzähler -1

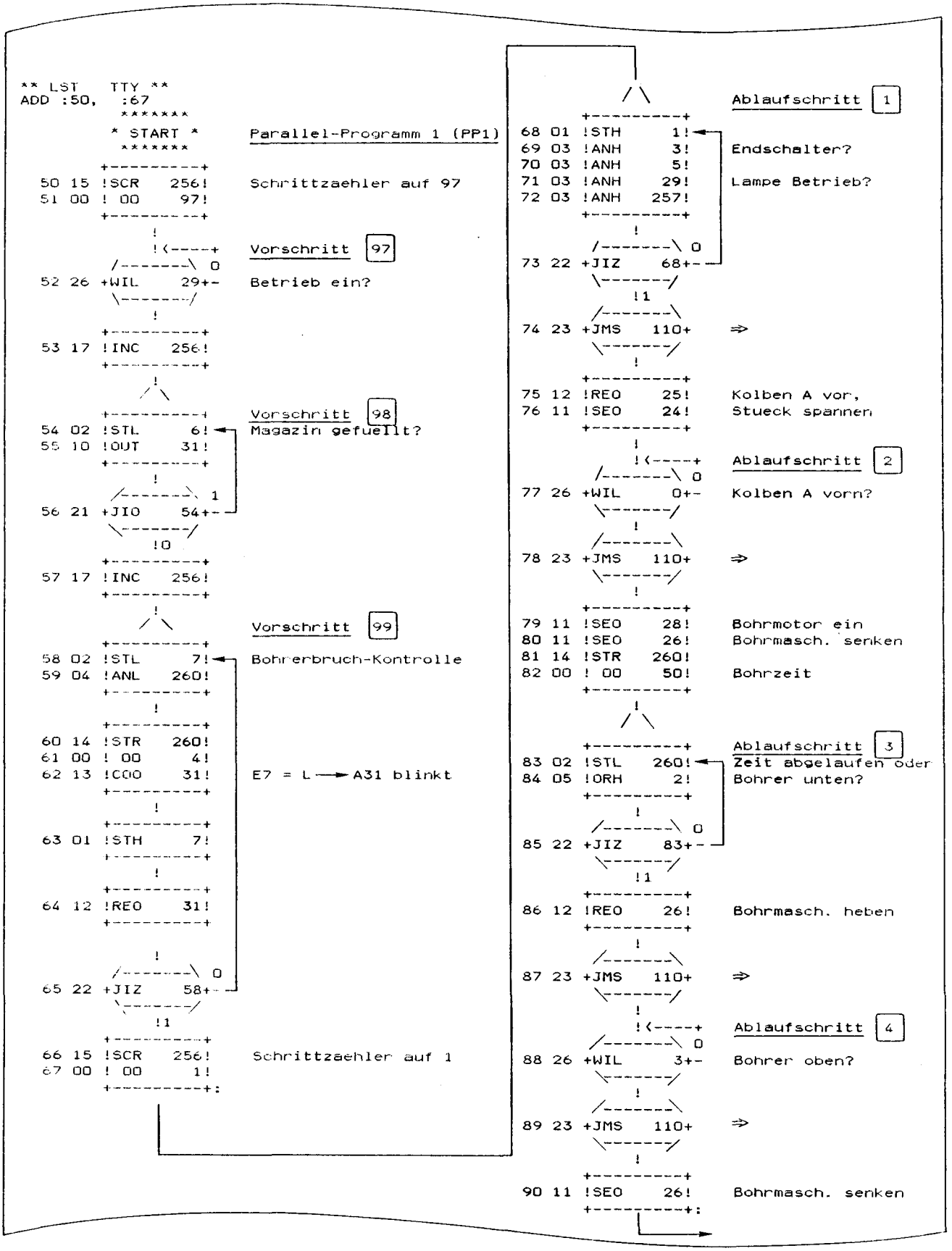
24.2.3 Programmerstellung

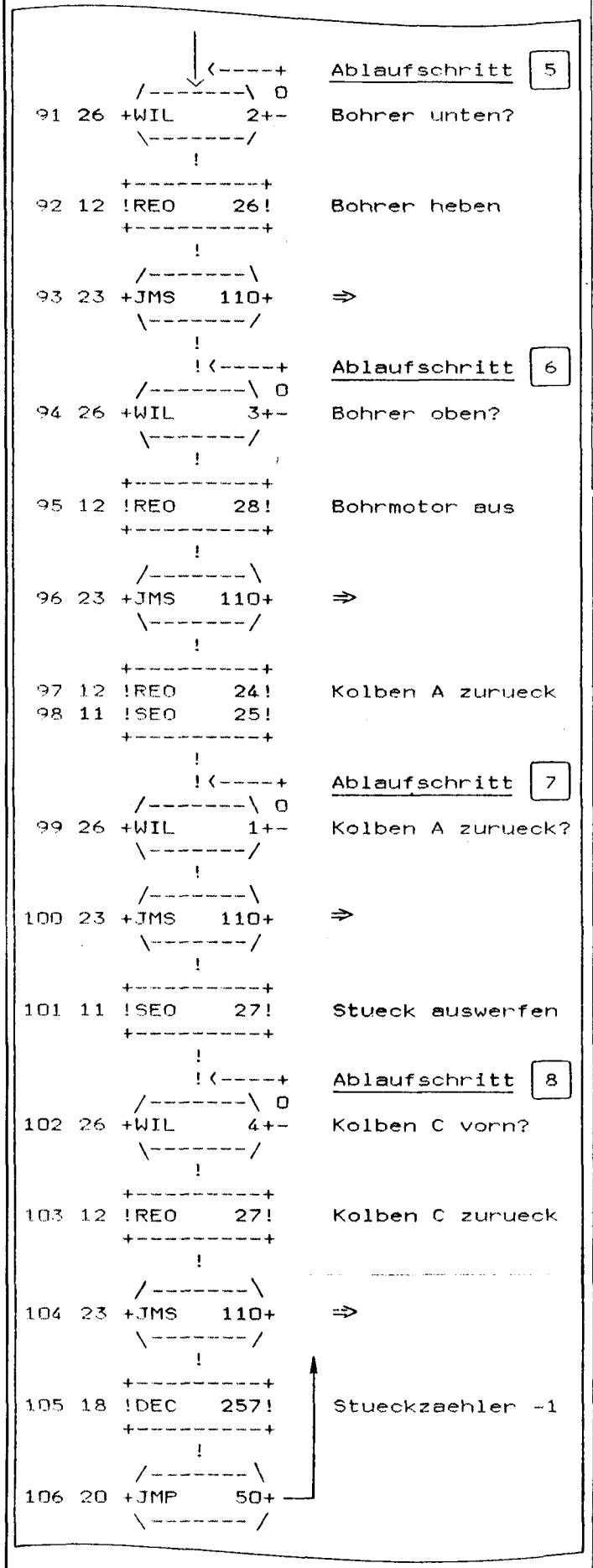
Das Ablaufprogramm kann aufgrund des DIN-Planes nun sehr leicht erstellt werden. Da die Funktion des Einzelschrittbetriebes und des Programmschrittzählers in jedem Ablaufschritt gleichermassen vorkommen, werden sie in eine Subroutine genommen. Dabei ist darauf zu achten, dass zuerst gefährliche Funktionen rückzusetzen sind, bevor auf den Schritttaster gewartet wird, d.h. generell zuerst alle REO, dann JMS, dann SEO.

