VEM

PROGRAMMIERBARE
STEUERUNG
PS 2000

Programmieranleitung

424812-0 Pa4

VEB NUMERIK "KARL MARX"
KARL-MARX-STADT

Programmierbare Steuerung

PS 2000

Programmieranleitung

Die vorliegende Programmieranleitung beschreibt die Programmierung der programmierbaren Steuerung PS 2000. Sie befähigt die Anwender, den gewünschten Funktionsinhalt selbst in das Programm umzusetzen.

Ausgabe	Änd.-Ben.-Nr.	Tag	Name	Nichtt. Maße:		Halbzeug, Werkstoff:	
				1978	Tag	Name	Benennung: (PS 2000) Programmieranleitung
				Gez.			
				Bearb.	1.12.	Rößler	
				Gepr.	14.12.	Roland	
				Stand.	14.12.	Schmidt	
				VEB NUMERIK „KARL MARX" Karl-Marx-Stadt			Zeichnungs-Nr.: 424812-0 Pa 4
				793176			Ersatz für:

Maßstab

Blatt 63 aus Blatt

Stkl.-Nr. 1

Pass.-Nr.

DIN 18 (N 12 - 61) Ag 307-69 BDR VLV Freiberg

Inhaltsverzeichnis

1. Kurzbeschreibung der programmierbaren Steuerung PS 2000

1.1. Prinzipielle Funktionsweise

Für das Aufstellen des Programms sind im Normalfall keine Kenntnisse der PS 2000-hardware erforderlich, da relativ formal vorgegangen werden kann.
Für das Verständnis der Programmierung komplexer Funktionen oder für die Programmentwicklung sind derartige Kenntnisse jedoch unerläßlich.
Das Zusammenwirken der Funktionseinheiten ist aus dem Blockschaltbild (Anhang, Bild 1) ersichtlich.

1.1.1. Wesentlichste Eigenschaften der PS 2000

- Die PS 2000 besitzt 1 Bit-Verarbeitung, demzufolge besteht der Akkumulator (AC) nur aus einem 1 Bit-Speicher.
- Alle zu verarbeitenden Informationen liegen parallel an den Eingängen der Steuerung an.
- Alle errechneten Signale liegen parallel an den Ausgängen der Steuerung an.
- Die Anpassung an den Prozeß (Pegel, Potentialtrennung u. a.) erfolgt durch die Peripheriebausteine.
- Die Funktion, die ausgeführt werden soll, steht als Programm in einem Festwertspeicher. Das Programm wird nach den in diesem Buch dargestellten Regeln vom Anwender aufgeschrieben. Danach erfolgt ein Übertragen des Programms in den Festwertspeicher unter Benutzung eines Programmiergerätes.
- Es erfolgt eine serielle Verarbeitung der im Programm stehenden Anweisungen (Worte).
- Die Steuerung des internen Ablaufes (3 Phasensteuerung sowie Ablauf innerhalb der Phasen) erfolgt durch das Steuerwerk.
- Der Programmumlauf (Zyklus) erfolgt ständig, auch wenn sich die Eingangsinformationen nicht ändern.

- Für das gleiche Programm ist die Zykluszeit stets konstant, da keine Sprunganweisungen möglich sind.
  Sie beträgt normal $T_z \approx 18$ ms ($\hat{=}$ 4K Worte)

  oder bei Verwendung von "END"

  $$T_z \approx (2 + 4 \cdot 10^{-3} \cdot \text{Wortanzahl})\ \text{ms}$$

1.1.2. Interner Ablauf eines Programmzyklusses

- Phase 1 (Eingabephase) :
  Alle 512 theoretisch möglichen Eingänge (Adressen) werden seriell aktiviert und deren Zustände (LOW oder HIGH) über den Multiplexer in den E-RAM geladen und dort gespeichert. Ist kein Eingang angeschlossen, wird LOW übernommen.

- Phase 2 (Verarbeitungsphase) :
  Das Programm wird wortweise (Befehl und Adresse) abgearbeitet.
  Die Wirkung ist von der Befehlsart abhängig:
  Transportbefehle:
  Der unter der angesprochenen Adresse stehende Wert der Information wird wahr/komplementiert aus dem E-RAM/A-RAM in den AC transportiert (LE, LEC, LA, LAC).
  oder:
  Der AC-Inhalt wird wahr/*komplementiert* in die der Adresse entsprechenden Zelle des A-RAM/E-RAM transportiert (TA, TEC).

  Verknüpfungsbefehle:
  Der unter der angesprochenen Adresse stehende Wert im E-RAM/A-RAM wird wahr/komplementiert UND/ODER - verknüpft mit dem Wert des Akkumulators. Das Ergebnis bleibt im AC erhalten.
  Auf diese Weise wächst Schritt für Schritt das Ergebnis eines Programmteilkomplexes (bzw.- einer Boolschen Gleichung), bis es dann als Endergebnis aus dem AC in den A-RAM transportiert werden kann.
  (UE, UEC, UA, UAC, OE, OEC, OA, OAC)

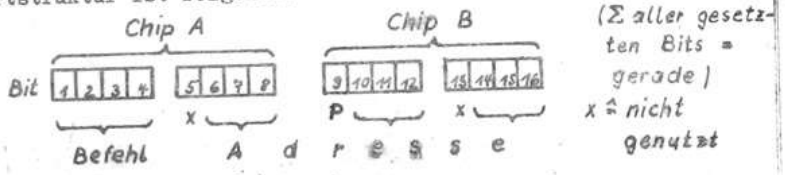
  Zusatzbefehle:
  Bei Zusatzbefehlen erfolgt keine Informationsverarbeitung. Es werden bestimmte interne Zustände der PS 2000 hergestellt, mit deren Hilfe die Programmierung vereinfacht werden kann (ausführliche Erläuterung in Abschnitt 2.3).

