saia® pc

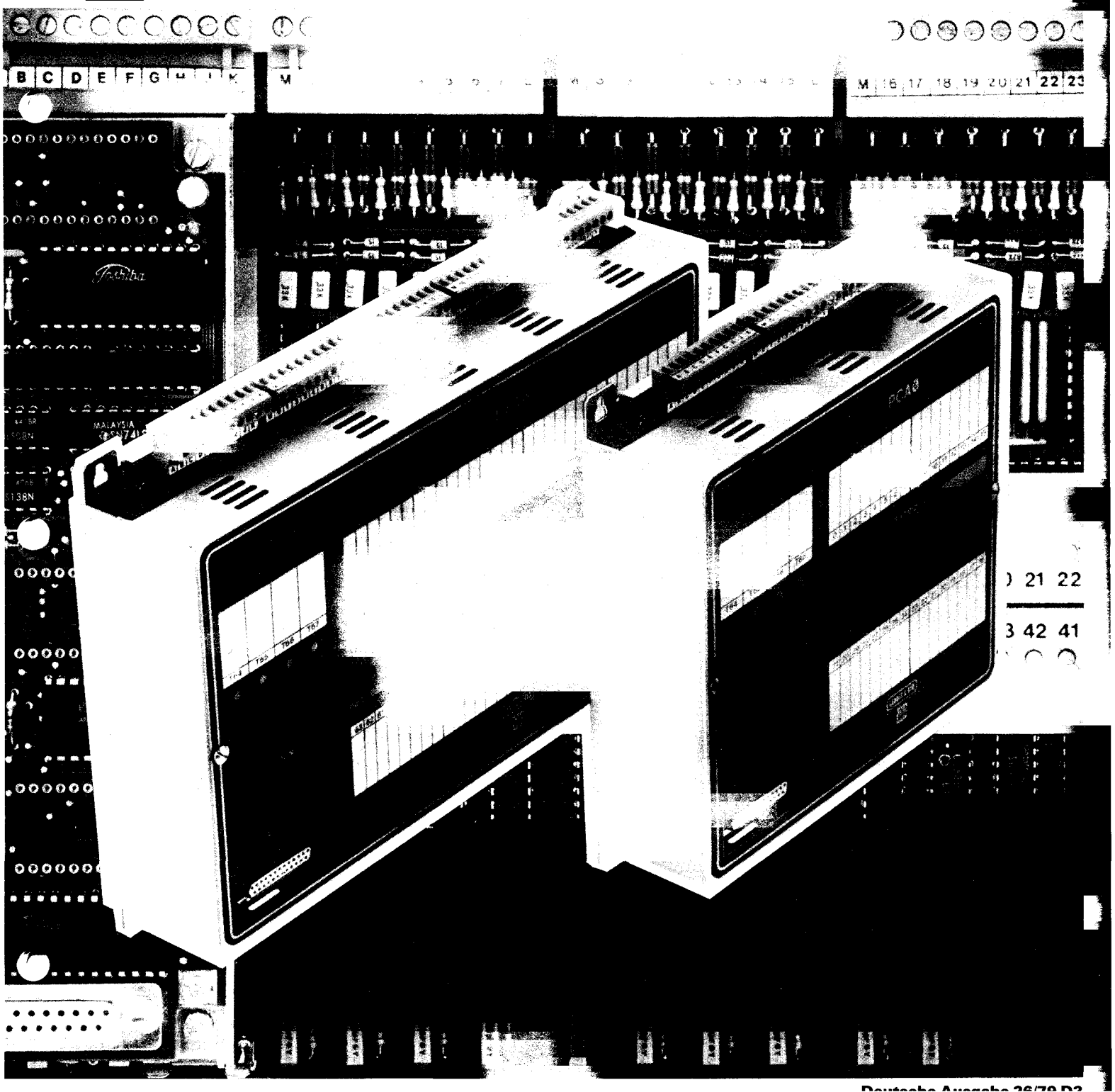
Speicherprogrammierbare Steuerungen

LANDIS & GYR

SAIA

Handbuch PCA0-Standard

Kompakt-SPS mit hoher Intelligenz



Deutsche Ausgabe 26/79 D2

saia® pc

Baureihe PCAØ

Die kompakte Speicherprogrammierbare Steuerung von hoher Intelligenz

- Vier Standardausführungen bis 64 E+A
- 4K Programmzeilen
- Programmierbar nach Kontaktplan, Logikplan, Flussdiagramm oder Funktionsplan nach DIN
- Arithmetik-Befehle sowie Befehle zur permanenten Funktionsüberwachung (Watch Dog und Check Sum)
- Leichter Einstieg für den Anfänger mit viel Leistungsreserve für den anspruchsvolleren SPS-Anwender

01.03.1987

Verkaufspreis: sFr. 20.--

SODECO-SAIA AG CH-3280 MURTEN SCHWEIZ

LANDIS & GYR

SAIA

Inhaltsverzeichnis

Kapitel 1 Die PCA0 von aussen und von ganz innen

Kapitel 2 Gemeinsame technische Daten

Kapitel 3 Die 4 Standard-Ausführungen

Kapitel 4 Wichtiges Zubehör

Kapitel 5 Schnellanleitung zur Handhabung der PCA0

Kapitel 6 Details über Stromversorgung und Watch Dog

Kapitel 7 Details über Eingänge und Ausgänge

Kapitel 8 Die Betriebsarten der PCA0

Kapitel 9 Programmieren in drei leichten Schritten

Kapitel A Programmieren nach Kontaktplan

Kapitel B Programmieren nach Flussdiagramm

Kapitel C Indexierung, Arithmetik und Check-Sum

Anhang: Programmierblätter zum Kopieren

Für wen ist das PCA0-Handbuch ?

Wir kennen Ihre Vorkenntnisse im Arbeiten mit speicherprogrammierbaren Steuerungen nicht. Vielleicht sind Sie ein Einsteiger oder schon ein routinierter SPS-Fachmann.

Dieses Handbuch ist als Lehrgang gestaltet, um dem Einsteiger einen problemlosen Zugang zur Welt der speicherprogrammierbaren Steuerungen zu ermöglichen. Lesen Sie vor allem die Kapitel 1 bis 5 sorgfältig durch, bevor Sie mit Hantieren beginnen, und lassen Sie sich von den Kapiteln 6 und 7 mit den vielen Schematas nicht verwirren. Diese sind vorderhand nicht wichtig, wenn Sie sich mit dem Simulier- und Speisegerät PCA2.SØ5 (gemäss Kapitel 5e) ausgerüstet haben.

Ab Kapitel 9 werden Sie dann Schritt um Schritt die Treppe A-B-C hinaufgeführt. Wir verwenden dabei einfache, transparente Beispiele, welche allesamt mit dem erwähnten Simuliergerät auf Ihrem Schreibtisch zum "Laufen" gebracht werden können.

Sollten Sie irgendwo hängenbleiben, dann profitieren Sie bitte vom Können unserer Fachleute in Ihrer Nähe oder schreiben Sie sich für den nächsten Workshop ein.

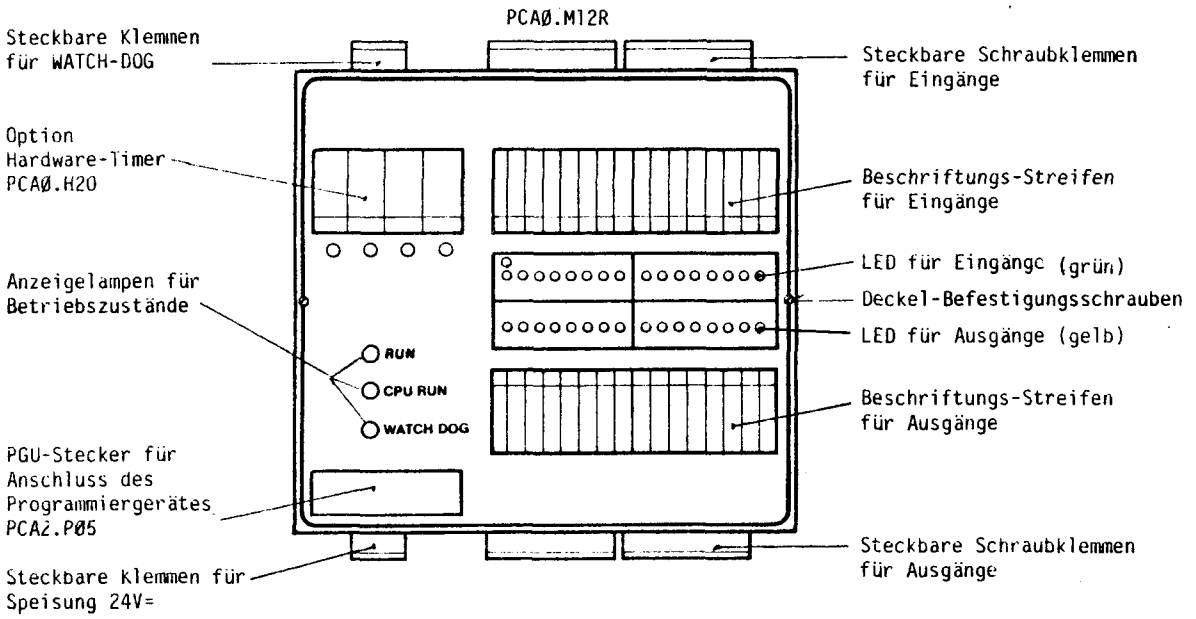
Wir wünschen Ihnen viel Vergnügen beim vielfältigen "Spielen" mit der PCA0 !

Als SPS-Profi können Sie einiges im Kapitel 9 überspringen und sich auf die Befehlstabellen zu Anfang der Teile A, B und C konzentrieren. Möchten Sie mehr Details über diese Befehle wissen, dann steht Ihnen auch das umfangreiche "Basis-Handbuch" zur Verfügung.

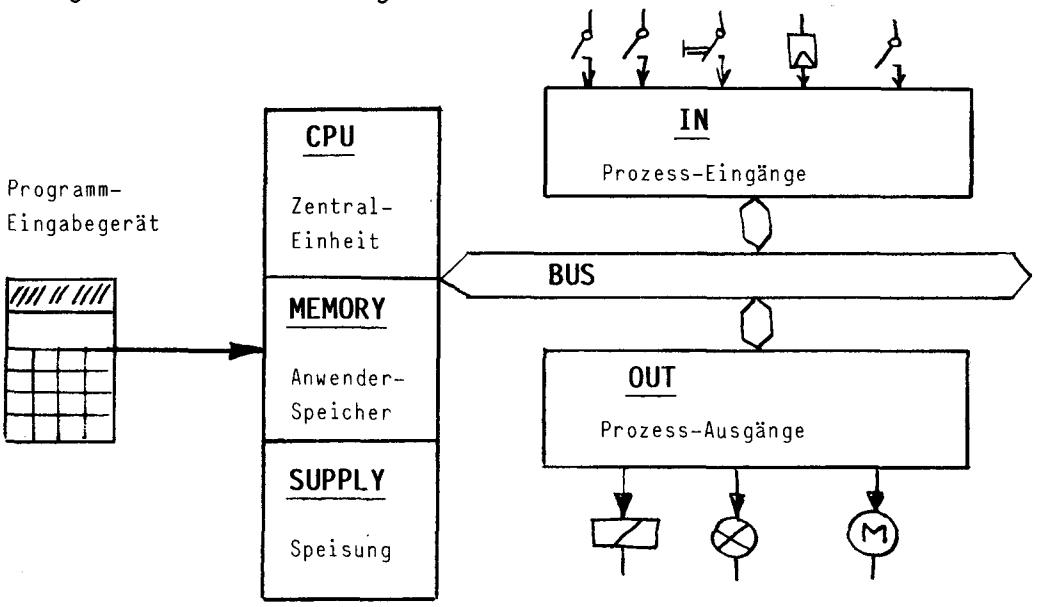
1. Die PCAØ von aussen und von ganz innen

Die PCAØ ist die kompakte Baureihe in der SAIA-PC-Systemfamilie. Die Ein- und Ausgangsebene ist nicht modular auf einem einzigen Print untergebracht. Dank der hohen Intelligenz können ganz einfache, aber auch recht komplexe Aufgaben mit der PCAØ gelöst werden.

Von aussen können folgende Funktionsteile unterschieden werden:



Von ganz innen sind folgende Funktionseinheiten zu erkennen:



Mit dem Programm-Eingabegerät wird das Anwenderprogramm in den Anwenderspeicher (RAM) eingegeben. Die CPU interpretiert dieses Programm, fragt die Zustände der Prozess-Eingänge ab und steuert dementsprechend die Prozess-Ausgänge.

saia®pc

2. Gemeinsame technische Daten

Mikroprozessor-System	8085-2
Zykluszeit pro Anwender-Befehl (Mittel)	70 µs
Befehlssatz-Niveau (1H)	32 Grundbefehle + 20 Zusatzbefehle inkl. Arithmetik
Anzahl Parallelprogramme	16
Anzahl Indexregister	16 (je 1 pro Parallelprogramm)
Anzahl Unterprogramm-Ebenen	3
Anwenderspeicher	4K Programmzeilen (≙ 8K Bytes)
Merker/Haftspeicher	477 * + 235 = 712
Anzahl Software-Zähler + Zeitglieder	64 Adressen (Z = 64, T = 32)
Zählkapazität	65'535 *
Zeitbereiche (Zeitbasis 0,1 bzw. 0,01 s)	0,1 (0,01) bis 6500 (650) sec.
Hardwaretimer PCAØ.H20	4 Zeitbereiche 0,9/3,7/30/240 s
Anschluss der Programmiergeräte	via 25 pol. PGU-Stecker
Betriebsarten	RUN, BREAK, STEP, MAN, PROG
Anzeige-Lampen	LED für RUN/CPU RUN/WATCH DOG LED für E/A
Eingänge (B90)	galvanisch verbunden, Quell- oder Senkbetrieb nominal +24V, H = +19 ... +32V L = Ø ... + 4V I = 10 mA, 24V=, t_E = 9 ms
Relais-Ausgänge (A21)	galvanisch getrennte Arbeitskontakte, Schaltleistung 3A, 250V~ AC1 3A, 24V= DC1
Transistor-Ausgänge (B90)	galvanisch verbunden, plusschaltend 0,5 mA ... 0,5A, 5 ... 36V=
Speisespannung	24V= ± 20 %
Umgebungstemperatur	Ø ... +50°C
Hohe Störsicherheit	gemäss IEC 255-4/E5 Klasse III, d.h. 2500 V und IEC 801-4 Klasse III (2000 V)

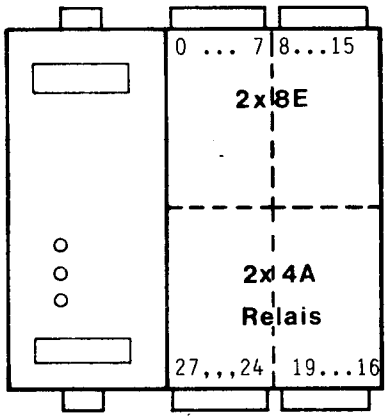
*) Durch Einhängen der Brücke "NV" werden alle Merker sowie alle Register für Timer und Zähler nullspannungssicher.

3. Die 4 Standard-Ausführungen

3.1 Mit Relais-Ausgängen

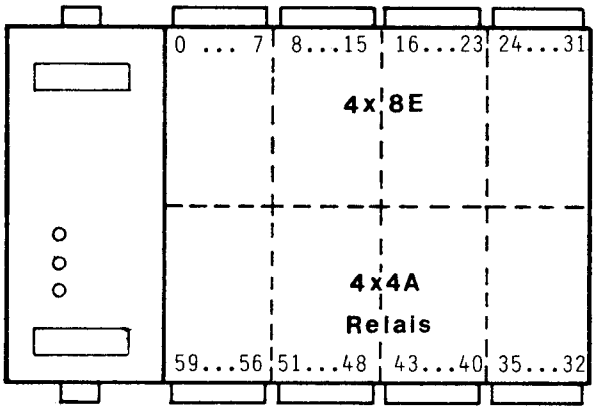
Typ PCAØ.M12R M4

16E, 24V-
8A, Relaiskontakt
max. 3A, 250V~



Typ PCAØ.M14R M4

32E, 24V-
16A, Relaiskontakt
max. 3A, 250V~

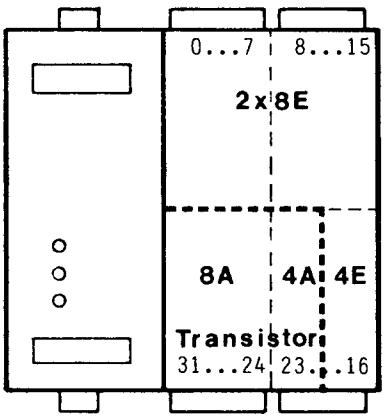


3.2 Mit Transistor-Ausgängen

M22T M4 – ausser serieller Schnittstelle

Typ PCAØ.M12T M4

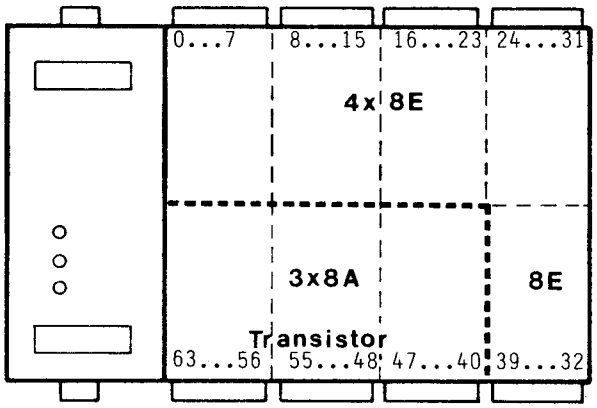
20E, 24V-
12A, 0,5A/24V-



M24T M4

Typ PCAØ.M14T M4

40E, 24V-
24A, 0,5A/24V-

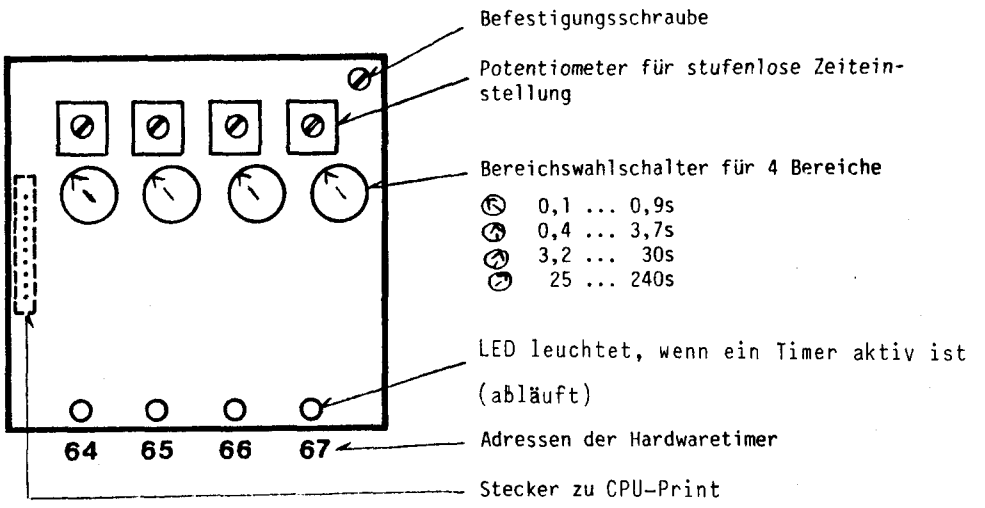


Details zu den Ein-/Ausgängen siehe Abschnitt 7

P.S.: Bei genügender Stückzahl stellen wir auch Ihnen Ihre massgeschneiderte Ausführung her. Nehmen Sie Kontakt mit unserer nächsten Vertretung auf.

4. Wichtiges Zubehör

4.1 Das Hardwaretimer-Modul PCA0.H20



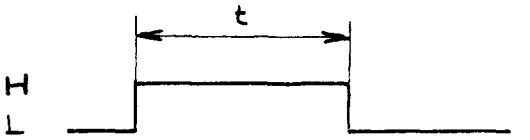
Dieses Modul ist als Option erhältlich und ermöglicht die problemlose Veränderung von 4 Zeiten, unabhängig vom Programm, im RUN-Betrieb (z.B. zum Justieren von Verzögerungszeiten).

Die Wiederholgenauigkeit beträgt:

- unter konstanten Bedingungen 0,1 % des eingestellten Zeitbereiches
- unter extremen Bedingungen 1 % des eingestellten Zeitbereiches
  (T = 0 ... 50°C, U = 24V ± 20 %)

Zum Abwarten einer eingestellten Zeit am Timer 64 genügt das folgende einfache Programm:

```
REO 64
SEO 64
WIH 64
```



Die zugehörige LED leuchtet, solange der Timer aktiviert ist (abläuft).

Bitte vergessen Sie nicht, dass dieses Modul nur erforderlich ist, wenn Sie mit den 32 Softwaretimern, die jede PCAØ in ihrer Grundausrüstung enthält, nicht auskommen. Die Softwaretimer können über 8 Eingänge via BCD-Schalter oder via Programm von 0,01s bis 6500s ebenfalls im RUN-Betrieb verändert werden (siehe Anwendungsbeispiel A8).