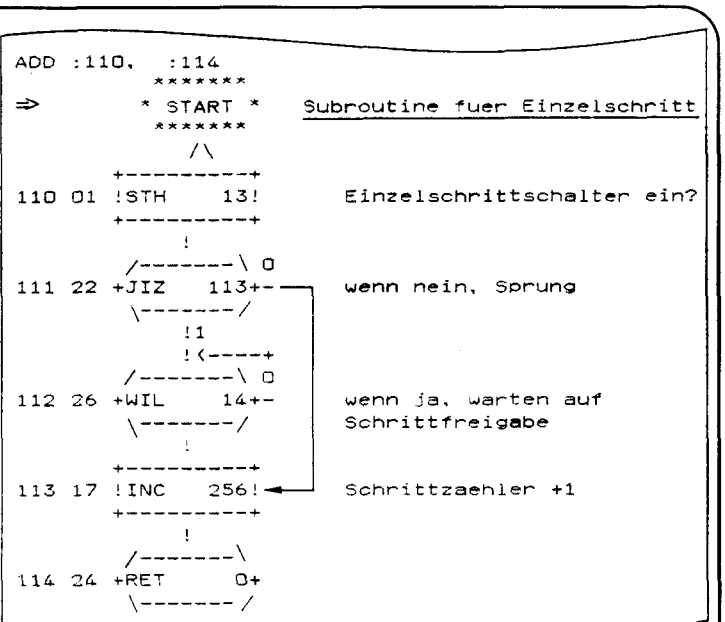
Im Umlaufprogramm finden wir, wie erwähnt, die ständig aktiven Funktionen. Interessant ist dabei der letzte Abschnitt mit der Programm-Rückstellung. Um das Ablaufprogramm an seinen Anfang zurückzubringen, werden indexiert alle wichtigen Ausgänge zurückgesetzt und anschliessend das PP1 wieder neu auf seiner Anfangsadresse assigniert.

```
KOMMENTIERTER PROGRAMMAUSDRUCK DANK DER PCA ASSEMBLER

********************  PP - ASSIGNIERUNG
ADDR NC   MNC   OPRD
   0 00   NOP      0
   1 29   PAS      1
   2 00    00     50
   3 20   JMP     10->
 ++++++++++++++++++ PARALLEL-PROGRAMM 0 (PPO)
->10 01   STH      8  } START/STOP ZYKLUS
  11 03   ANH      9  }
  12 11   SEO     29  LAMPE "BETRIEB" EIN
  13 02   STL      9
  14 12   REO     29  LAMPE "BETRIEB" AUS
 ------------------ STUECKZAEHLER SETZEN
  17 01   STH     11
  18 09   DYN    300
  19 15   SCR    257
  20 18    18     23
  21 02   STL    257
  22 10   OUT     30  LAMPE "STUECKZAHL ERREICHT"
 ------------------ ANZEIGE-VORWAHL
  25 01   STH     12
  26 31   DTC    257  ANZEIGE STUECKZAEHLER
  27 08   NEG      0
  28 31   DTC    256  ANZEIGE SCHRITTZAEHLER
 ------------------ PROGRAMM-RUECKSTELLUNG AUF ANFANG
  31 01   STH     10  }                  (RE-ASSIGNIERUNG)
  32 04   ANL      9  } RUECKSTELLUNG ?
  33 09   DYN    301  }
-> 34 22   JIZ     10->
  35 16   SEI      0
->36 12   REO   1024  RUECKSTELLEN DER AUSGAENGE
  37 27   INI      7
  38 21   JIO     36->
  39 29   PAS      1  } RE-ASSIGNIERUNG VON PP1
  40 00    00     50  } UND DAMIT RUECKSTELLUNG
  41 20   JMP     10->
```

Darstellung des PP1 im Flussdiagramm mittels PCA-Assembler







QUERVERWEIS EINIGER ELEMENTE UND DER ABSPRUNG-

ADRESSEN ZUR SUBROUTINE 110.

SCHRITTZAEHLER

```
ADD :0,   :114, OPERAND :256

  28 31   DTC    256
  50 15   SCR    256
  53 17   INC    256
  57 17   INC    256
  66 15   SCR    256
 113 17   INC    256
```

STUECKZAEHLER

```
ADD :0,   :114, OPERAND :257

  19 15   SCR    257
  21 02   STL    257
  26 31   DTC    257
  72 03   ANH    257
 105 18   DEC    257
```

SUBROUTINE 110

```
ADD :0,   :114, OPERAND :110

  74 23   JMS    110
  78 23   JMS    110
  87 23   JMS    110
  89 23   JMS    110
  93 23   JMS    110
  96 23   JMS    110
 100 23   JMS    110
 104 23   JMS    110
```

LAMPE BETRIEB

```
ADD :0,   :114, OPERAND :29

  12 11   SEO     29
  14 12   REO     29
  52 26   WIL     29
  71 03   ANH     29
```

OBERER ENDSCHALTER BOHRMASCHINE

```
ADD :0,   :114, OPERAND :3

  69 03   ANH      3
  88 26   WIL      3
  94 26   WIL      3
```

Vorgehen zur Lösung eines Steuerungsproblems durch Einsatz einer PLC

START

Erstellen eines Pflichtenheftes für den ganzen Prozess (evtl. mit Skizzen, Ablaufbeschreibung, Aufgliederung in Prozessblöcke usw.).

Grobkonstruktion der Anlage, aus welcher ersichtlich ist, was mechanisch/pneumatisch/hydraulisch oder elektrisch gelöst wird.

PLC-Dimensionierung

- Bestimmen der Anzahl E und A (unter Beachtung der MUX-Möglichkeiten)
- Festlegen der E/A-Arten (=/≈, Spannung, Strom)
- Bestimmen der PLC-Baugrösse (PCA1 ≤112 E+A, PCA2 >96 E+A), evtl. Aufteilung des Prozesses auf mehrere PLCs, Hierarchie.
- Festlegen der PLC-Module (Raster 8/16/32) unter Berücksichtigung einer Reserve von 1Ø - 2Ø% E+A.
- Bestimmen der Speichergrösse nach folgenden Grobrichtlinien
  - einfache Steuerung Speicherkapazität = ca. 5 x Anzahl E/A
  - mittlere Komplexität Speicherkapazität = ca. 1Ø x Anzahl E/A
  - hohe Komplexität Speicherkapazität = ca. 2Ø x Anzahl E/A
- Prüfen, ob Programm mit Standardfunktionen der SAIA°PLC ausführbar ist. Falls universelle arithmetische Funktionen oder umfangreiche Protokollmöglichkeiten gewünscht werden, sind Typen PCA14 oder PCA23 vorzusehen.
- Festlegen der definitiven Speicherart (Standard = EPROM).

PLC-Bestellung

Genaues Festlegen der PLC-Bestückung inkl. Zubehör wie Kabel, externe Interfaces, Displaymodul, Programmier- und Simulierzubehör.

ARBEITSTEILUNG

A R B E I T S T E I L U N G

Hardware

- Fertigstellen der Detail-konstruktion
- Bau der mechanischen Teile

Wenn bestellte PLC eingetroffen ist:
- Montage der PLC in der Anlage
- Verdrahtung der E/A
- Durchprüfen der E/A-Verdrahtung mittels LED und Betriebsart "MAN"

Erstellen des Programmes

- Bezeichnung der E/A
- Festlegen der Programmstruktur und der Programmierungsart der entsprechenden Teilprogramme (Programm-Module!)
- Erstellen des Programmes auf Papier (unter Berücksichtigung der Fehlerdiagnose mittels DOP, DTC und Watchdog).

Wenn Programmierplatz bereits vorhanden oder bestellte PLC eingetroffen:
- Eingeben des Programmes (NOP für Reserve und Aenderungen eingeben)
- Testen der Programmteile mit Simulierhilfen
- Entsprechende Fehler korrigieren
- Provisorische Dokumentation erstellen
- Sichern des Programmes auf EPROM

Inbetriebnahme

- Wenn simulierter Programmtest und E/A-Durchprüfung auf "MAN" erfolgreich waren ---> einschalten (evtl. gefährliche Stellglieder abtrennen, Start-tasten nicht sofort betätigen).
- Falls Einzelschrittmöglichkeit vorgesehen (nicht zu verwechseln mit Betriebsart STEP), dann zuerst in dieser Position den Prozess schrittweise durchprüfen.
- Automatik-Betrieb durchtesten (evtl. auch gefährliche Betriebszustände, wie Drahtbruch, Betätigung mehrerer Tasten gleichzeitig oder Spannungs-Ausfall bzw. Prozessor-Ausfall durchprüfen).
- Evtl. Korrektur des Programmes auf RAM.
- Wenn o.k., Programm in 2 Sätzen auf EPROM kopieren
  1 Satz für Betrieb, 1 Satz zur Programmsicherung.
- Programmdokumentation auf den letzten Stand bringen und so kommentieren, dass Drittpersonen das Programm ebenfalls verstehen können.
- Erstellen einer Anleitung zur Fehlerdiagnose für das Unterhaltspersonal.

E N D E

Alphabetische Befehlsübersicht Stufe 1H