4.2 Die Anwenderspeicher

Drei verschiedene Arten von Anwenderspeichern mit je 4K (4096) Anwenderschritten stehen zur Verfügung:

- RAM-Chip 8464 auf Stecksockel, Best.Nr. 4'502'4718'0

  Dieser Speicher erlaubt das beliebige Einschreiben, Löschen oder Ueberschreiben eines Programmes mit dem Handprogrammiergerät PØ5. Der Speicherinhalt wird bei Spannungsausfall durch die Pufferbatterie auf der CPU während ca. 2 Monaten gesichert. Das Programm ist jedoch nicht transportabel, da es beim Ausstecken des RAM-Chips verloren geht.

- Gepuffertes RAM-Speichermodul Typ PCA1.R95

  Im Gegensatz zum RAM-Chip ist auf diesem Speicher das Programm transportabel, da es durch eine integrierte Elektronik geschützt und durch eine Lithium-Batterie während ca. 8 Jahren gesichert wird.

  Programmierung durch Handprogrammiergerät PØ5.

- EPROM-Chip 2764 auf Stecksockel, Best.Nr. 4'502'4719'0

  Mit EPROM's bleibt ein Programm über ein Jahrzehnt zuverlässig erhalten. Das Programm kann jedoch nicht direkt auf der PCAØ ins EPROM geschrieben werden. Dazu stehen folgende Möglichkeiten zur Verfügung (verlangen Sie die spez. Dokumentation):

  a) mit dem EPROM-Ladegerät PCA2.P16
  b) mit dem Universal-Programmiergerät PCA2.P21
  c) mit den CPU's der Baureihe PCA2 (M31 und M32)

  Jedes EPROM kann fast beliebig oft mit einer entsprechenden UV-Lichtquelle gelöscht werden.

Je nach verwendetem Anwenderspeicher müssen die Wahlbrücken auf der CPU entsprechend eingestellt werden (siehe auch Abbildung in Kapitel 5).

für R95	für EPROM 2764 + R 94	für RAM 8464
2764 * 8464/R95 2764/R95 8464	2764 2764	8464 * 8464

Standard-Einstellung
ab Werk

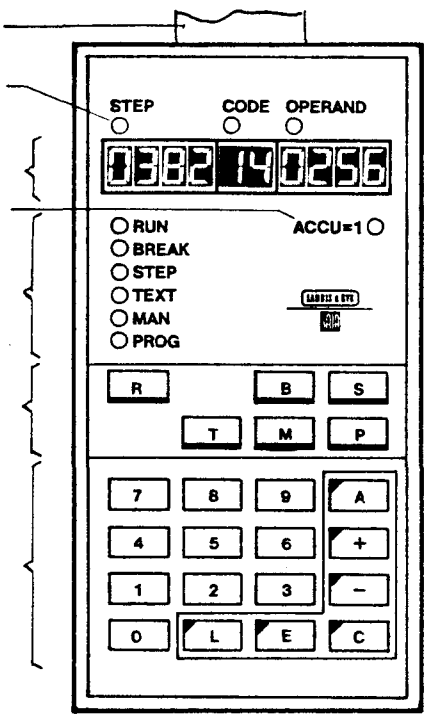
*) Stellung für Schutz gegen Ueberschreiben

P.S. Das Verändern der Brücken soll bei ausgeschalteter SPS erfolgen.

saia®pc

4.3 Die Programmiergeräte

- Das Handprogrammiergerät PCA2.PØ5

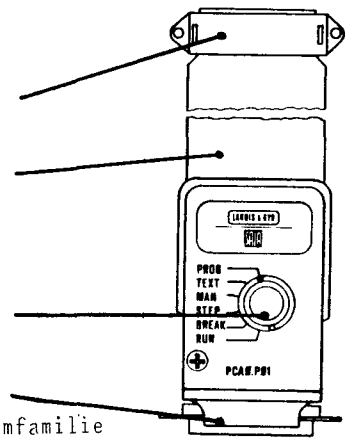


Dieses handliche Programmiergerät wurde besonders für die Baureihe PCAØ entwickelt. Es kann aber auch für die Baureihen PCA1 und PCA2 direkt benützt werden.

Alle Betriebsarten lassen sich durch Tasten anwählen. Die Programm-Eingabe erfolgt in der Betriebsart "PROG" mittels einer 10er-Tastatur im leicht erlernbaren Zahlencode. Alle Elemente (Eingänge, Ausgänge, Timer, Zähler) lassen sich in der Betriebsart "MAN" abfragen oder verändern.

Timer- und Zählerwerte können im RUN-Betrieb angezeigt werden. In der Betriebsart "STEP" kann auf jeden Anwenderschritt des 4K-Anwenderspeichers gesprungen werden. "BREAK" schliesslich erlaubt den Programmablauf bis zu einem gesetzten Breakpoint und weiterfahren im Einzelschritt. Details siehe Betriebsarten in Kapitel 8.

- Das Programmier-Interface PCAØ.PØ1

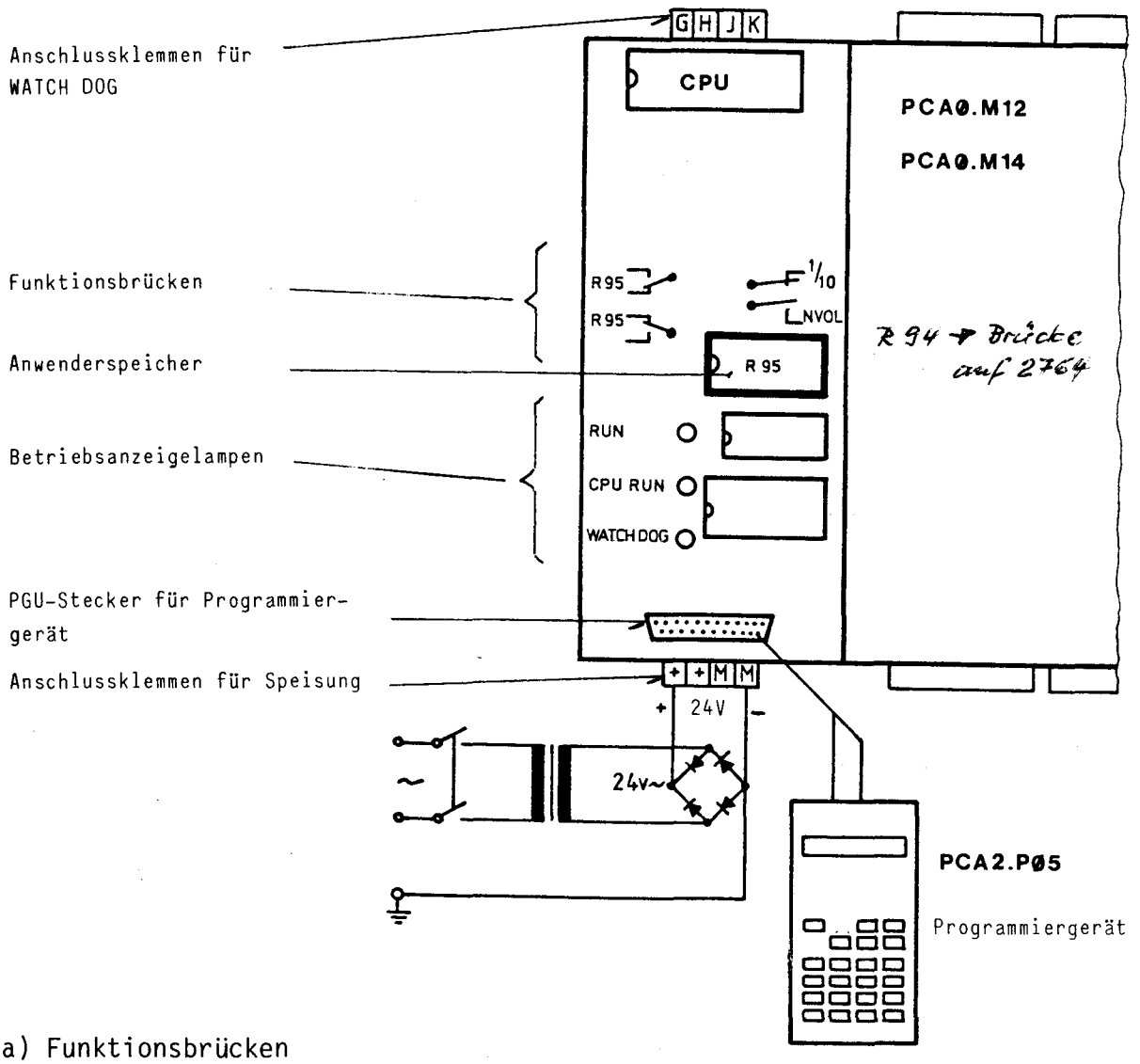


Mit diesem Interface lassen sich alle SAIA-PC-Programmiergeräte auch an der Baureihe PCAØ anschliessen. Es sind dies:

- P10 Handprogrammiergerät mit Zahlencode
- P18 Handheldcomputer mit vielfältigen Möglichkeiten (ab Soft Version V18-Ø4)
- P21 Universal-Programmiergerät
- IBM-PC mit SAIA®MACRO-ASSEMBLER

Damit steht das gesamte aufwärtskompatible Instrumentarium der SAIA-PC Systemfamilie auch für die PCAØ zur Verfügung.

5. Schnellanleitung zur Handhabung der PCAØ



a) Funktionsbrücken

(1) Im Anlieferungszustand sind die Funktionsbrücken wie folgt eingehängt:

- Zeitbasis "1/10" eingehängt (für Zeitbasis von 1/10 sec.)
- Nullspannungssicherheit für Merker und Register "NVOL" (nonvolatile) ~~nicht~~ eingehängt
- Brücken für Anwenderspeicher gemäss obenstehender Zeichnung für gepufferten RAM-Baustein PCA1.R95

Sollten sich die Brücken nicht in dieser Position befinden, so sind sie mittels eines kleinen Schraubenziehers entsprechend zu ändern. Selbstverständlich ist für den Zugang zur CPU der Deckel mittels zweier Schrauben zu entfernen.
Das Verändern der Brückenstellung soll aber nur bei ausgeschaltetem Gerät erfolgen.

b) Stromversorgung

(2) Man nehme einen Trafo (zum "Spielen" genügen 20 VA) mit einer Sekundärspannung von 24V~ und schliesse die Klemmen + und M der PCAØ über einen Brückengleichrichter an. (Im Simuliergerät PCA2.SØ5 ist diese Speisung bereits enthalten, siehe Abschnitt e).

(3) Ein Schalter ergibt den Vorteil, dass durch Ausschalten der PCAØ auf einfache Art alle rücksetzbaren Elemente und der STEP-Zähler in die definierten Ausgangslagen gebracht werden können.

c) Einsetzen des Anwenderspeichers R95 und des Programm-Eingabegerätes PØ5

(4) Das gepufferte RAM-Modul PCA1.R95 ist in der gezeichneten Lage (Nutenbezeichnung links) auf den leeren Anwender-Stecksockel zu stecken.

(5) Das Programm-Eingabegerät PCA2.PØ5 wird am 25-poligen PGU-Stecker angeschlossen.
Wird ein anderes Programmiergerät als PØ5 verwendet, so muss das Interface PCAØ.PØ1 dazwischengeschaltet werden.

d) Programmbeispiel "Blinker"

(6) Spannungsversorgung einschalten. Gelbe Lampe "CPU RUN" blinkt jetzt im 2 sec.-Takt (1 sec. ein, 1 sec. aus).

(7) Am Programm-Eingabegerät durch Betätigen der Taste [P] (mind. 0,5 sec) die Betriebsart PROG wählen. Als Quittung leuchtet auf dem PØ5 die rote LED PROG auf.

(8) Eintippen des folgenden Blinker-Programmes:

	STEP	CODE	OPERAND	Programm im Mnemocode
A, Ø, E	(0000) *	(00)	(0000)	
E	(0001)	02	256	→ STL 256
E	(0002)	14	256	STR 256
E	(0003)	00	5	0,5s
E	(0004)	13	24/40 **	COO 24/40 **
E	(0005)	20	1	JMP 1
E	(0006)	(00)	(0000)	

*) Werte in Klammern müssen nicht eingetippt werden, werden aber angezeigt

**) Für die kleine PCAØ.M12.. ist Ausgang 24, für die grosse PCAØ.M14.. ist Ausgang 40 einzugeben.

(9) Programmzähler auf Null stellen:

Tastenfolge [S] Betriebsart STEP, rote LED STEP leuchtet als Quittung

[A] Adresse, [Ø] [+]

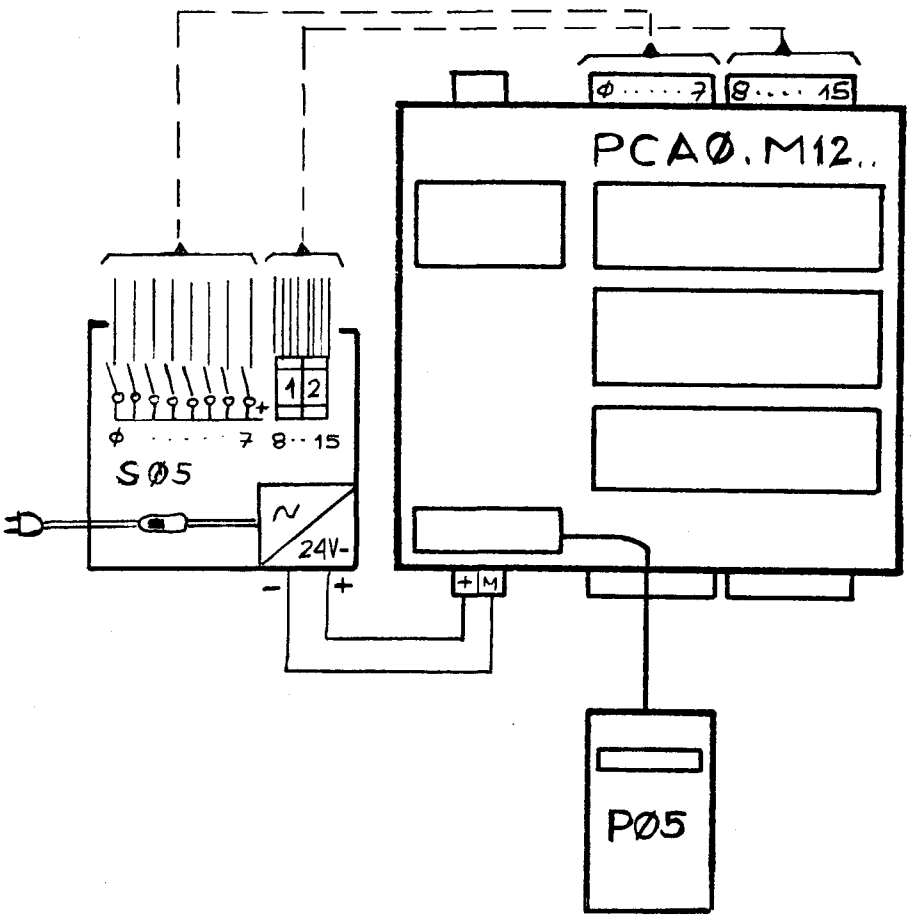
(10) Betriebsart RUN wählen:

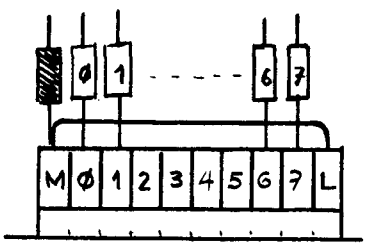
Taste [R] (RUN) 0,5 sec. betätigen

--► Rote LED RUN auf PØ5 leuchtet
--► Grüne Lampe RUN auf PCAØ leuchtet
--► Ausgang 24 bzw. 40 blinkt mit 0,5s ein und 0,5s aus (Frequenz 1 Hz)

e) Anschluss des Eingangs-Simuliergerätes PCA2.SØ5

Unter Einbezug des E-Simuliergerätes PCA2.SØ5 entsteht ein kompletter Programmier- und Uebungsplatz. Mit ihm können alle Programmbeispiele dieses Handbuches "durchgespielt" werden.



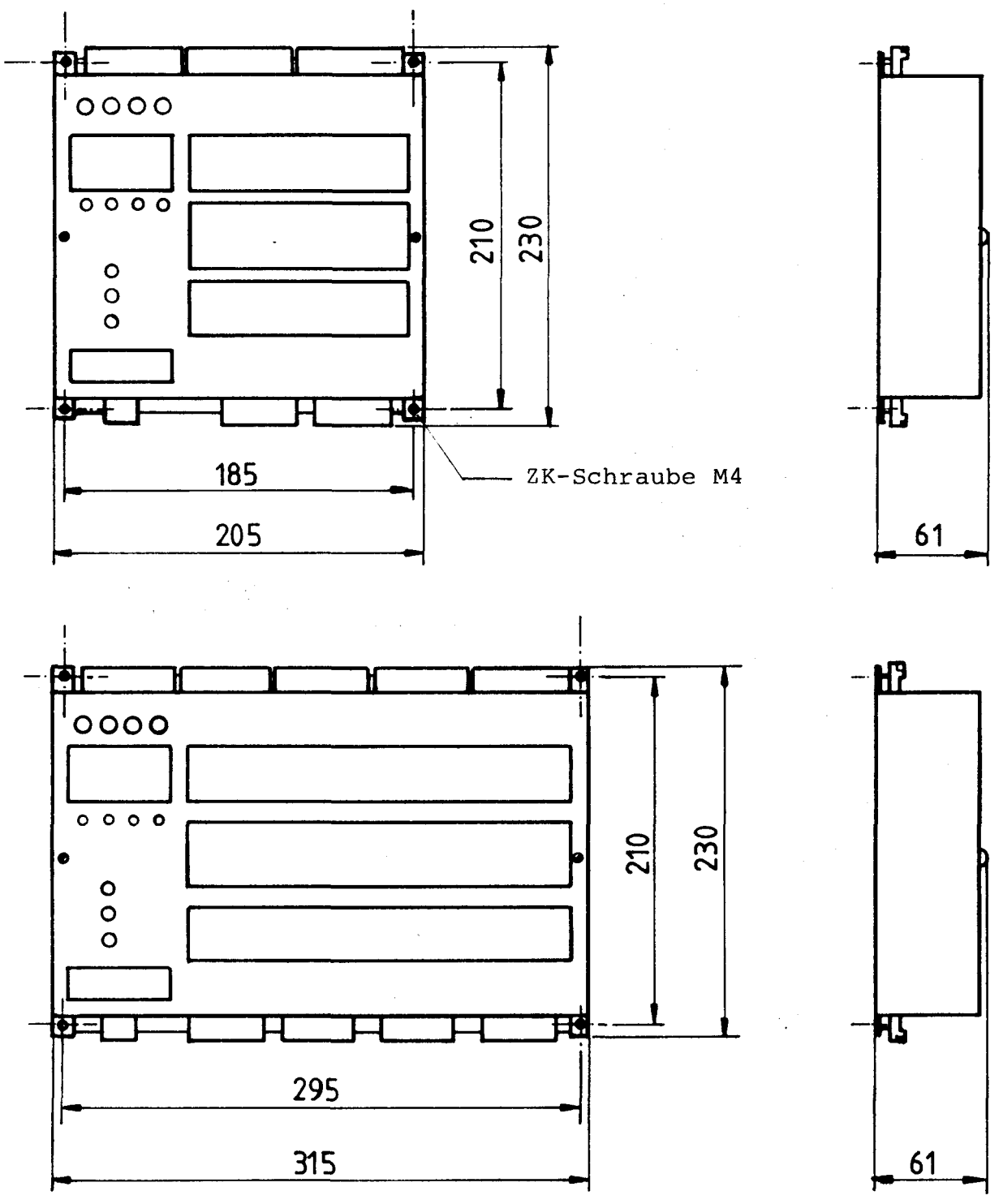


Anschluss des Kabels vom SØ5 am Eingangsstecker Ø...7. Verbindung M-L ergibt Quellbetrieb.

Anstelle des E-Simuliergerätes PCA2.SØ5 kann auch das grössere PCA2.S10 mit dem Uebergangskabel PCA1.K80 verwendet werden.

saia®pc

Massbild PCAØ

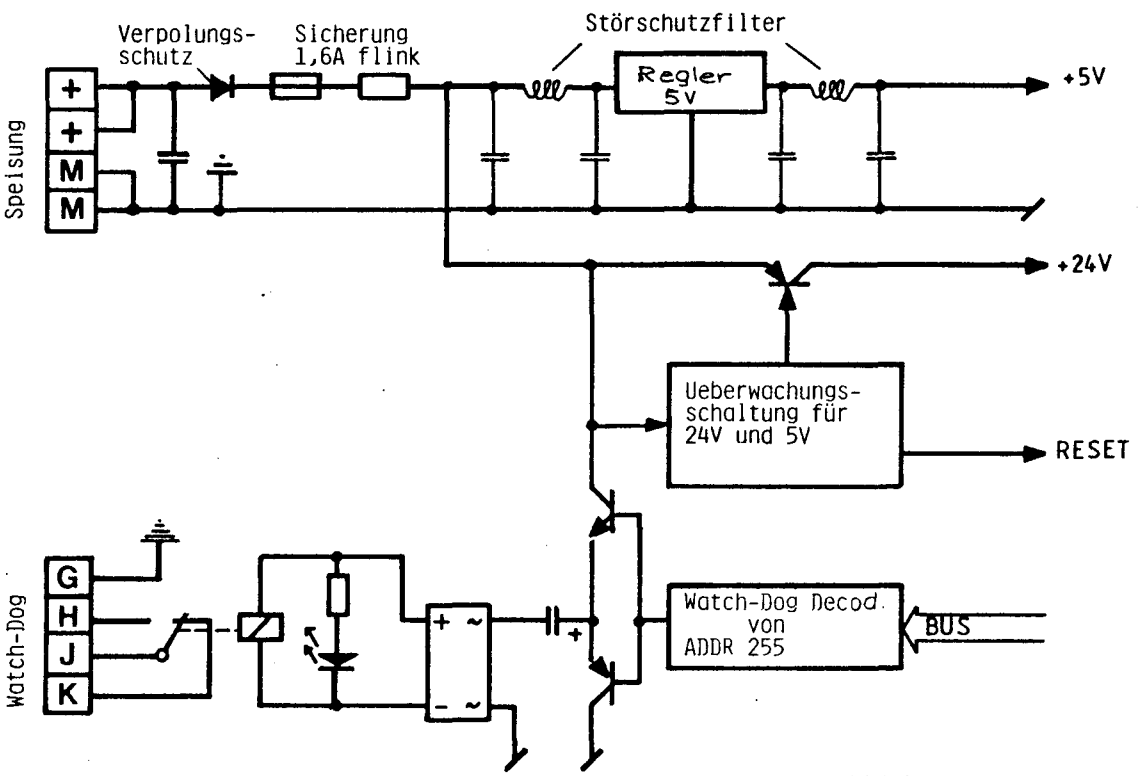


6. Details über Stromversorgung und Watch-Dog

6.1 Stromversorgung der PCAØ

Speisespannung Ue	24V- geglättet oder pulsierend
Toleranz für Ue	
- allgemein	± 20 %
- für Ausführung mit Relais	pulsierende Spannung ± 20 % (Tu = Ø ... 50°C) geglättete Spannung + 20 % (Tu = Ø ... 35°C) - 5 %
Speisestrom	
- PCAØ.M12T	max. 0,5A(inkl. PØ5 angeschlossen)
- PCAØ.M14R	max. 0,9A(inkl. PØ5 angeschlossen)
Sicherung	1,6A flink

Mehrere Komponenten schützen die PCAØ gegen Störspannungen, gegen falsche Polung und gegen Spannungsunterschreitung. Mit einem Schaltregler werden die 5V zur Speisung der Elektronik-Komponenten erzeugt.



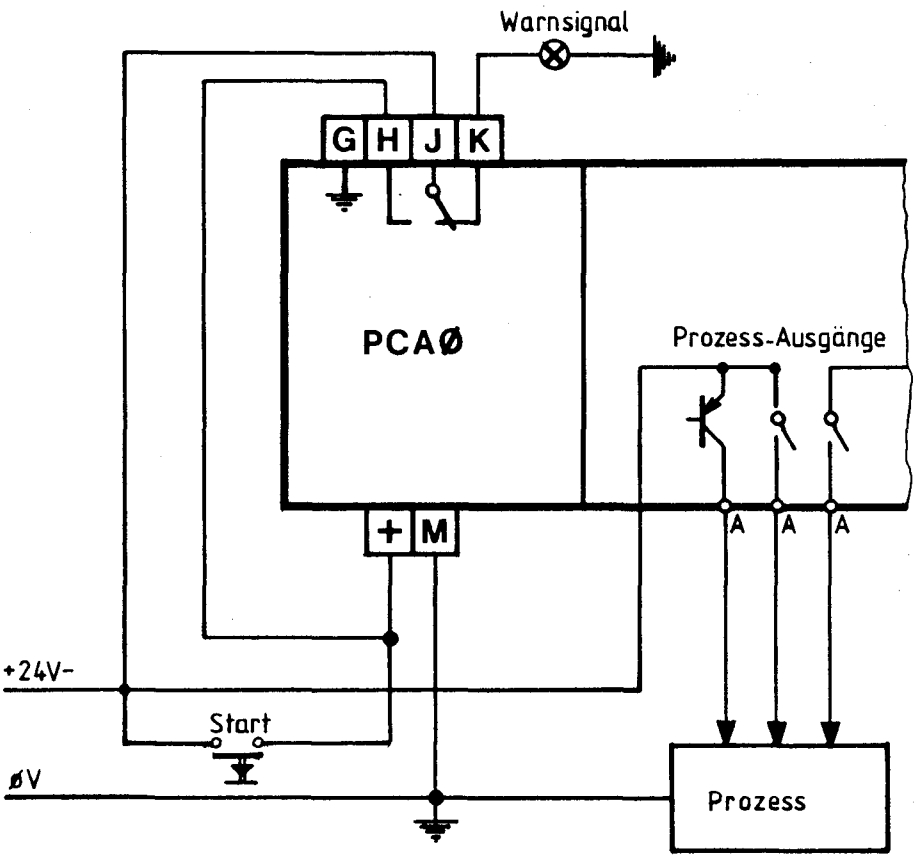
6.3 Die WATCH DOG-Ueberwachungsschaltung

Mit der WD-Schaltung kann das richtige Abarbeiten des Anwenderprogrammes mit hoher Zuverlässigkeit überwacht, und im Fehlerfall können wirksame Sicherheitsmassnahmen vorgekehrt werden.

Das WD-Relais ist solange erregt (Kontakt H-J geschlossen), wie der Adresse 255 ein Wechselsignal von ≥ 5 Hz zugeführt wird. Dieses Signal wird auf einfache Weise mit dem Befehl COO 255 in einem Umlaufprogramm erzeugt. Bei normalem Arbeiten der CPU im RUN-Betrieb bleiben somit die Klemmen H-J verbunden, die grüne WD-Lampe leuchtet. Sollte in der CPU eine Störung auftreten oder eine andere Betriebsart gewählt werden als "RUN", so wird Kontakt H-J geöffnet, die WD-Lampe verlöscht.

Für sicherheitstechnisch kritische Anlagen wird die Benützung der WD-Funktion zusammen mit der untenstehenden Sicherheitsschaltung empfohlen. Bei Abfall des WD-Relais wird die PCAØ spannungslos, wodurch auch alle Ausgänge sofort zurückgesetzt werden.

Schaltleistung des WD-Kontaktes: 1,5A, 48V≃

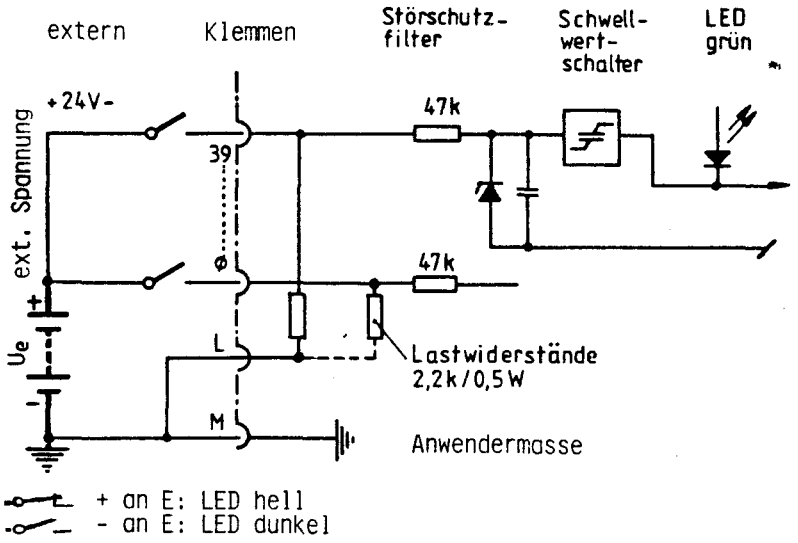


7. Details über Eingänge und Ausgänge

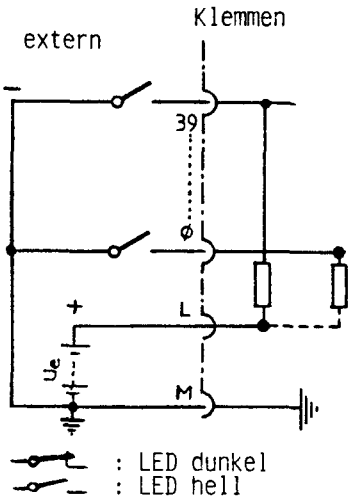
7.1 Eingänge 24V- (B90)

Eingangsspannung Ue	24V-, geglättet oder pulsierend
Spannungspegel	H: 19 ... 32V- L: Ø ... 4V-
Eingangsstrom bei 24V-	10 mA
Eingangsverzögerung	9 ms
Betriebsart	Quell- oder Senkbetrieb, je nach Anschluss

Quellbetrieb



Senkbetrieb



Klemmenbelegung

M	Ø	1	2	3	4	5	6	7	L

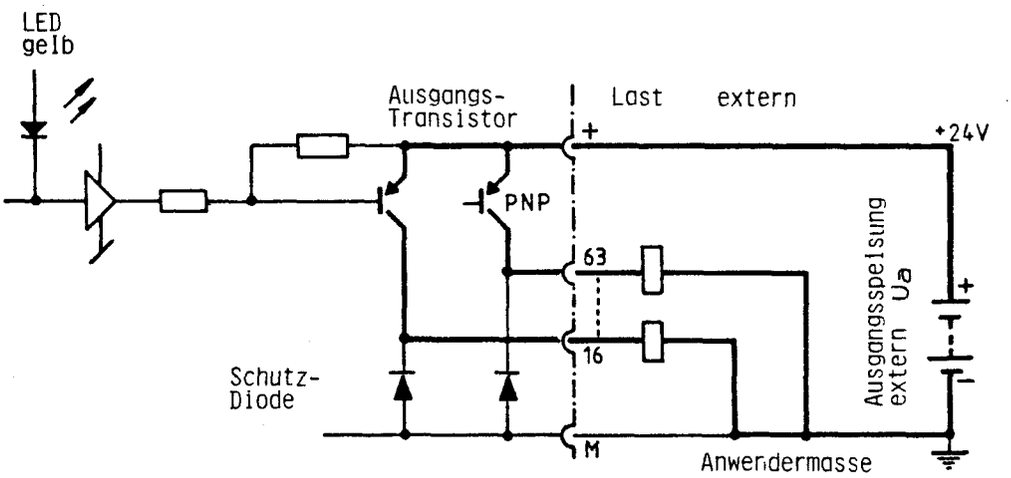
*) Für die Version PCAØ.M12T können die Eingänge 16 ... 19 nur im Quellbetrieb betrieben werden.

Klemmenbelegung

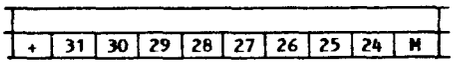
+	23	22	21	20	19	18	17	16	M
		A				E			

7.2 Ausgänge Transistor (B90) Typ T

Ausgangs-Strombereich	5 mA ... 0,5 A Im Spannungsbereich von 5 ... 24V- soll der Lastwiderstand mind. 48 Ohm betragen
Spannungsbereich Ua	5 ... 36V-, geglättet oder pulsierend
Spannungsabfall	max. 1,5V bei I = 0,5A
Betriebsart	Quellbetrieb (Plus ~~pulsschaltend~~)

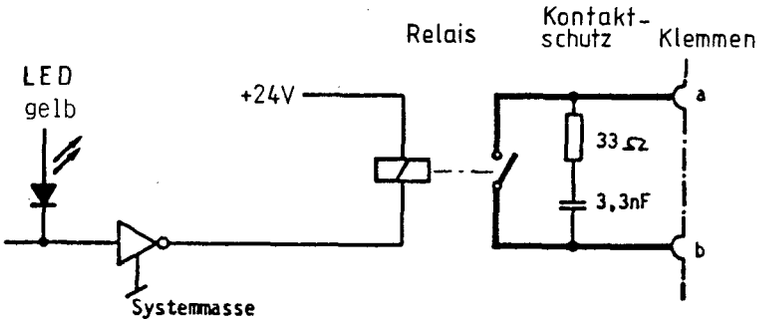


Klemmenbelegung

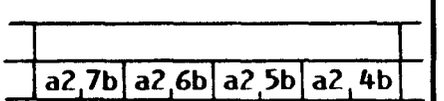


7.3 Ausgänge Relais (A21) Typ R

Kontaktart	Reinsilber, Schliesskontakt
Schaltleistung	3A, 250V~ AC1 1A, 250V~ AC11 3A, 24V- DC1 * 1A, 24V- DC11 *
Kontaktschutz	3,3 nF mit 33 Ohm
Kontaktlebensdauer (AC 1)	3A, 220V~ : 0,1 Mio Schaltspiele 1,5A, 220V~ : 0,5 Mio Schaltspiele 0,3A, 220V~ : 5 Mio Schaltspiele



Klemmenbelegung



*) Zur Schaltung von Gleichspannung 24V sind aus Gründen der Lebensdauer und der Schaltsicherheit Transistorausgänge vorzuziehen.

Die serielle Daten-Schnittstelle

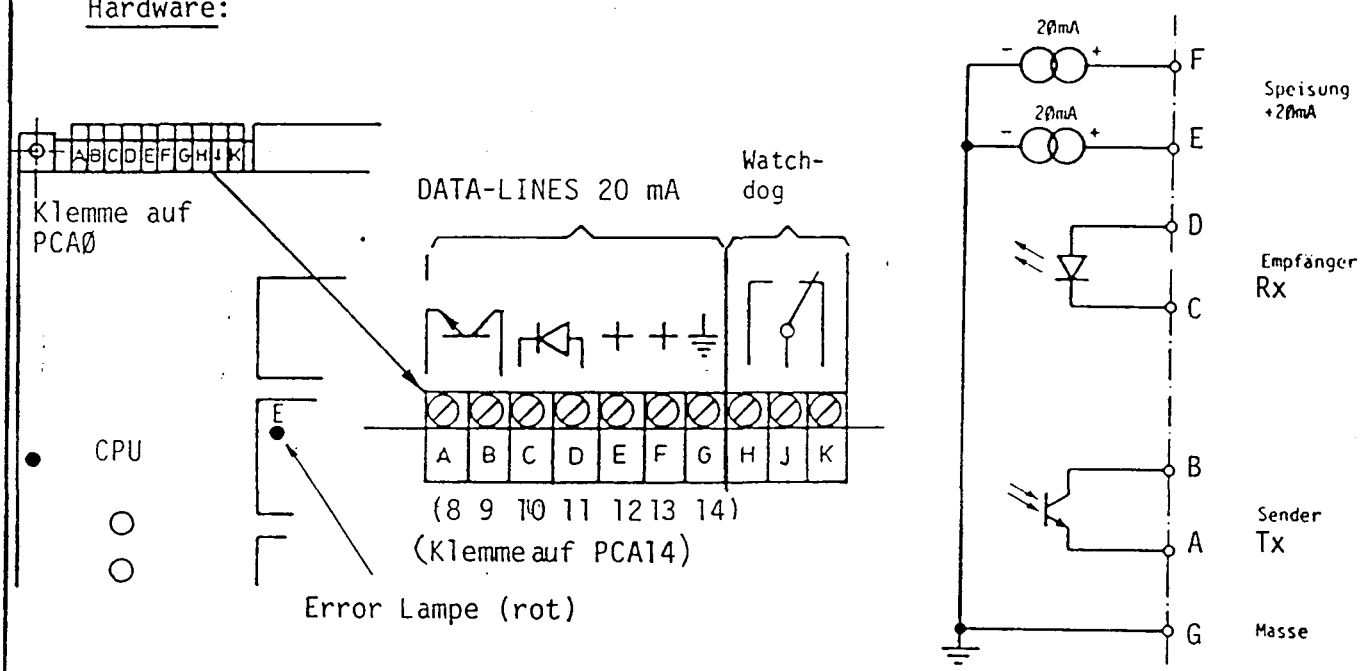
Funktionen durch Assignierung mit Befehl PAS 100:

- Modus E: Texteingabe via Editor
- Modus T: Intelligente Textausgabe
- Modus C: Ein- bzw. Ausgabe von Einzelcharaktern
- Modus M: Menueführung mittels Textausgabe und Einzeltasten
- Modus N: Ein- bzw. Ausgabe von numerischen Daten
- Modus P: Direkteingabe bzw. lesen von PC-spezifischen Daten
- Kombinationsmodus: Sinnvolle Kombination der Modi T bis P

Die PCAØ kann in dieser Ausführung nicht nur an jeder Standard-Terminal zur menuegeführten Prozess-Steuerung angeschlossen werden, sondern die PCAØ kann auch mit anderen intelligenten Systemen direkt oder über das SAIA-LAN 1 korrespondieren. Auch ein Anschluss aus Leitsystem (SAIA) PMS 1 ist möglich.

Sowohl bezüglich Hardware als auch Software ist die serielle Schnittstelle auf der PCAØ vollkommen kompatibel zur Schnittstelle auf der PCA14.

Hardware:



Unter Beachtung der unterschiedlichen Klemmenbezeichnungen ist die Schnittstelle vollkommen kompatibel zu PCA14. Details siehe Handbuch PCA14 Teil C3.3.

Anwender-Software:

Volle Kompatibilität mit allen PAS 100-Modi gemäss Handbuch PCA14 Teile C3.4/ C3.5 und C3.6.

8. Die Betriebsarten der PCAØ

Um Programme zu erstellen und auszutesten, ist man darauf angewiesen, eine SPS in verschiedene Betriebsarten zu versetzen. Dies geschieht via Tastendruck mit dem Kleinprogrammiergerät PCA2.PØ5 oder für andere Programmiergeräte mit dem Programmier-Interface PCAØ.PØ1.

Bitte beachten Sie, dass diese Betriebsarten-Tasten aus Sicherheitsgründen mind. 0,5s zu betätigen sind. Die Quittung für die effektive Wahl der entsprechenden Betriebsart wird über die Anzeige-LED des Programmiergerätes PØ5 gegeben. Wird der Stecker des Programmiergerätes an der PCAØ ausgesteckt, so bleibt die gewählte Betriebsart erhalten.

Beim Einschalten der PCAØ stellen sich folgende Betriebsarten automatisch ein:

- Bei eingestecktem Programmiergerät -➔ STEP (die LED "STEP" am PØ5 leuchtet, die grüne LED "RUN" auf der PCAØ leuchtet nicht !)
- Ohne Programmiergerät -➔ RUN (die grüne LED auf der PCAØ leuchtet).

8.1 Diese Betriebsarten sind (Uebersicht):

R	RUN	Normaler Programmablauf (Lampe RUN auf PCAØ brennt)
P	PROG	Ein Anwender-Programm kann auf einen RAM-Speicher (aufgesteckt am Anwender-Stecksockel der PCAØ) eingegeben werden.
M	MAN	Manuelles Abfragen und Setzen von Elementen (Eingänge, Ausgänge, Merker, Timer, Zähler).
S	STEP	Springen auf vorgewählte Schrittadresse des Anwenderprogrammes und Einzelschritt-Abarbeitung
B	BREAK	Programmabarbeitung bis zu einem gesetzten "Breakpoint" mit anschliessendem Einzelschritt-Betrieb
T	TEXT	Hat für die Standardausführung PCAØ keine Funktion

saia®pc

8.2 Detailbeschreibung der Betriebsarten

[R] RUN

Normaler Programmablauf

Beim Einschalten ohne Programmiergerät geht die PCAØ automatisch in RUN

[P] PROG

Programmieren

Ein Programm kann auf einem RAM-Speicher (auf dem Anwender-Stecksockel der PCAØ) neu eingegeben oder überschrieben (korrigiert) werden.

Step		Code	Operand	
[A] x x x x	[E]	x x	x x x x	
	[E]	x x	x x x x	bzw. [C] **Löschen einer falsch eingegebenen Zeile**
	. . . .			
	[+]	Schliesst Eingabe ab.		

Programm prüfen [+] [+] bzw. [−] [−]

[M] MAN **

Manuelles Abfragen bzw. Setzen von Elementen

(Elemente = Eingänge, Ausgänge, Merker, Zähler, Timer)

Abfragen: [A] x x x (Step / Elementadresse) ⟶ logische Anzeige im Operand (0/1)

Setzen: [A] x x x [E] [1] ⟷ bzw. [0]

Elementadresse

[S] STEP

[+] ⟶ Anzeige, *wo* Programm steht.

Springen auf vorgewählte Schrittadresse des Anwenderprogrammes

[A] 139 [+] ⟶ Programm springt auf Schritt 139, anschliessend

[+] [+] . . . . schrittweises Abarbeiten des Programmes, wobei Verknüpfungsergebnis überprüfbar ist ☀ ACC = 1.* Jederzeit Umschalten auf RUN möglich.

Bei Parallelprogrammen wird auf STEP *nur das angesprungene Parallelprogramm* abgearbeitet.

[B] BREAK

Unterbruch des Programmablaufes mit anschliessendem Einzelschritt

[+] ⟶ Anzeige, *wo* Programm steht.

[+] [+] . . . . schrittweises Abarbeiten des Programmes, wobei Verknüpfungsergebnis überprüfbar ist ☀ ACC = 1.* Jederzeit Umschalten auf RUN möglich.

Bei Parallelprogrammen werden *alle Programme* «parallel» abgearbeitet (wie auf RUN).

Setzen eines «Breakpoint»

[A] 820 [+] ⟶ Programm läuft bis Schritt 820, anschliessend

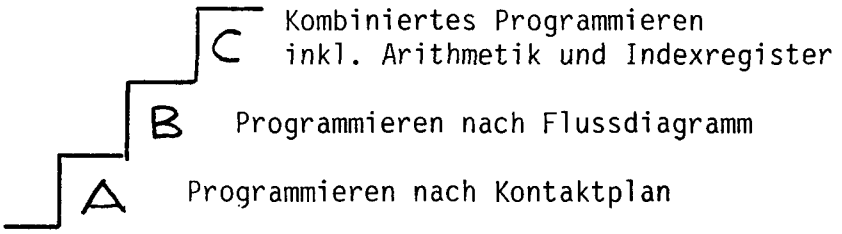
[+] [+] . . . . . Einzelschritt über «kritische» Stelle hinweg.

---

*) Mit ACC = Accumulator wird der Zustand des Verknüpfungsspeichers angezeigt. Ist ACC = 1 (Verknüpfungsresultat = 1), so werden die nachfolgenden Schaltbefehle ausgeführt.

**) Wird der Adresse eines Timers oder Zählers eine 3 voraus eingegeben (z.B 3260 für Zähler 260), so kann der Wert dieses Registers gelesen bzw. mit [E] Wert [+] auch manuel eingegeben werden.

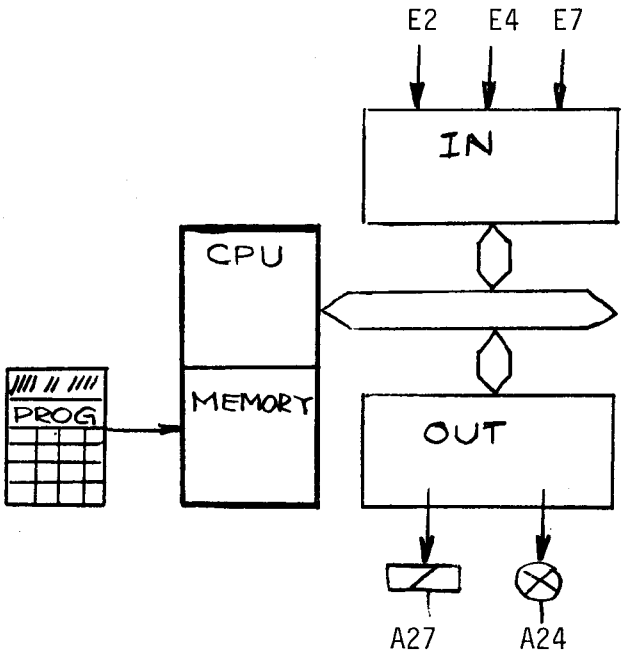
9. Programmieren in drei leichten Schritten



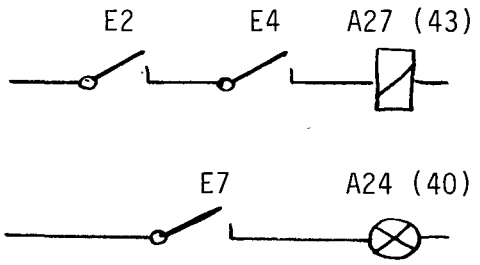
Die PCAØ besitzt einen sehr leistungsfähigen Befehlssatz des Niveaus (1H) . Damit können auch komplexe Steuerungsprobleme elegant gelöst werden. Die Programme der PCAØ sind auch jederzeit direkt auf andere Baureihen der SAIA-PC Systemfamilie übertragbar. Ihre Steuerung kann somit Ihren Ansprüchen gemäss wachsen, ohne dass die Programme jedesmal neu geschrieben werden müssen.

Um dem Einsteiger aber den Beginn des Programmierens zu erleichtern, kann die Leistungsfähigkeit der PCAØ in drei leichten Treppenschritten kennengelernt werden. Vielleicht ist Ihr Steuerungsproblem aber so einfach, dass es bereits auf Stufe A gelöst werden kann.
Beginnen wir also beim Programmieren nach Kontaktplan, obschon sich dieser für eine moderne Programmierung nicht eignet, weil Prozessabläufe besonders in Relaismatik oft nur schwer darstellbar sind.
Aber Sie werden sehen, Ihre PCAØ versteht jeden Kontaktplan.

(A) Programmieren nach Kontaktplan



Kontaktplan



Programm

```
1   STH   2
2   ANH   4
3   OUT  27
4   STH   7
5   OUT  24
    :
    JMP   1
```

Das Programm, entsprechend dem Kontaktplan, wird mit dem Programmiergerät in den Anwenderspeicher (RAM) eingetippt. Im RUN-Betrieb liest die CPU dieses Programm Zeile um Zeile und fragt die entsprechenden Eingänge ab. Ist einer der Kontakte E2 bzw. E4 geschlossen, so wird in der CPU ein "H" (High = Spannung grösser +19V) abgespeichert und nach dem Lesen der zweiten Zeile mit dieser UND-verknüpft. Sind beide Kontakte geschlossen (H), so ist der Verknüpfungsspeicher = 1 (ACCU = 1) und der Ausgang 27 wird aktiviert, das Schütz an Ausgang 27 schaltet ein.

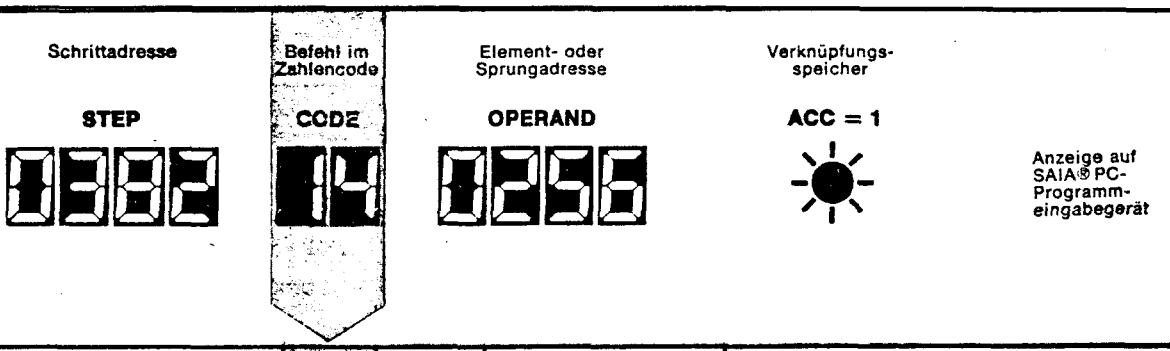
Auf Zeile 4 fragt die CPU den Eingang 7 ab. Sofern der Kontakt geschlossen ist, wird der Ausgang 24 aktiviert, die Meldelampe leuchtet.

Wir können weitere solche Verknüpfungen aneinanderreihen, denn unser Anwenderspeicher ist mit 4 K = 4096 Programmzeilen ja riesig gross. Sind alle Verknüpfungen programmiert, so müssen wir der CPU sagen, dass sie wieder auf Zeile 1 zurückspringen soll. Auf diese Weise wird unser Programm mit hoher Geschwindigkeit zyklisch dauernd durchlaufen, und alle Veränderungen werden an den Eingängen als Verknüpfungsresultat sofort auf die Ausgänge übertragen.

Befehlssatz (A) Kontaktplan

Die Befehle, welche wir zur Programmierung solcher Verknüpfungen zur Verfügung haben, gliedern sich in Logikbefehle und Schaltbefehle.

Die nachstehende Tabelle nennt ihre Funktionen. Lassen Sie sich von der Vielfalt nicht verwirren. In der Praxis der Kontaktplanprogrammierung werden nur etwa 8 davon häufig gebraucht, die man daher schnell auswendig kennt.



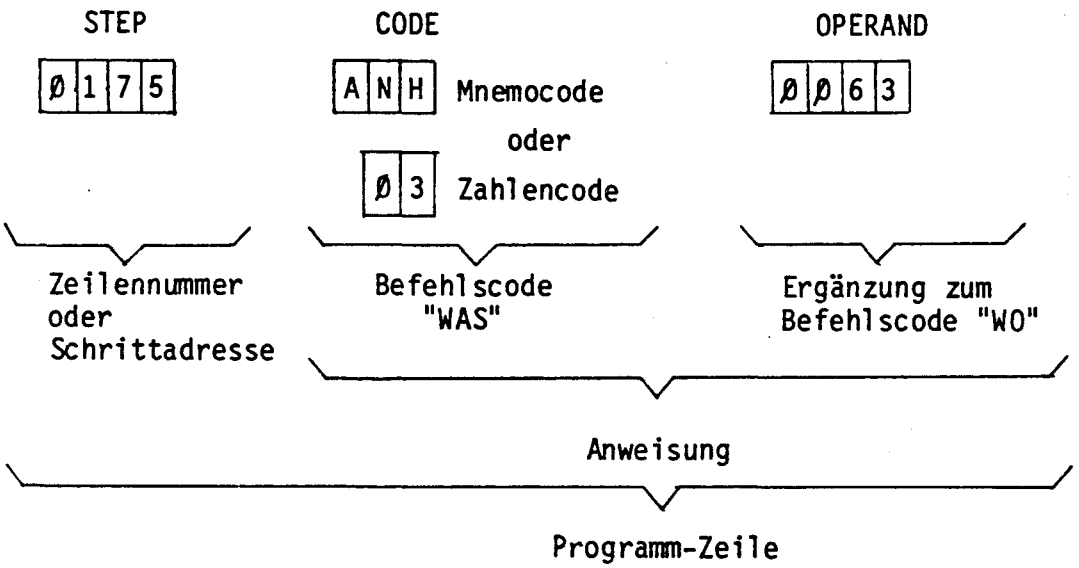
	Zahlen-code	Mnemo-code	Befehl englisch	Beschreibung
Logikbefehle	Ø1	STH	Start High	Beginn einer Verknüpfung mit Element abgefragt auf ⌐ High
	Ø2	STL	Start Low	Beginn einer Verknüpfung mit Element abgefragt auf ⌙ Low
	Ø3	ANH	AND High	UND-Verknüpfung zwischen ACCU und Element abgefragt auf ⌐ High
	Ø4	ANL	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
Zeitbefehle	14*	STR*	Set Timer	Setze Zeitglied auf vorgewählten Wert und starte es
Sprungbefehle	2Ø	JMP	Unconditional Jump	Unbedingter Sprung auf Schrittadresse
Hilfsbefehle	ØØ	NOP	No Operation	Keine Operation

* zweizeilige Befehle (zweite Zeile enthält vorgewählten Wert)

Die Programmzeile

Jede Anweisung im Anwenderprogramm besteht aus 1 Programmzeile (in gewissen Fällen aus 2 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.

Aufbau der Programm- bzw. Befehlszeile:



STEP — Mit der Zeilen-Nr. wird der Platz der Anweisung im Anwender-Speicher festgelegt. Dezimale Numerierung von Ø ... 4095 (4K).

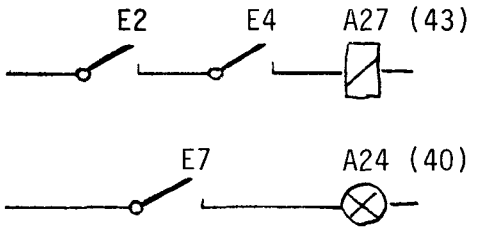
CODE — Je nach Programmiergerät kann der Befehlscode als 3stelliger Mnemocode oder als Zahlencode von Ø bis 31 eingegeben werden; Der Mnemocode ist die Abkürzung des englischen Ausdruckes für den entsprechenden Befehl. Er ist daher leicht im Gedächtnis zu behalten und auch international verständlich.

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.

A1 Ein erstes Programmierbeispiel

Bevor mit der Programmeingabe begonnen werden kann, muss das Programm im Mnemocode auf Papier festgehalten werden. Dazu verwendet man mit Vorteil die Programmierlisten, wie sie am Ende dieses Manuals zum Kopieren angefügt sind.



Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
Ø	NOP	ØØ	Ø	Leerzeile
1	STH	Ø1	2	Abfrage E2
2	ANH	Ø3	4	UND E4
3	OUT	10	27	Ausgang A27
4	STH	Ø1	7	Abfrage E7
5	OUT	1Ø	24	Ausgang A24
6	JMP	2Ø	1	Rücksprung
7				

Zur Programmeingabe und zur Simulation richten wir uns den gleichen Programmierplatz ein, wie er bereits in Abschnitt 5 beschrieben worden ist. Alle nachfolgenden Beispiele beziehen sich dabei immer auf die Verwendung der kleinen PCAØ mit nur 16 bzw. 20 Eingängen. Für die grösseren Ausführungen sind jeweils die Ausgangsadressen in Klammern beigefügt.

Mit dem Programmiergerät PCA2.PØ5 kann nun das obige Programm in den eingesteckten Anwenderspeicher (RAM oder R95) eingetippt werden:

P --► Betriebsart PROG

A Ø E 1)
2) E Ø 1 2
E Ø 3 4
E 1 Ø 2 7
E Ø 1 8
3) C Ø 1 7
E 1 Ø 2 4
E 2 Ø 1
4) +

	Schritt (STEP)	Mnemo-CODE	Zahlen-CODE	OPERAND
	0000	(NOP)	00	0000
→	1	(STH)	01	2
	2	(ANH)	03	4
	3	(OUT)	10	27
	4	(STH)	01	8* 3)
	4	(STH)	01	7
	5	(OUT)	10	24
	6	(JMP)	20	1

1) Anwahl der Schrittadresse Ø (A = Adresse) und löschen des Inhalts durch E = Enter.

2) Mit jedem weiteren E wird die Schrittadresse um 1 erhöht und der Inhalt der alten Programmzeile gelöscht und für die neue Eingabe vorbereitet.

3) Anstatt 7 wurde falscherweise 8 eingetippt. Korrektur durch C und Wiederholung der Anweisung. Die Schrittadresse wird durch C nicht erhöht.

4) Die letzte Eingabe muss mit + , E , A oder - abgespeichert werden.

Nach der Programmeingabe empfiehlt es sich, das tatsächlich im Anwenderspeicher abgelegte Programm mit der schriftlichen Vorlage schrittweise zu vergleichen:

[A] [1] [+] [+] [+] .....

Nun wollen wir natürlich prüfen, wie dieses Programm läuft.

[S] ---➤ Die LED für Betriebsart "STEP" leuchtet

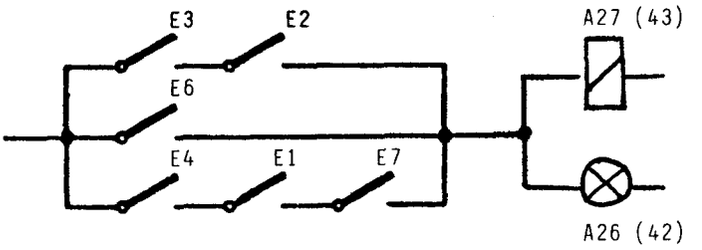
[A] [1] [+] ---> Der Programmablauf soll auf Anwenderschritt 1 beginnen

[R] ---> Die LED "RUN" leuchtet, das Programm läuft

Schliessen Sie nun die Schalter E2 und E4 --➤ die LED von A27 leuchtet auf.
Bei Schliessen von E7 -➤ A24 leuchtet auf.

(A2) Parallelschaltung

Kontaktplan



Programm

Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
→ 10	STH	01	3	Abfrage E3
11	ANH	03	2	UND E2
12	ORH	05	6	ODER E6
13	ORH	05	4	ODER E4
14	ANH	03	1	UND E1
15	ANH	03	7	UND E7
16	OUT	10	27	Ausgabe A27
17	OUT	10	26	Ausgabe A26
18	JMP	20	10	Rücksprung

Hinweise:

- Jeder Befehl OR (ODER) beginnt einen neuen Parallelzweig ganz von vorne links. Es können anschliessend wieder UND-Verknüpfungen in diesen Parallelzweig angefügt werden. Das ganze Programm wird aber nur als eine einzige Verknüpfung bezeichnet.

- Sind die Verknüpfungen abgeschlossen, so können beliebig viele Aktionen (welche von dieser Verknüpfung abhängig sind) angefügt werden.

- Geben Sie dieses Programm ein mit:

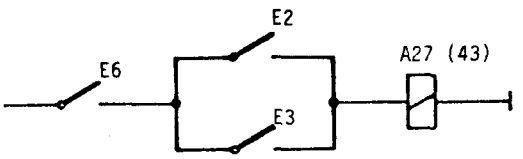
| P | → LED "PROG" für Betriebsart Programmierung leuchtet

| A | | 1 | | 0 | | E | → Die PCA0 ist bereit, ab Schrittadresse 10 das obige Programm aufzunehmen.

Weiteres Vorgehen wie bei Beispiel A1, jedoch mit S A 10 + R das Programm ab Schritt 10 laufen lassen.

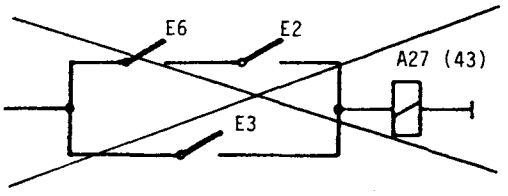
(A3) ODER ist "stärker" als UND bzw. Bekanntschaft mit anderen "Elementen"

Es sei folgende Kontaktanordnung zu programmieren:



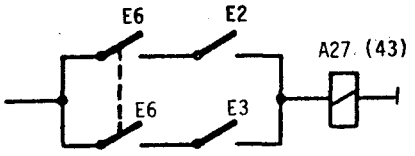

```
STH  6
ANH  3
ORH  3
OUT  27
```

Man ist versucht, das Programm in der dargestellten Weise zu schreiben. Da OR jedoch immer einen Parallelzweig eröffnet, würde dies folgendem Schema entsprechen:



Es bestehen 2 Möglichkeiten für die Realisierung der gewollten Funktion:

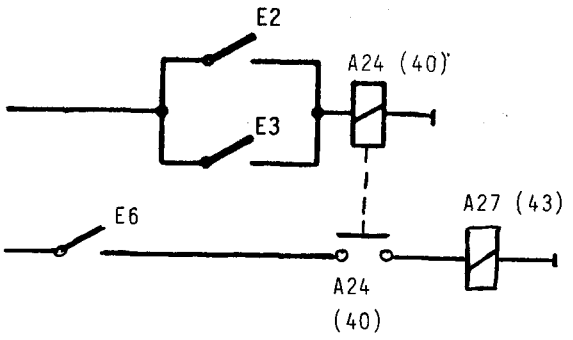
a)




```
20  STH  01   6
21  ANH  03   2
22  ORH  05   6
23  ANH  03   3
24  OUT  10  27
25  JMP  20  20
```

Der Kontakt E6 wird als Doppelkontakt in beiden Parallelzweigen programmiert.

b)




```
20  STH   2
21  ORH   3
22  OUT  24
-------------
23  STH   6
24  ANH  24
25  OUT  27
26  JMP  20
```

Es wird in zwei Schritten programmiert. Aus der Variante b) wird auch ersichtlich, dass Ausgänge beliebig in Verknüpfungen wiederverwendet werden dürfen.

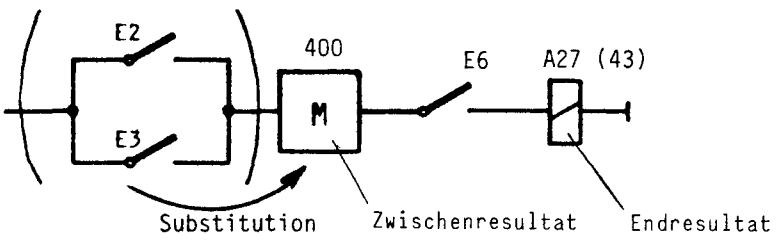
Extra einen Ausgang für diese einfache Aufgabe zu opfern, wäre allerdings schade. Dazu gibt es in jeder PCAØ 712 MERKER !

Die Elemente und Register der PCAØ:

Elemente ADDR.		Register ADDR.
999 … 765	Haftmerker H bzw. nullspannungssichere Merker	
764 … 320	Merker M*	
319 … 288	32 Counter C* die auch als Merker verwendet werden können.	319 … 256: 64 Register * als Counter bzw. Timer zu 16 Bit
287 … 256	32 Timer T* oder Counter C *	
67 … 64	4 Hardwaretimer, sofern das Modul PCA0.H20 montiert ist	
63 … 0	max. 64 Eingänge E und Ausgänge A	

*) Durch Einhängen der Brücke "NVOL" auf der CPU können alle diese Speicherstellen nullspannungssicher gemacht werden, d.h. beim Ausschalten der PCAØ gehen diese Informationen nicht verloren.

Unter Zuhilfenahme eines Merkers kann der Kontaktplan nun wie folgt gezeichnet werden:

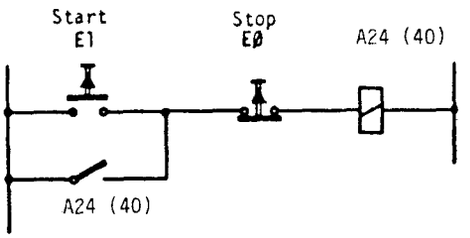



```
ADDR  NC  MNC  OPRD
 30   01  STH     2   Abfrage E2
 31   05  ORH     3   ODER E3
 32   10  OUT   400   Abspeicherung des Zwischenresultates auf Merker 400
-------------------
 33   01  STH   400   Abfrage des Merkers 400(u. damit der ODER-Funktion)
 34   03  ANH     6   UND E6
 35   10  OUT    27   Resultatausgabe auf A27
 36   20  JMP    30   Rücksprung zum Anfang
```

(A4) Start/Stop-Schaltung mit Selbsthaltung auf zwei Arten

a) Nach Kontaktplan

Aus der Schützenschalttechnik ist das folgende klassische Beispiel bekannt:



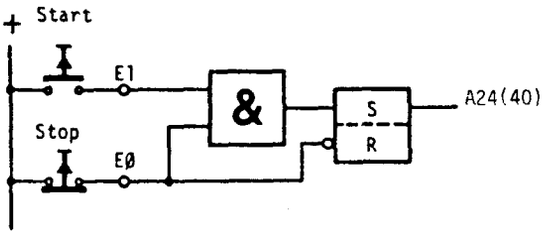
Programm

```
ADDR  NC  MNC  OPRD
 40   Ø1  STH     1   }
 41   Ø5  ORH    24   } Start
 42   10  OUT   401   }
-----------------
 43   Ø1  STH   401   }
 44   Ø3  ANH     Ø   } Stop
 45   10  OUT    24   }
 46   20  JMP    40
```

Analog zum Beispiel A3 wird auch hier mit Hilfe eines Merkers programmiert. Wie das Programm zeigt, wird der Oeffnungskontakt EØ nach "H" verknüpft, da nur bei geschlossenem EØ A24 aktiviert werden kann.

Diese Programmierungsart ist auch sicher gegen Drahtbruch. Bricht nämlich ein Draht in den Leitungen von EØ, E1 oder A24, so wird A24 immer stillgesetzt.

b) Nach Logikplan



Programm

```
ADDR  NC  MNC  OPRD
 50   Ø1  STH     1   }
 51   Ø3  ANH     Ø   } Start
 52   11  SEO    24   }
-----------------
 53   Ø2  STL     Ø   }
 54   12  REO    24   } Stop
 55   20  JMP    50
```

SEO(11): SetOutput Mit diesem Befehl wird ein Element (Ausgang oder Merker) solange gesetzt, bis es mit REO wieder zurückgesetzt wird.

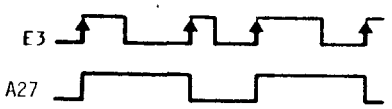
REO (12): Reset Output Mit SEO/REO können wir also die Funktion eines Flip-Flop programmieren. Beide Befehle werden nur ausgeführt, wenn das Verknüpfungsergebnis positiv ist (ACCU = 1).

Der Setz-Befehl im Beispiel b) ist 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.

(A5) Die Befehle DYN und COO im Beispiel Impulsuntersetzer (Schrittschalter)

Unter Impulsuntersetzer (Schrittschalter) verstehen wir die folgende Funktion:



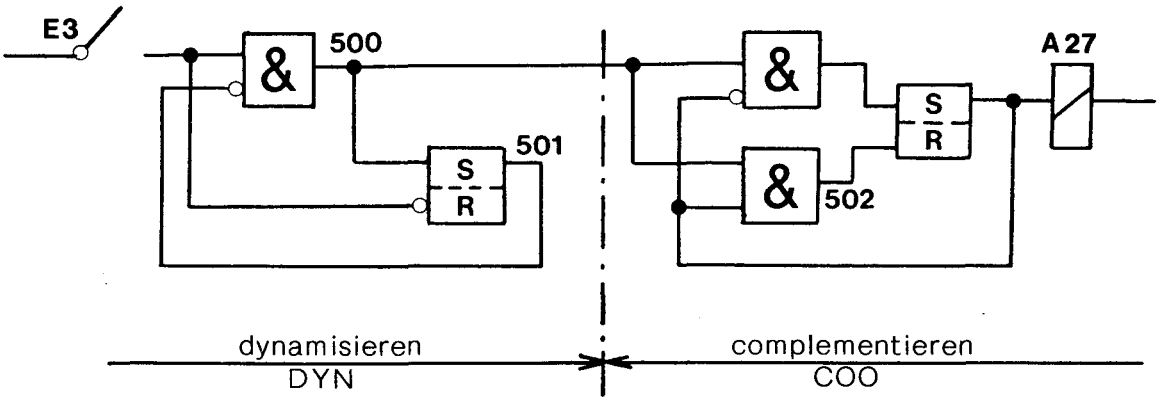
Der Ausgang A27 wird nur bei jeder Einschaltflanke von E3 verändert.

Soll diese Funktion im Kontaktplan nachgebildet werden, so ergibt sich ein recht kompliziertes Schema und ein entsprechend langes Programm.

a) Nach Kontaktplan

Versuchen Sie ruhig selbst ein Relaisschema zu entwerfen, welches diese Funktion darstellt. Es wird nicht einfach sein!

b) Mit SEO/REO obige Funktion nach Logikplan erstellt wird schon wesentlich einfacher.



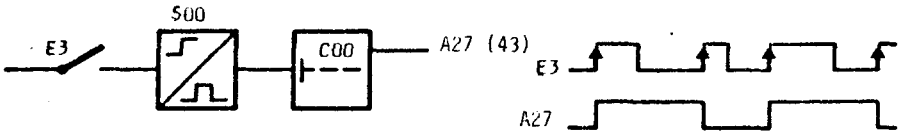
ADDR	NC	MNC	OPRD		
60	01	STH	3	Schalterabfrage	Signal dynamisieren (DYN)
61	04	ANL	501	mit	
62	10	OUT	500	UND-Verknüpfung	
63	11	SEO	501		
64	02	STL	3	Flip-Flop	
65	12	REO	501		
66	01	STH	500	untere	Impuls-untersetzer bzw. Ausgang complementieren (COO)
67	03	ANH	27	UND-	
68	10	OUT	502	Verknüpfung	
69	01	STH	500	obere UND-	
70	04	ANL	27	Verknüpfung	
71	11	SEO	27		
72	01	STH	502	Flip-Flop	
73	12	REO	27	mit Ausgang	
74	20	JMP	60		

c) Mit den Befehlen DYN und COO

DYN (09): Dynamisieren einer Abfragefunktion oder Flankentriggerung. Unter Verwendung des DYN-Befehles (zusammen mit einem Merker im Operand) wird vom vorangehenden Abfragebefehl nur die positive Veränderung (ansteigende Flanke) ausgewertet. Ein Dauerzustand des H-Zustandes oder die abfallende Flanke werden ignoriert.

COO (13): Complement Output. Ein Ausgang oder Merker wird auf seinen logischen Zustand abgefragt und in sein Gegenteil verändert, d.h. ist ein Ausgang gesetzt, wird er durch COO rückgesetzt und umgekehrt.

Mit diesen effizienten Befehlen wird die Lösung dieser Aufgabe denkbar einfach.




```
  ADDR  NC  MNC  OPRD
→ 60    01  STH     3  Abfrage E3
  61    09  DYN   500  Flankentriggerung, abspeichern in Merker 500
  62    13  COO    27  Abfrage von A27 und Umkehrung seines Zustandes
  63    20  JMP    60
```

Wäre in diesem Beispiel der DYN-Befehl weggelassen, so würde bei geschlossenem Schalter E3 bei jedem Programmdurchlauf der Ausgang 27 komplementiert; in dieser kleinen Programmschleife etwa 3000 mal pro Sekunde.

Mit DYN wird beim Schliessen von E3 nur der 1. Programmdurchlauf den Ausgang 27 komplementieren. Jeder weitere Durchlauf hat keine Wirkung mehr, bis sich der Signalzustand von E3 geändert hat und erneut eine Einschaltflanke entsteht.

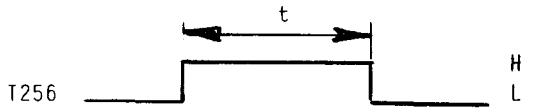
(A6) Die Softwaretimer in Abfallverzögerung

Im Beispiel A3 sind alle Elemente jeder PCAØ-Grundausführung mit ihren entsprechenden Adressen aufgezeigt worden. Die 32 Softwaretimer belegen die Adressen 256 ... 287. Zu jeder Adresse gehört ein 16-bitiges Register, in welchem Werte von 0 ... 65535 abgelegt werden können.

Zum Setzen und Starten eines Softwaretimers ist ein zweizeiliger Befehl erforderlich:

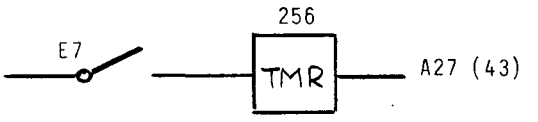
```
STR (14)   256:
ØØ          20:
```

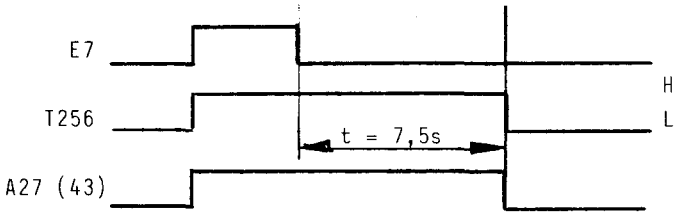
Starten des Timers mit Adr. 256
Laden des entsprechenden Registers mit dem Wert 20.
(Direkteingabe bis max. 2047 bei Code ØØ).



Liest die CPU diesen Befehl, so wird der logische Zustand von Timer 256 "H" gesetzt und der Wert des Registers im Takt der Zeitbasis (1/10 sec.) verkleinert. Nach Ablauf der 20 mal 1/10 sec. = 2 sec. nimmt der Timer 256 wieder den logischen Zustand "L" an.

Mit dieser Grundfunktion lässt sich auf einfache Weise eine Abfallverzögerung realisieren:






```
  ADDR  NC  MNC  OPRD
  70    01  STH     7   Abfrage E7
  71    14  STR   256   Zeitglied setzen und starten  } 2-zeiliger Befehl
  72    00   00    75   Zeiteingabe in 1/10 sec.      }
  ----------------
  73    01  STH   256   Abfrage Zeitglied T256
  74    10  OUT    27   Uebertragung auf A27
  75    20  JMP    70
```

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).

P.S.: Aus Kapitel 5 geht hervor, dass durch Aushängen der Brücke "1/10" auf der CPU der Zeittakt auf 1/100 sec. verkleinert werden kann, wodurch die Auflösung verfeinert wird.

(A7) Verwendung des Hardwaretimer-Moduls PCA0.H20

Um Zeitfunktionen in den Programmen anzuwenden, stehen uns neben den 32 präzisen Softwaretimer (die in jeder Grundausführung enthalten sind) auch 4 Hardwaretimer (zusätzliches Modul) zur Verfügung.

Die 4 Hardwaretimer mit den Adressen 64 ... 67 können ähnlich behandelt werden, wie die Software-Timer.

Softwaretimer

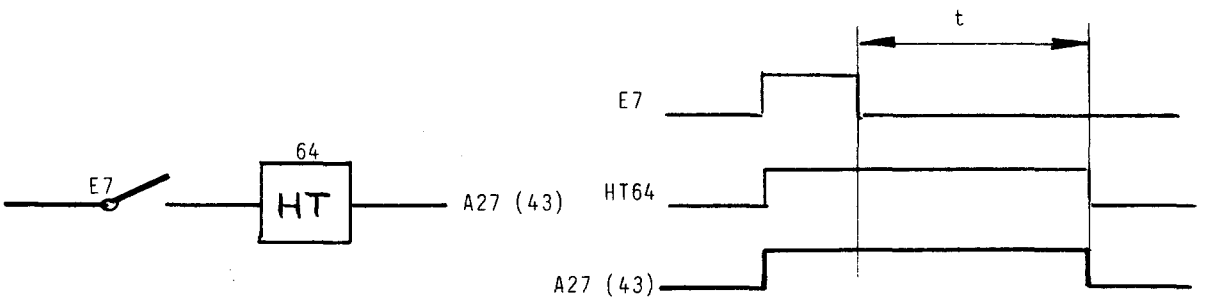
STR	256
00	t

Hardwaretimer

REO	64
SEO	64

Beim Hardwaretimer werden Zeitbereich und Zeit an dem Hardwaremodul PCA0.H20 direkt eingestellt.

Beispiel entsprechend A6:




```
  ADDR  NC  MNC  OPRD
  80    01  STH     7   Abfrage E7
  81    12  REO    64 } Hardwaretimer HT 64 starten
  82    11  SEO    64 }
  ------------------
  83    01  STH    64   Abfrage HT 64
  84    10  OUT    27   Uebertragung auf A27
  85    20  JMP    80
```

Gleich wie beim Softwaretimer wird auch bei HT 64 der Zeitablauf sofort gestartet. Solange E7 jedoch eingeschaltet ist, wird HT 64 bei jedem Programmdurchlauf wieder rückgesetzt. Der eigentliche Zeitablauf beginnt erst beim Oeffnen von E7, wodurch obige Ausschaltverzögerung entsteht.

(A8) Softwaretimer einschaltwischend mit externer Zeiteingabe im BCD-Code

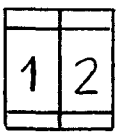
Die PCAØ erlaubt auch das direkte Einlesen von BCD-Werten in die Register der Timer und der Counter. Damit können Zeitwerte extern über BCD-Schalter jederzeit verändert werden. Wenn wir gemäss Kapitel 5 das Eingangssimuliergerät PCA2.SØ5 verwenden, so sehen wir, dass dort die Eingangsadressen E8 ... E15 von einem zweistelligen BCD-Schalter bedient werden. Ein BCD-Schalter gibt dabei die folgenden Signale an die Eingänge ab:

Binärsignale an 4 Eingängen (BCD-Schalter)				
E 12	E 13	E 14	E (15)	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

Der STR-Befehl kann nun so erweitert werden, dass die Verzögerungszeit direkt ab dem BCD-Schalter gelesen wird.

Code	OPERAND	
STR (14)	256	1. Zeile
NC	(En)	2. Zeile

BCD-Schalter (2 Stk.)



E8 ... 15
(En)

(En): Im Operand wird die höchste Eingangsadresse von 2 BCD-Schaltern eingesetzt (z.B. (15))

NC: Der numerische Code hat die Werte 16, 17 oder 18 mit den folgenden Bedeutungen:

16 Zeitwert Ø...99 x Zeittakt = Ø...9,9s *
oder 17 Zeitwert Ø...990 x Zeittakt = Ø..99s *
oder 18 Zeitwert Ø...9900 x Zeittakt = Ø..99Øs *

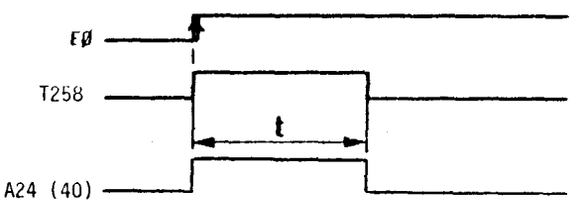
*) Diese Zeiten entstehen beim Standard-Zeittakt von 1/10 sec. Durch Aushängen der Brücke "1/10" auf 1/100 sec. veränderbar.

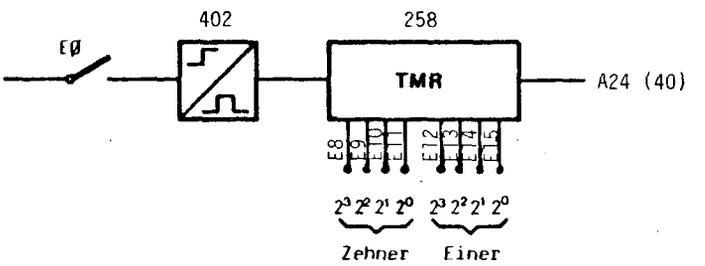
saia®pc

Aufgabe:

Der Zeitbereich von 1 bis 99 sec. soll extern mit einem BCD-Schalter einstellbar sein. Die Eingänge E8 ... 15 werden vom BCD-Schalterpaar benützt. Die Zeitfunktion soll einschaltwischend sein.

Durch den Befehl **DYN** wird nur die Einschaltflanke von EØ berücksichtigt, wodurch der Timer 258 frei ablaufen kann.





Programm:

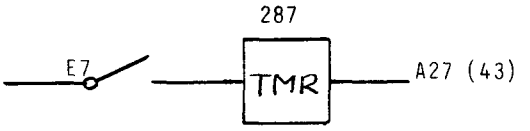
```
ADDR  NC  MNC  OPRD
 90   01  STH     0   Abfrage EØ
 91   09  DYN   402   Flankentriggerung, Timer wird nur im 1. Zykl. gesetzt
 92   14  STR   258   Timer starten und setzen mit
 93   17   17    15   Externwert x 10 x 1/10 sec. ab Eingängen 8 ... 15
 94   01  STH   258   Abfrage Zeitglied
 95   10  OUT    24   Uebertragung auf A24
 96   20  JMP    90
```

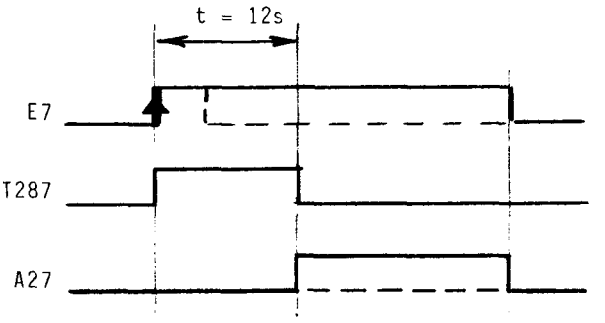
Soll die gleiche Funktion durch einen Hardwaretimer ausgeführt werden, so müssen lediglich die Zeilen 92 und 93 wie folgt ersetzt werden:

```
ADDR  NC  MNC  OPRD
 92   12  REO    66  } Starten von Hardwaretimer HT 66
 93   11  SEO    66  }
 94   01  STH    66   Abfrage Hardwaretimer HT 66
```

(A9) Einschaltverzögerung mit zwei neuen, sehr nützlichen Befehlen

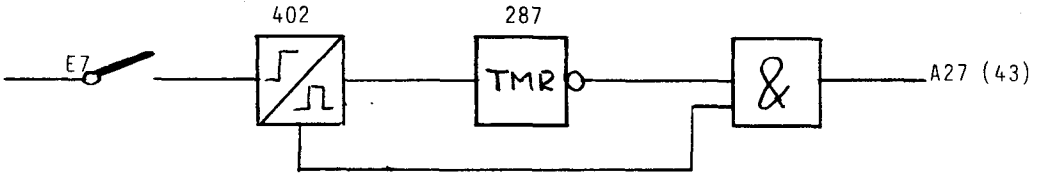
Aufgabe:





Bei der Einschaltverzögerung wird der Timer ebenfalls mit der Einschaltflanke von E7 gestartet wie beim Einschaltwischer. Hingegen soll der Ausgang 27 erst nach Ablauf der Zeit eingeschaltet werden.

Für die Aktivierung von A27 muss somit die Einschaltflanke von E7 erfolgt sein und der Timer abgelaufen (L) sein. Dies kann durch folgenden Logikplan dargestellt werden:



Da wir mit einer intelligenten SPS arbeiten, soll der Zeitablauf auch noch auf dem Programmiergerät sichtbar gemacht werden. Dies erlaubt uns der Befehl **DTC (31): "Display Timer or Counter"**. Sofern dieser Befehl mindestens alle Sekunden einmal abgearbeitet wird (was bei Umlaufprogrammen kein Problem ist), so wird im Operandfeld des P05 der aktuelle Inhalt des entsprechenden Timers oder Counters angezeigt. Bedingung für die Aktivierung des DTC ist, dass ACCU = 1 ist. Deshalb stellen wir den Befehl **SEA (19) 0: "Set Accumulator"** davor.

Programm:

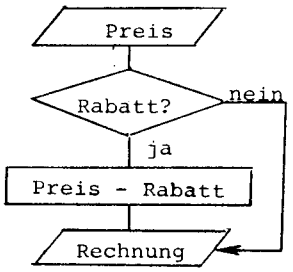
```
  ADDR  NC  MNC  OPRD
 ┌►100  01  STH     7  Abfrage E7
 │ 101  09  DYN   402  Flankentriggerung
 │ 102  14  STR   287  Timer starten und setzen mit
 │ 103  ØØ  ØØ    120  120 x 0,1 sec = 12 sec
 │ ------------------
 │ 104  01  STH   402  Abfrage des Flanken-Merkers von E7
 │ 105  04  ANL   287  UND Timer abgelaufen (L)
 │ 106  10  OUT    27  dann Ausgabe auf A27
 │ ------------------
 │ 107  19  SEA     Ø  ACCU = 1 setzen
 │ 108  31  DTC   287  Anzeige des Timerinhaltes
 └─109  20  JMP   100
```

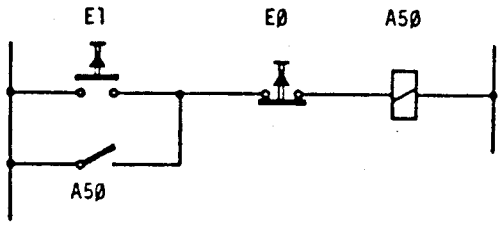
Soll diese Funktion durch den Hardwaretimer HT 67 ausgeführt werden, so müssen die Zeilen 102/103 und 105 wie folgt ersetzt werden:

```
  ADDR  NC  MNC  OPRD
   102  12  REO    67  Starten von Hardwaretimer HT 67
   103  11  SEO    67
    :                :
   105  04  ANL    67  UND HT 67 abgelaufen (L)
```

(B) Programmieren nach Flussdiagramm

Zeigen Sie einem 12-jährigen Kind einen Kontaktplan und daneben dieses Flussdiagramm. Was glauben Sie kann das Kind interpretieren ?

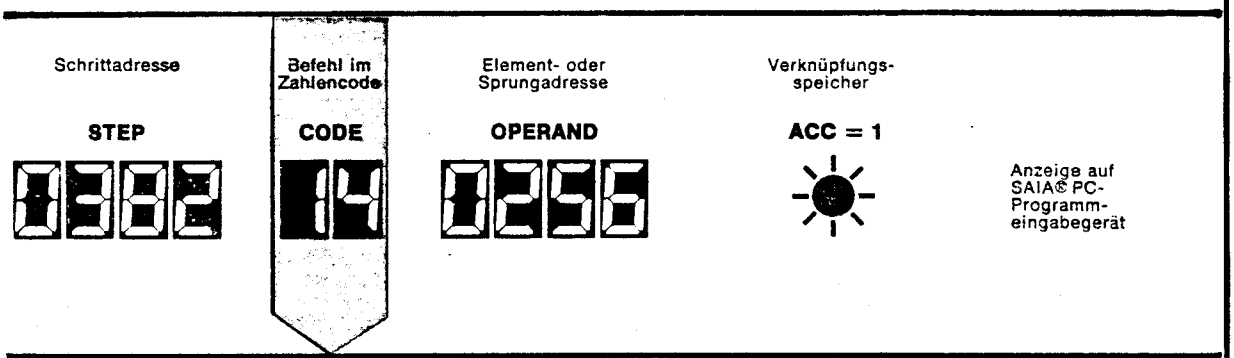




Ist es da ein Wunder, dass auch viele industrielle Prozesse je länger desto lieber in Flussdiagramm-Darstellung beschrieben werden. Gerade in der Chemie oder der Lebensmittelverarbeitung, aber auch in vielen Sektoren der Maschinenindustrie gibt es viele Abläufe, welche sich auf einfache und verständliche Weise mit dem Flussdiagramm beschreiben lassen.

In der PCAØ stecken eine Anzahl Befehle, welche zur Programmierung nach Flussdiagramm wirksam eingesetzt werden können, wodurch die Funktionen wesentlich einfacher und transparenter werden.

Zähler-Befehle können selbstverständlich sowohl in der Programmierung nach Kontaktplan wie auch nach Flussdiagramm eingesetzt werden. Um aber die Stufe (A) nicht zu hoch werden zu lassen, haben wir sie der Stufe (B) zugeordnet.

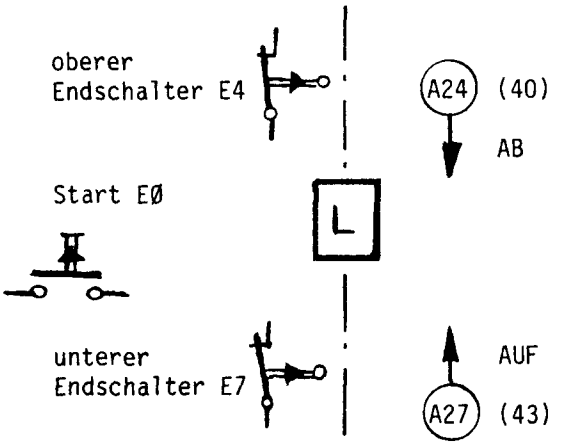


	Zahlen-code	Mnemo-code	Befehl englisch	Beschreibung
Zählbefehle	15*	SCR*	Set Counter	Setze Zähler auf vorgewählten Wert
	17 18	INC DEC	Increment Counter Decrement Counter	Erhöhe / Erniedrige den Stand des Zählers um 1
Sprungbefehle	2Ø	JMP	Unconditional Jump	Unbedingter Sprung auf Schrittadresse
	21 22	JIO JIZ	Jump if Accu is One Jump if Accu is Zero	Springe, wenn Accu = 1 / Accu = Ø auf Schrittadresse
	23 24	JMS RET	Jump to Subroutine Return from Subrout.	Springe in Unterprogramm Rücksprung vom Unterprogramm
Wartebefehle	25 26	WIH WIL	Wait if High Wait if Low	Warte, solange Element High / Low
Hilfsbefehle	ØØ	NOP	No Operation	Keine Operation
	19	SEA	Set Accu	Setze Accu = 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)

saia®pc

(B1) Auf/Ab-Bewegung



Aufgabe:

Durch einen Impuls auf EØ soll die Last L nach oben bewegt werden (AUF = A27) bis E4 öffnet. Anschliessend soll die Last wieder gesenkt werden (AB = A24) bis E7 öffnet.

Randbedingungen:

- Bei Dauerimpuls auf EØ soll sich diese Sequenz dauernd wiederholen.
- Nach Spannungsunterbruch soll erst nach Impuls auf EØ eine Aufwärtsbewegung ausgelöst werden.
- Hinweis: Vor dem Start des Programmes müssen die Schalter E4 und E7 geschlossen sein.

Lösung nach Flussdiagramm:

Flussdiagramm	Programm	Kommentar
Warten, solange EØ (offen / zu)	110 WIL (26) Ø (L / H)	Warten, solange EØ offen (Low) ist. Wenn EØ schliesst, dann ....
AUF↑ (A27)	111 SEO (11) 27	Ausgang 27 setzen (AUF)
Warten, solange E4 (zu / offen)	112 WIH (25) 4 (H / L)	Warten, solange E4 zu (High) ist. Wenn E4 öffnet, dann...
STOP! (A27) AB↓ (A24)	113 REO (12) 27 114 SEO (11) 24	A27 rücksetzen (STOP) und A24 setzen (AB).
Warten, solange E7 (zu / offen)	115 WIH (25) 7 (H / L)	Warten, solange E7 zu (High) ist. Wenn E7 öffnet, dann...
STOP! (A24)	116 REO (12) 24	A24 rücksetzen (STOP)
Rücksprung	117 JMP (20) 110	Rücksprung zum Anfang

Wie Sie leicht aus diesem Beispiel ersehen können, wird beim Programmieren nach Flussdiagramm der Ablauf einer Sequenz Schritt für Schritt "erzählt". Die Phasen des Ablaufes werden gegliedert in Bedingungen (Warten auf einen Eingangs-Zustand) und Aktionen (Ausgänge setzen oder rücksetzen).

Wenn wir dieses Programm im Einzelschrittbetrieb in der Betriebsart STEP (Tastenfolge [S] [A] 110 [+] [+])..... verfolgen wollen, werden wir feststellen, dass der Prozessor in den Warteschleifen selber solange gefangen bleibt, bis die Bedingung zum Weitergehen erfüllt ist. Dies heisst: Das Programm läuft nicht dauernd zyklisch um wie beim Programm nach Kontaktplan, sondern das Programm läuft synchron zum Fortschritt des Prozesses ab.

Dies ergibt drei verschiedene Vorteile:

- Die Programme werden transparent, indem sie den Prozessablauf widergeben und nicht irgend eine Umsetzung in einen ablauffremden Kontaktplan.

- Es werden jeweils nur diejenigen Programmteile bearbeitet, welche im Prozessablauf gerade von Bedeutung sind. Dadurch entfallen ev. Fehlfunktionen durch die Bearbeitung unrelevanter Programmteile, vor allem aber steigt dadurch die Reaktionsgeschwindigkeit zwischen Schrittbedingung und Aktion ganz erheblich.

- Bleibt nach der ersten Auf/Ab-Bewegung der Schlitten L stehen, so kann mit dem Programmiergerät in der Betriebsart STEP [+] [+] ... eindeutig festgelegt werden, wo das Programm "hängen" geblieben ist. Es wird Schritt 110 mit WIL Ø angezeigt. Die Kontrolle der LED von EØ wird ebenfalls bestätigen, dass dieser Eingang nicht aktiviert ist, womit der Fehler rasch auf den Kontakt oder die Verbindungsleitung von EØ lokalisiert werden kann.

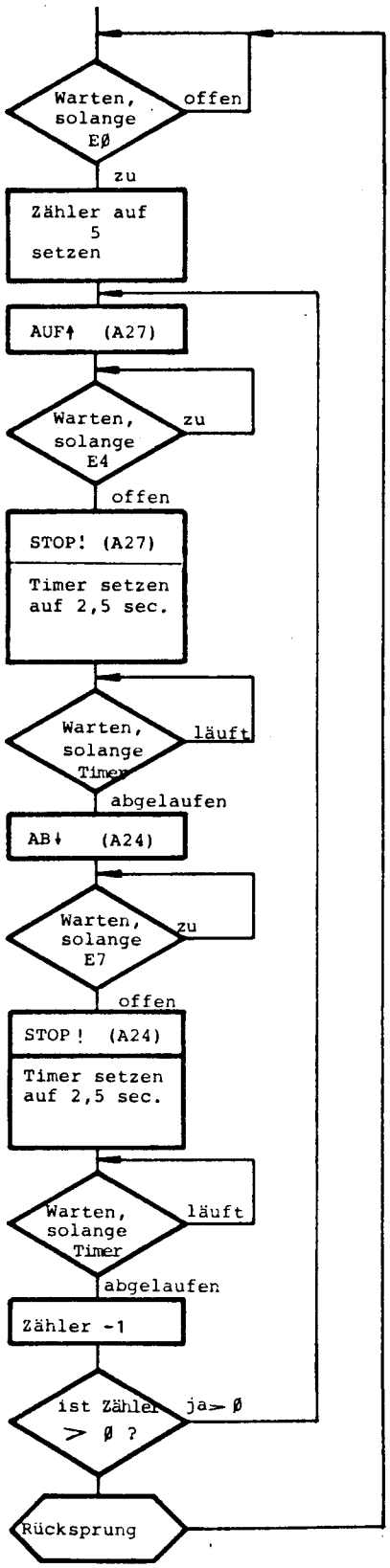
saia®pc

(B2) Auf/Ab-Bewegung mit Timer und Zähler

Aufgabe:
Die Bewegung gemäss Beispiel B1 soll nicht nur einmal, sondern eine gewisse Anzahl mal (z.B. 5 mal) durchlaufen werden, wobei in den Umkehrpunkten jeweils 2.5s gewartet werden soll.

Lösung:

Flussdiagramm



Programm

```
130   WIL (26) Ø      L
                      H
131   SCR (15) 300
132        ØØ    5
133   SEO (11)  27
134   WIH (25)  4     H
                      L
135   REO (12)  27
136   STR (14) 256
137        ØØ   25
138   WIH (25) 256    H
                      L
139   SEO (11)  24
140   WIH (25)  7     H
                      L
141   REO (12)  24
142   STR (14) 256
143        ØØ   25
144   WIH (25) 256    H
                      L
145   DEC (18)  300
146   STH (Ø1)  300
147   JIO (21) 133
148   JMP (20) 130
```

Kommentar

Warten, solange EØ offen (Low) ist. Wenn EØ schliesst, dann....

Zähler 300 setzen auf 5

Ausgang 27 setzen (AUF)

Warten, solange E4 zu (High) ist. Wenn E4 öffnet, dann....

A27 rücksetzen (STOP). Timer 256 setzen auf 2,5 sec.

Warten, solange Timer läuft (H ist).

Ausgang 24 setzen (AB).

Warten, solange E7 zu (High) ist. Wenn E7 öffnet, dann....

A24 rücksetzen (STOP). Timer 256 setzen auf 2,5 sec.

Warten, solange Timer läuft (H ist).

Zähler 300 um 1 dekrementieren.

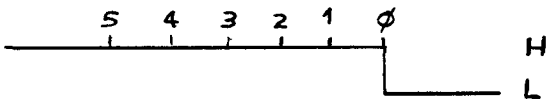
Zähler abfragen, ob > Ø (H ist).

Wenn ja, Sprung

Rücksprung zum Anfang.

Gegenüber der Aufgabe B1 wurden hier in den Aktionsteilen auch Timer gestartet, auf deren Ablauf dann in zusätzlichen Bedingungsschleifen auch gewartet werden muss (WIH 256).

Neu ist der Einsatz des Befehles **SCR (15): Setze Zähler**.
Dabei handelt es sich um einen zweizeiligen Befehl (wie beim Timer). Der Zähler wird in unserem Beispiel auf den Wert 5 gesetzt. Solange der Zähler C300 grösser als Null ist, so ist auch sein logischer Zustand "H".



Mit dem Befehl **DEC (18): Decrement Counter** vermindern wir den Zählerinhalt jedem Umgang um 1.

Ueber den Abfrage-Befehl STH 300 können wir nun laufend den logischen Zustand des Zählers abfragen. Ist nach der Abfrage der ACCU = 1, so ist der Zählerinhalt noch grösser als Null, es muss also nochmals eine Schleife durchlaufen werden. Das Rückspringen an den Schleifenanfang erfolgt durch den bedingten Sprungbefehl **JIO (21): Springe, wenn ACCU = 1**.

Mit anderen Worten: ist der Zähler grösser als Null, so wird nochmals eine Schleife durchlaufen. Bei Zählerstand = Null wird nicht mehr zurückgesprungen, so dass der letzte Befehl JMP wieder zum Programm-Anfang zurückführen kann.

Soll anstelle des Softwaretimers T256 der Hardwaretimer HT 65 verwendet werden, so sind die entsprechenden Programmteile wie folgt zu ersetzen:

```
STR  256 }          { REO 65
 00   25 } ----->   { SEO 65
WIH  256 }          { WIH 65
```

(B3) Was tun wir, wenn wir parallel zur Auf/Ab-Bewegung noch andere Funktionen ablaufen lassen möchten ?

Dann eröffnen wir ein zweites oder drittes Parallelprogramm von den total 16 Parallelprogrammen, die in jeder PCA0-Grundausführung drin stecken !

Aufgabe:

Parallel und unabhängig vom Programm B2 sollen in einem Parallelprogramm ein Blinker von 0.3 sec. Blinkdauer aktiviert und der aktuelle Zählerstand der Auf/Ab-Bewegung auf dem Programmiergerät dauernd angezeigt werden.

Programm:

```
 120 | PAS (29)   (1) |} Dem Auf/Ab-Programm, beginnend mit der Schrittadresse
  21 |  00        130 |} 130 wird die Parallelprogramm-Nummer (1) zugeordnet
  22
┌→123   DTC (31)  300   Der Zählerstand des Zählers 300 soll    )
│                       dauernd angezeigt werden                )
│ 124   STH         2 )                                         )
│  25   ANL       257 )                                         > Parallel-
│  26   STR       257 > Ausgang 16 blinkt im Takt von 0,3 sec., ) programm (0)
│  27    00         3 ) sobald E2 geschlossen ist               )
│  28   COO        16 )                                         )
└─129   JMP       123
```

Mit dem zweizeiligen Befehl **PAS (29) 1 (Programm-Assignierung)** weisen wir ganz zu Anfang im sog. Assignierungsteil dem Parallel-Programm, beginnend mit der Schrittadresse 130 die Nr. (1) zu.

Weitere Parallel-Programme könnten ohne weiteres mit PAS (2) oder PAS (3) hinzugefügt werden.

Das Programm von Schrittadresse 123 bis 129 ist ein sog. Umlaufprogramm ohne Warteschleifen. In ihm könnten weitere Funktionen wie z.B. Ueberwachungen oder andere dauernd aktivierte Aufgaben aufgeführt werden. Dieses Parallel-Programm ohne Assignierung erhält automatisch die Nummer (0) .

Wenn Sie nun die PP 0 ab Schrittadresse 123 und PP1 ab Schrittadresse 130 parallel zueinander in Betrieb setzen wollen, dann springen Sie auf die Anfangsadresse 120 mit der Assignierung: [S] [A] 120 [+] [R] .

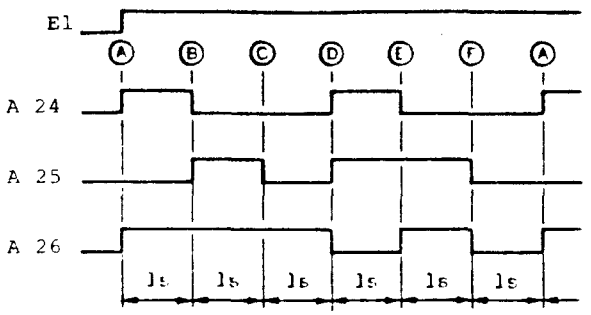
Probieren Sie jetzt beide Programme aus ! Beide laufen vollkommen unabhängig und asynchron zueinander ab. Wie oben erwähnt, könnten mit der PCA0 (wie mit allen SAIA-PC) bis zu 16 beliebig lange Programme parallel betrieben werden.

(B4) Mit Unterprogrammen sparen Sie viel Zeit und machen kürzere und überschaubare Programme

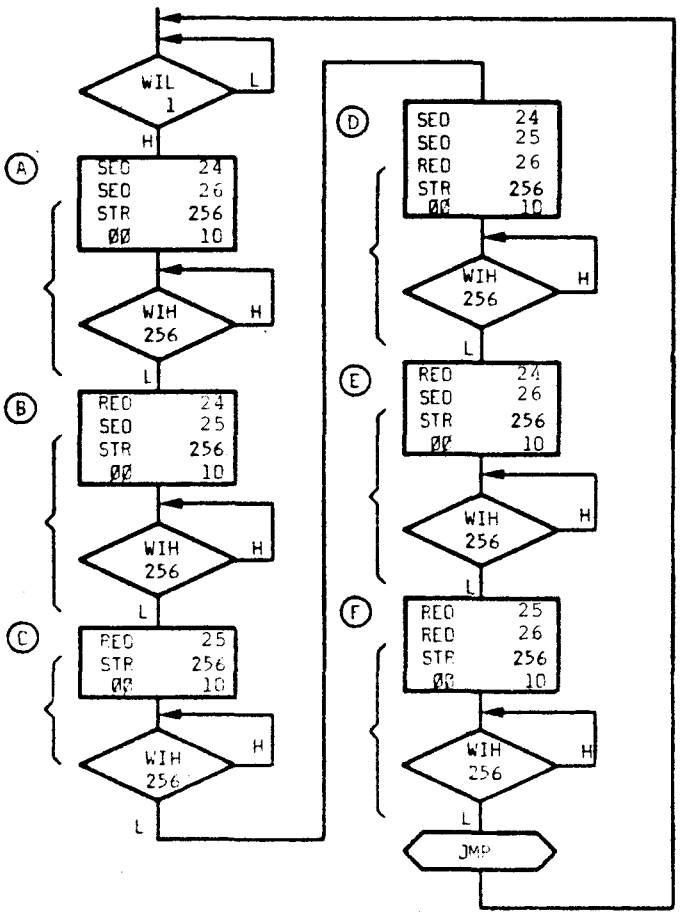
---

Aufgabe:

Die nachfolgende Sequenz soll nach Schliessen von E1 ablaufen:



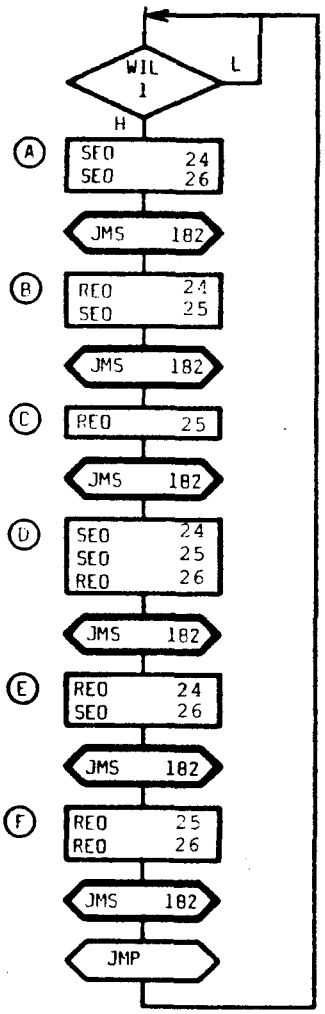
Lösung a: Selbstverständlich im Flussdiagramm (wer Lust hat, darf es auch mal im Kontaktplan versuchen !)



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.

Lösung b: Mit Unterprogramm

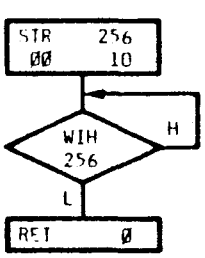
Hauptprogramm




```
********************* Haupt-Programm
ADDR   NC   MNC   OPRD
 160   26   WIL      1   Warte, wenn E1 offen
 161   11   SEO     24
 162   11   SEO     26
 163   23   JMS    182 ⇒ Springe zur Subr. 182
 164   12   REO     24
 165   11   SEO     25
 166   23   JMS    182 ⇒ Springe zur Subr. 182
 167   12   REO     25
 168   23   JMS    182 ⇒ Springe zur Subr. 182
 169   11   SEO     24
 170   11   SEO     25
 171   12   REO     26
 172   23   JMS    182 ⇒ Springe zur Subr. 182
 173   12   REO     24
 174   11   SEO     26
 175   23   JMS    182 ⇒ Springe zur Subr. 182
 176   12   REO     25
 177   12   REO     26
 178   23   JMS    182 ⇒ Springe zur Subr. 182
 179   20   JMP    160
```

Der Befehl **JMS (23): Jump to Subroutine** erlaubt es, die immer gleichen Programmteile nur einmal zu schreiben. Jede Subroutine endet mit dem Befehl **RET (23): Return** mit dem Operanden 0.

Subroutine "182"



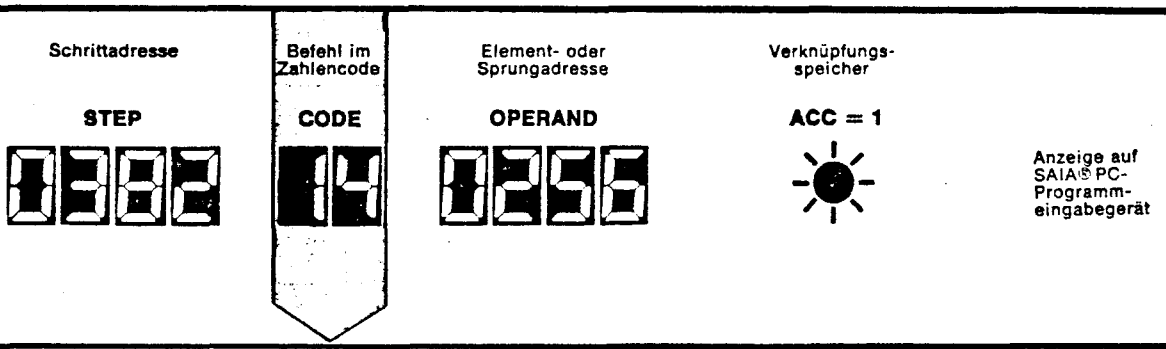

```
================= Subroutine 182(1 Sek. warten)
⇒182   14   STR   256
 183   ØØ   ØØ    10
 184   25   WIH   256
 185   24   RET     Ø
```

Es könnte auch die Subroutine von der Subroutine von der Subroutine, d.h. bis in die 3. Ebene programmiert werden.

© Indexierung, Arithmetik und Check Sum
Auf der Programmier-Stufe C gehören Sie schon zu den Profis

Man kann durchaus glücklich sein, ohne auf diese Stufe zu klettern. Einmal oben, werden Sie aber stolz auf sich und die PCAØ sein, dass Sie Ihre Programme so elegant und leistungsfähig gestaltet haben.

Dies sind die verbleibenden Befehle für das Software-Niveau (1H) der SAIA-PC-Systemfamilie:



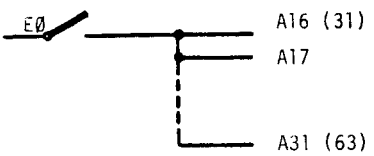
	Zahlen-code	Mnemo-code	Befehl englisch	Beschreibung
Transfer-Befehle	15	SCR	Set counter	
	19		2. Zeile	Einlesen 5 x 4 Bit BCD
	20			Ausgeben 5 x 4 Bit BCD
	21			Ausgeben 8 Bit binär
	22			Ausgeben 12 Bit binär
	23			Ausgeben 16 Bit binär
	24			Einlesen 8 Bit binär
	25			Einlesen 12 Bit binär
	26			Einlesen 16 Bit binär
	31			Transfer Counter-Counter (IR)
Arithmetik-Befehle	15	SCR	Set counter	
	27		2. Zeile	addiere +
	28			subtrahiere -
	29			multipliziere x
	30			dividiere :
Indexier-Befehle	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
			OPERAND	
Spezial- Befehle	29	PAS	18	Veränderung der Anzahl der aktiven Parallelprogramme
Diese Befehle sind zweizeilig	29	PAS	30...34	Check-Sum

saia®pc

(C1) Mit Adressindexierung lässt sich mit kurzen Programmen grosse Wirkung erzielen

Aufgabe:

Alle Ausgänge der PCA0 sollen beim Schliessen von Eingang E0 aktiviert werden.



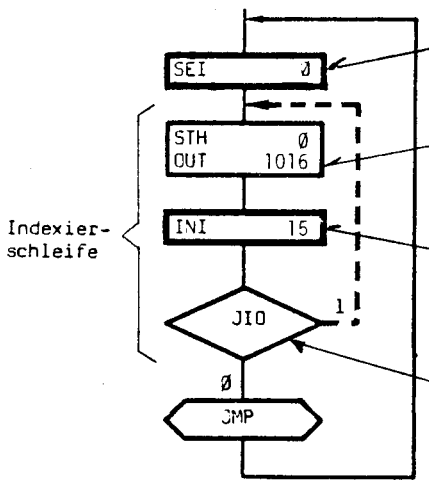
Bei einer PCA0 mit 32 Ausgängen würde dieses Programm ohne Adressindexierung 34 Schritte umfassen.

Lösung:

Mit Indexierung wird das Programm unabhängig von der Anzahl Ausgänge immer nur 6 Zeilen betragen.

```
  ADDR  NC  MNC  OPRD
  200   16  SEI     0  Setze Indexregister (IR) auf Anfangswert 0
  201   01  STH     0  Abfrage Eingang 0
  202   10  OUT  1016  Setzen der indexierten Ausgänge
  203   27  INI    15  Inkrementieren des IR bis Endwert 15
  204   21  JIO   201  Rücksprung, solange IR < 15
  205   20  JMP   200  -->
```

Aus dem nachstehenden Flussdiagramm sind die einzelnen Funktionen klarer erkennbar:



Mit dem Befehl **SEI (16)** wird das Indexregister auf Ø gesetzt (max. Wert 255)

Zur ersten Ausgangsadresse 16 wird 1000 dazugezählt. Damit wird dieser Befehl von der CPU indexiert behandelt wie folgt:
1016 - 1000 + IR

Durch den Befehl **INI (27)** wird das Indexregister bei jedem Durchlauf um 1 erhöht bis es den Wert 15 erreicht hat.

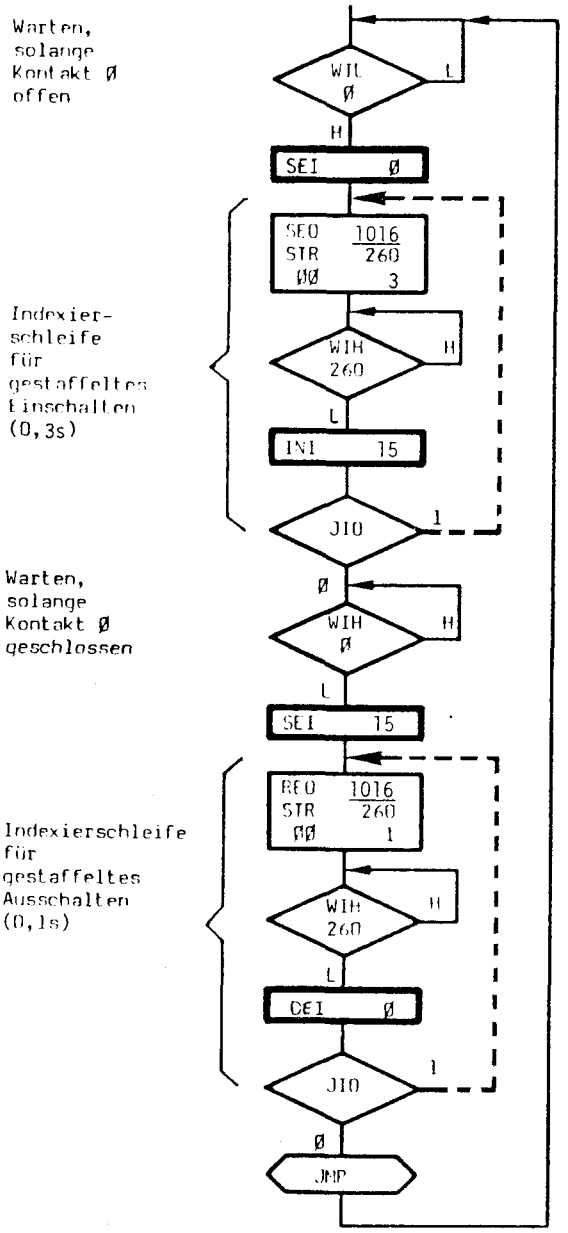
Solange IR < 15, ist der ACCU = 1, wodurch der Rücksprung erfolgt. Ist IR = 15, so wird die Indexierschleife verlassen. Alle Ausgänge von 16 ... 31 sind behandelt worden.

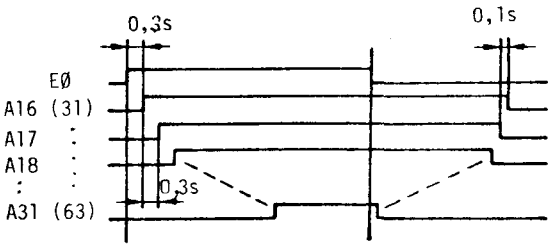
(C2) Durch Vorwärts- und Rückwärts-Indexierung erhält man grosse Flexibilität

Aufgabe:

Durch Schliessen von EØ sollen im Takt von Ø.3 sec. die Ausgänge A16 ... 31 (63) nacheinander einschalten.
Durch Oeffnen von EØ sollen die Ausgänge A31 (63) ... 16 in umgekehrter Reihenfolge im Takt von 0.1 sec. wieder ausschalten.

Lösung:






```
ADDR   NC   MNC   OPRD
-------------------        EINSCHALTPHASE
 210   26   WIL     Ø
  11   16   SEI     Ø
 212   11   SEO  1016   A16 + 1000 = 1016
  13   14   STR   260
  14        ØØ     3
  15   25   WIH   260
  16   27   INI    15   A31 - 16 = 15
 217   21   JIO   212
-------------------        AUSSCHALTPHASE
 218   25   WIH     Ø
  19   16   SEI    15   A31 - 16 = 15
 220   12   REO  1016   A16 + 1000 = 1016
  21   14   STR   260
  22        ØØ     1
  23   25   WIH   260
  24   28   DEI     Ø
  25   21   JIO   220
 226   20   JMP   210   -->
```

Die obere Indexierschleife läuft ähnlich ab wie in Beispiel C1. Bei der unteren Schleife wird das Indexregister vorerst auf 15 gesetzt. Beim ersten Durchgang wird REO 1016 - 1000 + 15 = REO 31 auf Ausgang 31 angewendet. Der Befehl **DEI (28)** vermindert den Wert des IR um 1, so dass beim folgenden Durchgang der Ausgang 30 ausgeschaltet wird usw. Erreicht das IR = Ø, so erfolgt kein Rücksprung mehr, die Indexierschleife wird verlassen.

Bei der Ausführung mit Relais-Ausgängen werden Sie bemerkt haben, dass jeweils nach 4 Relais eine Pause eintritt. Dies rührt daher, dass jeweils auch die 4 nicht bestückten Adressen 20 ... 23 etc. behandelt werden.

saia®pc

(C3) Auch rechnen können Sie mit der kleinen PCAØ

Die 64 Zählerregister der PCAØ können äusserst vielseitig eingesetzt werden. In den Befehlen **STR** und **SCR** haben wir bisher nur die Codes ØØ und 16, 17, 18 der zweiten Zeile kennengelernt. Wie aus der nachfolgenden Tabelle entnommen werden kann, stehen insgesamt 32 Funktionen zur Verfügung.

Mit den Codes Ø1 ... 15 können die Bereiche von 2048 bis 65535 erreicht werden.

Mit den Codes 19 ... 26 können 8 ... 20 bitige digitale Werte ins Zählerregister geladen oder auf Elemente ausgegeben werden.

Mit den Codes 27 ... 30 schliesslich können arithmetische Funktionen ausgeführt werden.

Code 31 dient dem Uebertragen von Werten von Indexregister in ein Zählerregister oder von einem Zählerregister ins andere.

	Mnemo-Code	Zahlen-Code	Operand	Erläuterungen
1. Zeile	STR / SCR	14 / 15	256...319	Adresse des Registers
2. Zeile		ØØ	xxxx	Wert im Operand + Ø
		Ø1		+ 2Ø48
		Ø2		+ 4Ø96
		Ø3		+ 6144
		Ø4		+ 8192
		Ø5		+ 1Ø24Ø
		Ø6		+ 12288
		Ø7		+ 14336
		Ø8		+ 16384
		Ø9		+ 18432
		1Ø		+ 2Ø48Ø
		11		+ 22528
		12		+ 24576
		13		+ 26624
		14		+ 28672
		15	xxxx	Wert im Operand + 3Ø72Ø
		16	höchste Adresse von 8 aufeinanderfolgenden Elementen	2 x 4 Bit BCD mal 1
		17		2 x 4 Bit BCD mal 1Ø
		18		2 x 4 Bit BCD mal 1ØØ
		19	höchste Adresse der Elementenreihe	Einlesebefehl für 5 x 4 Bit BCD
		2Ø		Ausgabebefehl für 5 x 4 Bit BCD
		21		Ausgabebefehl für 8 Bit binär
		22		Ausgabebefehl für 12 Bit binär
		23		Ausgabebefehl für 16 Bit binär
		24		Einlesebefehl für 8 Bit binär
		25		Einlesebefehl für 12 Bit binär
		26		Einlesebefehl für 16 Bit binär
		27	xxx	Addition (mit einer Konstanten (Ø...255) oder mit dem Inhalt eines T/C (256...319)*)
		28	xxx	Subtraktion
		29	xxx	Multiplikation
		3Ø	xxx	Division
		31	iii	iii = Ø: Zähler wird mit dem Wert des Indexregisters geladen iii = 256...319*: Zähler wird mit dem Wert des entsprechenden T/C geladen.

C 1 Bitprozessor-Befehle kompatibel mit PCA2.M30

STR/SCR

Die Zeit- bzw. Zählregister wurden von 15 auf 16 Bit erweitert; was der Erweiterung in Dezimalzahlen von 32'767 auf 65'535 entspricht. Die Codes der 2. Zeile dieser Befehle waren bisher nur von 00 bis 18 belegt.
Zusätzlich zu diesen Instruktionen wurden den restlichen Codes bis 31 folgende Funktionen zugewiesen:

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

CODE (2. Zeile)

19 ≙ 5 x 4 Bit BCD (5 Dekaden)
24 ≙ 8 Bit binär
25 ≙ 12 Bit binär
26 ≙ 16 Bit binär

Beispiel:

STR	256
24	431

8 Bit, binär — 8 Merker 424...431

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

CODE

20 ≙ 5 x 4 Bit BCD (5 Dekaden)
21 ≙ 8 Bit binär
22 ≙ 12 Bit binär
23 ≙ 16 Bit binär

Beispiel:

SCR	300
21	175

8 Bit, binär — 8 Ausgänge 168...175

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

Falls iii = 0 : Der Zähler ccc wird mit dem Inhalt des Indexregisters (des entsprechenden Parallelprogrammes) geladen.
Falls iii = 256...287 : Der Zähler ccc wird mit dem Inhalt des adressierten T/C geladen.
Falls iii = 288...319 : Der Zähler ccc wird mit dem Inhalt des adressierten C geladen.

d) Arithmetische Operationen
Die Operationen werden zwischen zwei Zählerregistern 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 oder STR → SCR (15) ccc ← Adresse des 1. T/C
CODE xxx ← Adresse des 2. T/C oder Wert einer Konstanten ≤ 255

CODE der 2. Programmzeile (27...30)

Hinweis: Alle Elementadressen können indexiert behandelt werden.

CODES

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:

a) Vor der Addition : C256 = 3Ø, C26Ø = 54
Operation :
```
SCR (15) 256
     27  26Ø
```
Nachher : C256 = 84, C26Ø = 54, ACCU = 1

b) Vor der Subtraktion : C258 = 124, C274 = 146
Operation :
```
SCR (15) 258
     28  274
```
Nachher : C258 = 65'514, C274 = 146, ACCU = Ø

124 - 146 = -22
⟶ 65'536 - 22 = 65'514

c) Vor der Multiplikation: : C26Ø = 12, C282 = 6
Operation :
```
SCR (15) 26Ø
     29  282
```
Nachher : C26Ø = 72, C282 = 6, ACCU = 1

d) Vor der Division : C31Ø = 1942, Konstante = 23
Operation :
```
SCR (15) 31Ø
     3Ø   23
```
Nachher : C31Ø = 84, ACCU = 1, Der Rest (1Ø) wird nicht gespeichert.

C 1 Bitprozessor-Befehle kompatibel mit PCA2.M30

STR/SCR

Die Zeit- bzw. Zählregister wurden von 15 auf 16 Bit erweitert; was der Erweiterung in Dezimalzahlen von 32'767 auf 65'535 entspricht. Die Codes der 2. Zeile dieser Befehle waren bisher nur von 00 bis 18 belegt.
Zusätzlich zu diesen Instruktionen wurden den restlichen Codes bis 31 folgende Funktionen zugewiesen:

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

CODE (2. Zeile)

19 ≙ 5 x 4 Bit BCD (5 Dekaden)
24 ≙ 8 Bit binär
25 ≙ 12 Bit binär
26 ≙ 16 Bit binär

Beispiel:

STR	256
24	431

8 Bit, binär — 8 Merker 424...431

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

CODE

20 ≙ 5 x 4 Bit BCD (5 Dekaden)
21 ≙ 8 Bit binär
22 ≙ 12 Bit binär
23 ≙ 16 Bit binär

Beispiel:

SCR	300
21	175

8 Bit, binär — 8 Ausgänge 168...175

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

Falls iii = 0 : Der Zähler ccc wird mit dem Inhalt des Indexregisters (des entsprechenden Parallelprogrammes) geladen.
Falls iii = 256...287 : Der Zähler ccc wird mit dem Inhalt des adressierten T/C geladen.
Falls iii = 288...319 : Der Zähler ccc wird mit dem Inhalt des adressierten C geladen.

d) Arithmetische Operationen
Die Operationen werden zwischen zwei Zählerregistern 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 oder STR →

SCR	(15)	ccc ← Adresse des 1. T/C
CODE		xxx ← Adresse des 2. T/C oder Wert einer Konstanten ≤ 255

CODE der 2. Programmzeile (27...30)

Hinweis: Alle Elementadressen können indexiert behandelt werden.

CODES

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:

a) Vor der Addition : C256 = 3Ø, C26Ø = 54
Operation :
```
SCR (15) 256
     27  26Ø
```
Nachher : C256 = 84, C26Ø = 54, ACCU = 1

b) Vor der Subtraktion : C258 = 124, C274 = 146
Operation :
```
SCR (15) 258
     28  274
```
Nachher : C258 = 65'514, C274 = 146, ACCU = Ø

124 - 146 = -22
⟶ 65'536 - 22 = 65'514

c) Vor der Multiplikation: : C26Ø = 12, C282 = 6
Operation :
```
SCR (15) 26Ø
     29  282
```
Nachher : C26Ø = 72, C282 = 6, ACCU = 1

d) Vor der Division : C31Ø = 1942, Konstante = 23
Operation :
```
SCR (15) 31Ø
     3Ø   23
```
Nachher : C31Ø = 84, ACCU = 1, Der Rest (1Ø) wird nicht gespeichert.

Aufgabe:

Um die Arithmetik-Möglichkeiten mit der PCAØ zu zeigen, wollen wir in diesem Beispiel etwas mit Zahlen spielen. Die Behandlung von Zahlenwerten spielt eine wichtige Rolle bei Zählaufgaben oder wenn analoge Werte verarbeitet werden sollen.

Den Eingängen an unserem Simuliergerät werden die folgenden Funktionen zugeordnet:

EØ	: Addition	+
E1	: Subtraktion	-
E2	: Multiplikation	x
E3	: Division	:

E7	: Abspeichern des 1. Wertes ab BCD-Vorwahlschalter
E6	: Abspeichern des 2. Wertes ab BCD-Vorwahlschalter
E5	: Auslösen der arithmetischen Operation

E8 ... 15: BCD-Vorwahlschalter

C260	: Register für erste Zahl
C270	: Register für zweite Zahl
C266	: Display-Register für DTC

A24 (56)	: Quittung für Abspeicherung 1. Wert (E7)
A25 (57)	: Quittung für Abspeicherung 2. Wert (E6)
A26 (58)	: Quittung für Ausführung der Operation

Die einzeln mit E7 und E6 abgespeicherten Werte sollen im Operand-Display angezeigt werden. Ebenso das Resultat der durch EØ ... E3 vorgewählten arithmetischen Operation.

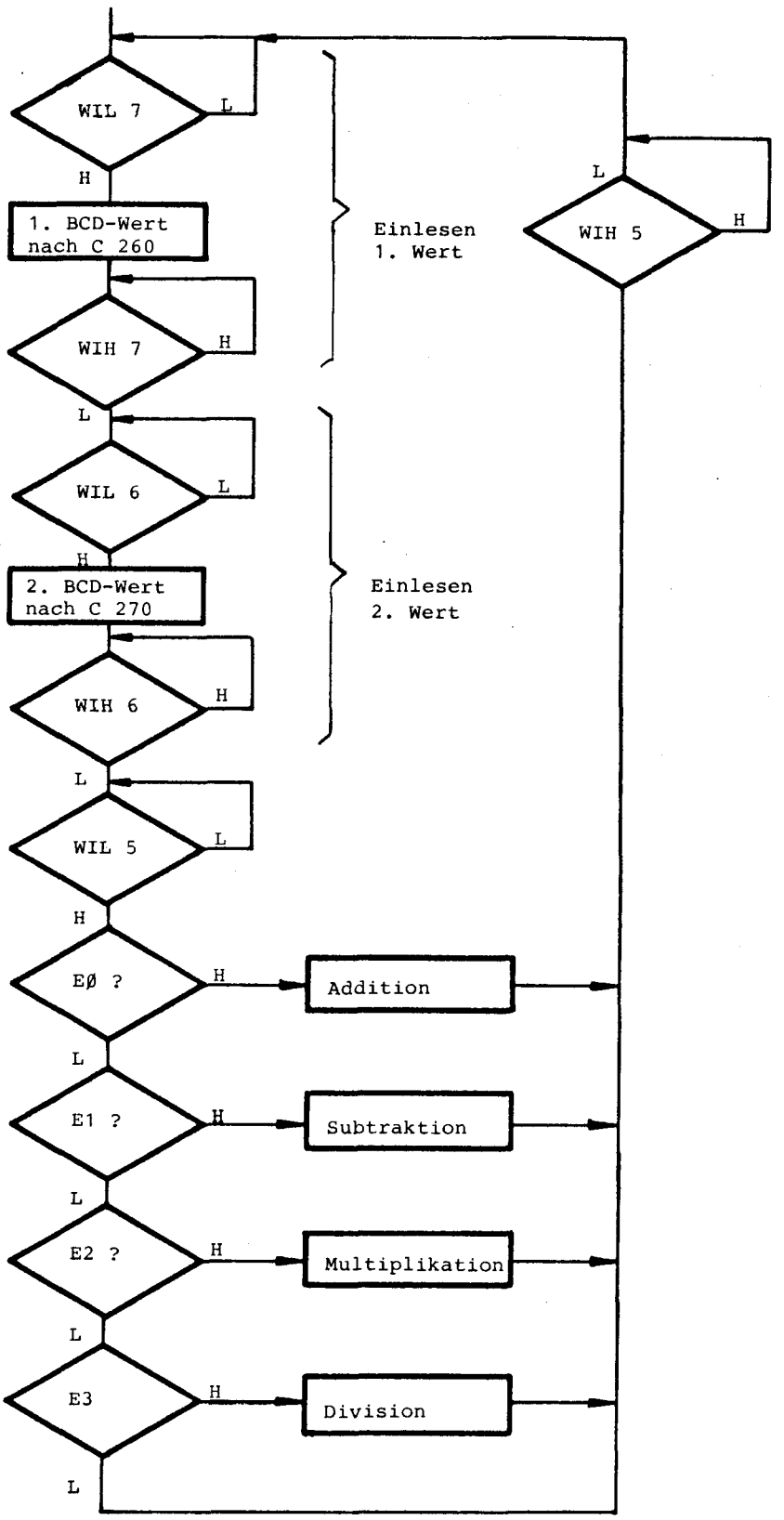
Bedienungsbeispiel: 87 - 25 = 62

- E1 einschalten --> Subtraktion
- 87 am BCD-Schalter einstellen
- mit E7 ein und wieder aus wird die 1. Zahl in Zähler C260 eingelesen
- 87 erscheint im Operand-Display, gleichzeitig leuchtet LED 24 als Quittung für den ersten Schritt
- 25 am BCD-Schalter einstellen
- mit E6 ein und wieder aus wird die 2. Zahl in Zähler C270 eingelesen
- 25 erscheint im Operand-Display, gleichzeitig leuchtet auch LED 25 als Quittung für den 2. Schritt
- mit E5 ein: Operation wird ausgeführt
- 62 erscheint als Resultat im Operand-Display, gleichzeitig leuchtet auch LED 26 als Quittung für 3. Schritt
- mit E5 aus werden die Ausgänge 24, 25, 26 zurückgesetzt, mit E7 kann eine neue Eingabe erfolgen
- Würde bei der Subtraktion ein negatives Resultat entstehen (2. Zahl > als 1. Zahl), so erscheint auf dem Display 9999
- Wird in einer Division durch Ø geteilt, so erscheint 8888. In beiden Fällen wird vom Umstand profitiert, dass die CPU in diesen Sonderfällen den ACCU = Ø setzt.

Lösung:

Wir arbeiten mit dem Flussdiagramm, das uns erlaubt, Schritt für Schritt den Vorgang zu verfolgen.

Grobflussdiagramm



Programm:

Um ein permanentes Display der Zahlenwerte zu haben, behandeln wir DTC 266 im parallelen Umlaufprogramm PP 0, während das Flussdiagramm mit PP 1 ab Adresse 238 assigniert wird.

```
ADDR  NC  MNC  OPRD
 230  29  PAS     1 ) Assignierung PP 1 ab
 231      00   238 ) Adresse 238
------------------
 232  00  NOP     0
 233  00  NOP     0
==================
 234  31  DTC   266 ) PP 0 Display C266
 235  20  JMP   234 )
==================
 238  26  WIL     7   PP1                                  )
  39  11  SEO    24   Quittung für 1. Schritt             ) abspeichern
  40  15  SCR   260   BCD-Wert auf C260                   ) der 1. Zahl
  41      16    15                                        ) (2-stellig)
  42  15  SCR   266   C260 kopieren auf C266              )
  43      31   260   für Display                          )
  44  25  WIH     7                                       )
------------------
 246  26  WIL     6                                       )
  47  11  SEO    25   Quittung für 2. Schritt             )
  48  15  SCR   270   BCD-Wert auf C270                   ) abspeichern
  49      16    15                                        ) der 2. Zahl
  50  15  SCR   266   C270 kopieren auf C266              ) (2-stellig)
  51      31   270   für Display                          )
  52  25  WIH     6                                       )
------------------
 254  26  WIL     5                                       )
  55  11  SEO    26   Quittung für 3. Schritt             )
  56  01  STH     0                                       )
  57  21  JIO   265 --> Sprung zur Addition              )
------------------                                       )
  58  01  STH     1                                       )
  59  21  JIO   270 --> Sprung zur Subtraktion           ) Operation
------------------                                       ) ausführen
  60  01  STH     2                                       )
  61  21  JIO   280 --> Sprung zur Multiplikation        )
------------------                                       )
  62  01  STH     3                                       )
  63  21  JIO   285 --> Sprung zur Division              )
------------------                                       )
 264  20  JMP   295                                       )
```

```
ADDR  NC  MNC  OPRD
→265  15  SCR   260   C260 + C270 --> C260          )
  66       27   270 +                               )
  67  15  SCR   266   C260 kopieren auf C266        ) Addition
  68       31   260   für Display                   )
  69  20  JMP   295                                 )
-------------------
→270  15  SCR   260   C260 - C270 --> C260          )
  71       28   270 -                               )
  72  22  JIZ   276   Sprung, falls Resultat neg.   )
  73  15  SCR   266   C260 kopieren auf C266        ) Subtraktion
  74       31   260   für Display                   )
  75  20  JMP   295                                 )
→276  15  SCR   266 } Display-Zähler mit            )
  77       ØØ  9999*} 9999 laden                    )
  78  20  JMP   295                                 )
-------------------
→280  15  SCR   260   C260 x C270 --> C260          )
  81       29   270 x                               )
  82  15  SCR   266   C260 kopieren auf C266        ) Multiplikation
  83       31   260   für Display                   )
  84  20  JMP   295                                 )
-------------------
→285  15  SCR   260   C260 : C270 --> C260          )
  86       30   270 :                               )
  87  22  JIZ   291   Sprung, falls Division durch Ø )
  88  15  SCR   266   C260 kopieren auf C266        ) Division
  89       31   260   für Display                   )
  90  20  JMP   295                                 )
→291  15  SCR   266 } Display-Zähler mit            )
  92       ØØ  8888*} 8888 laden                    )
-------------------
→295  25  WIH     5   Operations-Schritt beendet ?
  96  12  REO    24 }
  97  12  REO    25 } Rücksetzen der Quittungs-Ausgänge
  98  12  REO    26 }
 299  20  JMP   238  -----> Rücksprung zum Anfang von PP1
-------------------
```

Manuelles Abfragen oder Laden eines Zählerregisters:
Falls Sie zu irgend einem Zeitpunkt den Inhalt eines Zählerregisters manuel abfragen oder verändern möchten, dann gehen Sie wie folgt vor:

Z.B. C270 anzeigen

[M] Taste "MAN" 0,5 sec. betätigen

[A] 3270 (also 3000 mehr als Zähleradresse 270)

--> Zählerinhalt wird angezeigt

---

*) Da der Operand max. 2047 betragen kann, muss (gemäss Tabelle in Beispiel C3) wie folgt eingegeben werden:
statt ØØ 9999 --> Ø4 1807 (4 x 2048 + 1807)
statt ØØ 8888 --> Ø4 696 (4 x 2048 + 696)

Z.B. in C266 den Wert 1234 eingeben

[M] Taste "MAN" 0,5 sec. betätigen

[A] 3266 (Adresse + 3000)

[E] 0 1234 (bei Werten < 10'000 ist 0 voranzustellen)

[+]

Ist Schalter E7 geöffnet, so wird uns der eingegebene Wert in obigem Programm nun auch via DTC im RUN-Betrieb angezeigt.

Kaum zu glauben, was die kleine PCA0 alles kann !

saia®pc

(C4) Wenn einmal grosse Sprünge programmiert werden müssen

Ein Programm von 2047 Schritten ist für eine PCAØ schon ein grosses Programm. Der Anwenderspeicher hat jedoch eine Kapazität für Anwenderschritte von Ø...4095 Schritten.

Sollen grosse Sprünge mit Zieladressen in der zweiten Hälfte des Anwenderspeichers, d.h. ab 2048 bis 4095 programmiert werden, so sind diese Sprungbefehle zweizeilig auszuführen.

Dies gilt für alle Sprungbefehle JMP, JIO, JIZ und JMS.

a) Sprungbefehle mit Operanden 1 bis 2047

(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 2048 bis 4095

Beispiel: Sprung in die Subroutine 3280

Bei der Programmierung:

```
501  JMS (23)     Ø   E
502       ØØ  3280   E
```

Bei der Kontrolle:

```
501  JMS (23)     Ø   +
502       Ø1  1232   C   ≙ convert
502       EE  3280   +
```

Der Wert Ø1 1232, wie er im Anwenderspeicher steht, entspricht der Sprungadresse 3280. Ø1 steht für das Vielfache von 2048 und 1232 ist der Rest (1 x 2048 + 1232 = 3280).

Mit der Taste C wird die wirkliche Sprungadresse angezeigt, wobei im Code die Zeichen EE stehen (gilt für Programmiergerät PCA2.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 (20)  Ø
     ØØ   Ø
```

c) Ein praktisches Beispiel anhand eines Blinkers in einer Subroutine. Die Subroutine befindet sich auf der Adresse 3500.

Hauptprogramm

```
ADDR  NC  MNC  OPRD
500   26  WIL     1
501   23  JMS     Ø   } ⇒
502       ØØ  3500   }
503   20  JMP   500
```

Subroutine

```
ADDR  NC  MNC  OPRD
3500  Ø2  STL   256
3501  14  STR   256
3502      ØØ     2
3503  13  COO    24
3504  24  RET    Ø -->
```

(C5) Mit dem Befehl **PAS 18** werden die Parallelprogramme gestoppt, wenn sie nicht mehr benötigt werden

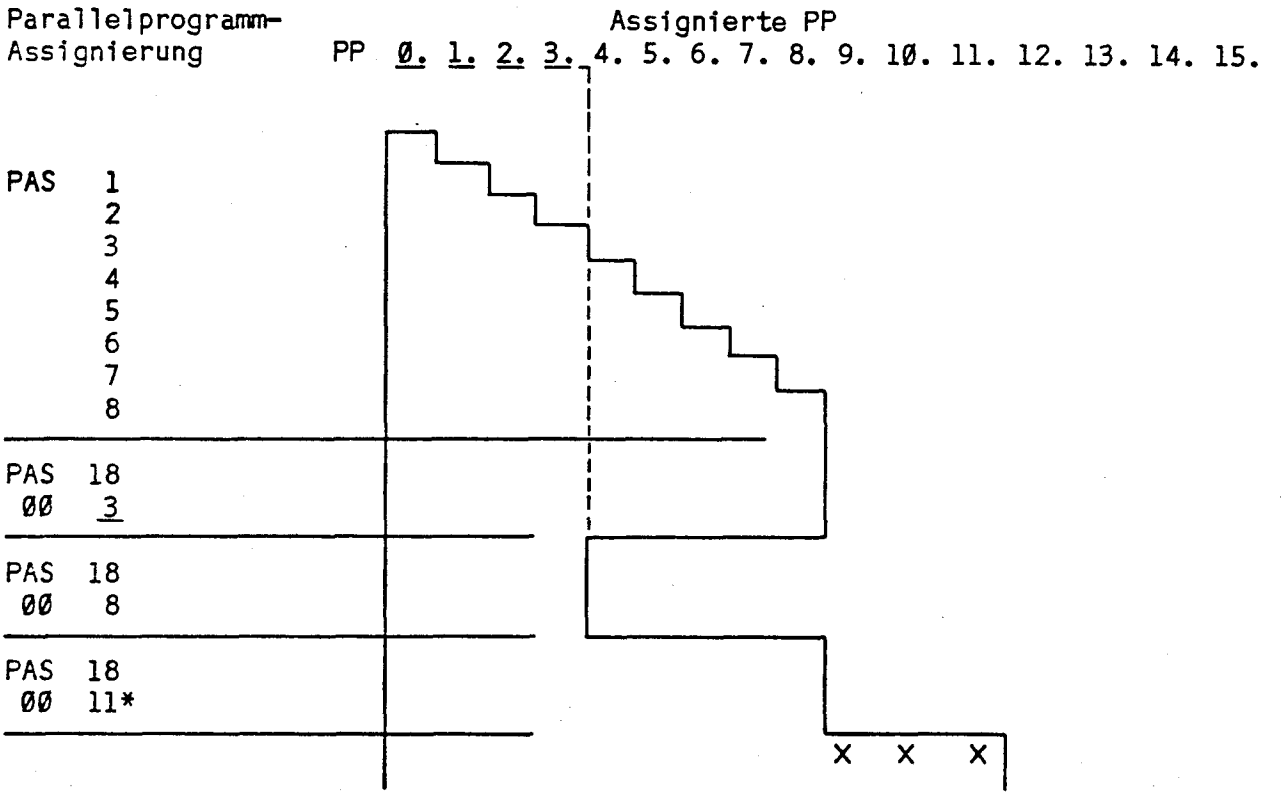
PAS 18 Begrenzung der assignierten Parallelprogramme

Alle SAIA-PC erlauben es, bis zu 16 Parallelprogramme zu assignieren und parallel ablaufen zu lassen. Wird ein PP nicht mehr 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 bei der PCAØ jetzt jedoch die Anzahl der aktiven 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

1. Zeile	PAS	18	
2. Zeile	ØØ	x	; x von Ø...15



*) 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.

saia®pc

(C6) Mit dem Befehl "Check-Sum" erhöhen Sie die Zuverlässigkeit Ihrer PCAØ noch mehr

PAS 3Ø und PAS 31 ... 34 "Check-Sum" des System- und Anwenderprogrammes

Die "Check-Sum"-Funktion dient zur Prüfsummenbildung der Speicherinhalte des Systemprogrammes (PAS 3Ø) oder der Anwender-Programme bzw. -Texte (PAS 31...34); es kann damit sichergestellt werden, dass in den überprüften Speichern keine Inhaltsveränderung stattgefunden hat.

Nach Ausführung des Befehls wird:

- ACCU = 1 wenn der Vergleich richtig ist
- ACCU = Ø wenn die Bezugsgrösse mit der Prüfsumme nicht übereinstimmt.

Die Befehle PAS 3Ø...34 werden unabhängig vom ACCU immer ausgeführt.

Bei einer Inhaltsveränderung der Speicher kann der Anwender die ihm dafür notwendig erscheinenden Massnahmen programmieren: Auslösung eines Alarmes, Rücksetzen des Watch-Dog usw..

```
PAS  3Ø        ; Prüfsumme des Systemprogrammes
 ØØ   Ø        ; 2. Zeile, immer ØØ  Ø
```

```
PAS  31...34   ; Prüfsumme des Anwenderprogrammes, wobei 31...34 entspre-
               ;   chend den Programmabschnitten  1.K...4.K eingegeben wird.
 XX  XXXX      ; Bezugsgrösse
```

Die richtige Bezugsgrösse für das Anwenderprogramm erhält man durch Abarbeiten des entsprechenden PAS-Befehls in der Betriebsart STEP. Die PCAØ zeigt diese Prüf-Summe auf dem Display des Eingabegerätes einige Sekunden lang an. In der Betriebsart PROG kann anschliessend die entsprechende Bezugsgrösse in der 2. Zeile des PAS 31...34 Befehles eingegeben werden, wobei berücksichtigt werden muss, dass jede Eingabe die Prüfsumme des entsprechenden Programmabschnittes erneut verändert.

Achtung: die Ausführzeit für diese Befehle ist relativ lang:

PAS 3Ø ≙ 28,Øms
PAS 31...34 ≙ 8,3ms

Die "Check-Sum" soll daher nur ausgeführt werden, wenn der zu überwachende Ablauf dies zulässt, z.B. beim Einschalten der PC, am Ende eines Ablaufzyklus etc.

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 Aenderung dieser "Check-Sum".

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 1540   } 2.K
25   JIZ   35

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

35 ←SEO    16  } Alarmausgang setzen
36  JMP    35  } ausserhalb des Ar-
                  beitsprogrammes
```

Vorgehen:
- Nach Programmeingabe und Prüfung ist Betriebsart STEP zu wählen.
- ADR 23 + eintippen
  → die Bezugsgrösse für 2.K (PAS 32) erscheint für ca. 2Ø sec.
- Bezugsgrösse in Betriebsart PRG auf ADR 24 eingeben (+)
- Gleiches Vorgehen für PAS 31

saia pc

ANLAGE:

Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				

Mnemocode	Zahlencode	Befehl wird nur ausgeführt wenn ACCU =	Befehl setzt ACCU = 1	ACCU wird beeinflusst
NOP	ØØ			
STH	Ø1			X
STL	Ø2			X
ANH	Ø3			X
ANL	Ø4			X
ORH	Ø5			X
ORL	Ø6			X
XOR	Ø7			X
NEG	Ø8			X
DYN	Ø9			X
OUT	1Ø			
SEO	11	1		
REO	12	1		
COO	13	1		
STR	14	1		
SCR	15	1		
SEI	16		X	X
INC	17	1		
DEC	18	1		
SEA	19		X	X
JMP	2Ø		X	X
JIO	21		X	X
JIZ	22		X	X
JMS	23		X	X
RET	24		X	X
WIH	25		X	X
WIL	26		X	X
INI	27			X
DEI	28			X
PAS	29			
DOP	3Ø	Ø		
DTC	31	1		

Ein-, Ausgänge	Ø - 255
Zeitglieder, Zähler	256 - 287
Merker	288 - 764
Haftspeicher	765 - 999

saia® pc

ANLAGE:

Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
ØØ				
1				
2				
3				
4				
5				
6				
7				
8				
9				
1Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
2Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
3Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
4Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				

Schritt	Mnemo-code	Zahlen-code	Operand	Kommentar
5Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
6Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
7Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
8Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				
9Ø				
1				
2				
3				
4				
5				
6				
7				
8				
9				

Logik-Befehle										Aktions-Befehle									Sprung-Befehle							Hilfs-Befehle				
ØØ	Ø1	Ø2	Ø3	Ø4	Ø5	Ø6	Ø7	Ø8	Ø9	1Ø	11	12	13	14	15	16	17	18	19	2Ø	21	22	23	24	25	26	27	28	29	3Ø	31
NOP	STH	STL	ANH	ANL	ORH	ORL	XOR	NEG	DYN	OUT	SEO	RED	COO	STR	SCR	SEI	INC	DEC	SEA	JMP	JIO	JIZ	JMS	RET	WIH	WIL	INI	DEI	PAS	DUP	DIC

Eingänge Ausgänge	Ø-255
Zeitglieder Zähler	256-287
Merker	288-764
Haftspeicher	765-999

SODECO-SAIA AG CH-3280 MURTEN SCHWEIZ

LANDIS & GYR

SAIA