Die Phase 2 endet mit dem Wort 4095 oder mit dem Befehl END. Bei ihrem Abschluß ist garantiert, daß alle Ergebnisse berechnet und reif zur Ausgabe sind.

- Phase 3 (Ausgabephase):
  Alle 512 Zellen des A-RAM werden seriell aktiviert und deren Zustände (LOW oder HIGH) über den Demultiplexer an die Ausgänge gegeben, unabhängig davon, ob Ausgänge angeschlossen sind oder nicht.
  Nach Ablauf der Phase 3 beginnt ein neuer Programmzyklus mit der Phase 1.

1.2. Programmsprache und interne Kodierung

Jedes Wort im Festwertspeicher stellt intern ein Bitmuster dar, bestehend aus 16 Bit. Davon entfallen 4 Bit auf den Befehl (Operation) und 9 Bit auf die Adresse (Variation). 2 Bit werden nicht genutzt. Die Wortlänge von 16 Bit ergibt sich aus der Wortlänge der verwendeten Festwertspeicher (pROMs), die je 8 Bit beträgt. Deshalb sind 2 Chips "Parallelgeschaltet" (Chip A, Chip B).
Die Wortstruktur ist folgende:



Um die äußerst fehleranfällige Arbeit mit dem Bitmuster zu umgehen, werden jeweils 4 Bits zu einer Gruppe zusammengefaßt und hexadezimal kodiert. In dieser Kodierung arbeitet auch das Programmiergerät.
Unter Berücksichtigung der nicht genutzten Bits ergibt sich die Wortstruktur zu:

Befehl ≙ 1 Zeichen hexadez. (Ø ... 9, A ... F)

Adresse ≙ 3 Zeichen oktal (jeweils Ø ... 7) (ohne P)

Mit diesen 4 Zeichen eines Wortes ist bereits gut arbeitbar, für den Anwender haben sie jedoch keinen Bezug auf das zu lösende Problem. Deshalb erweist sich die Einführung einer problembezogenen Programmsprache als sehr nützlich.

Für den Befehl wird eine Symbolik eingeführt, die unmittelbaren Bezug auf die durchzuführende Operation nimmt und leicht erlernbar ist. Weiterhin läßt sie sich direkt aus der Darstellung des Problems als Signal- bzw. Funktionsschaltplan ableiten. Zahl, Bedeutung und interne Kodierung der Befehlskurzzeichen sind im Anhang, Tabelle 1 dargestellt.
Anstatt der 3-stelligen oktalen Adresse wird direkt mit der Kurzbezeichnung der Information (bzw. dem Prozeßelement) gearbeitet, die der jeweiligen Adresse symbolisch zugeordnet ist. Die Zuordnung der Informationen zu den Adressen erfolgt bei der Projektierung der PS 2000 und soll hier nicht betrachtet werden.
Somit ist das Steuerungsproblem relativ unabhängig von der PS 2000 selbst programmierbar.
Das Programm hat z. B. folgende Struktur:

Anwender		intern		
Befehl	Name	Chip A	Chip B	
.	.			
.	.			
.	.			
.	.			
LE	START	5 Ø	Ø 3	
UEC	D 101	Ø Ø	3 4	P nicht berücks.
UA	SP 1	8 2	5 7	
TA	Y 7	E 3	Ø Ø	
.	.			
.	.			
.	.			
.	.			

Das vorliegende Programmierhandbuch ist generell in dieser Programmsprache abgefaßt. In dieser Sprache hat auch der Anwender unter Beachtung der Regeln dieses Buches den Programmbeleg auszufüllen.
Für das Umsetzen der Programmsprache in die interne Kodierung unter Beachtung der projektierten Adressen kommen maschinelle Verfahren zur Anwendung.
Form und Art der anzuwendenden Belege sowie des einzusetzenden Verfahrens sind von der Lieferform der PS 2000 abhängig.
(vgl. Projektierungsvorschrift 424813-9 Pro 4).

2. Programmieren von Elementarfunktionen

2.1. Datentransport

Um die Daten im Akkumulator verarbeiten zu können, müssen sie erst einmal dorthin geladen werden.
Deshalb beginnt jedes Programmteilstück mit einem Ladebefehl.
Es ist zu unterscheiden, ob der Datentransport

- aus dem E-RAM (Eingänge) oder
  aus dem A-RAM (Ausgänge)
- wahr oder nicht wahr

erfolgen soll. Somit ergeben sich die Befehle:

LE, LEC, LA, LAC.

Die in einem Wort hinter dem Befehl stehende Adresse gibt an, welche Information transportiert werden soll, z. B.

```
LE  S 24
```

Nach erfolgter Verknüpfung (Verarbeitung) ist das Ergebnis des behandelten Programmteilstückes aus dem Akkumulator in den A-RAM zu übertragen, damit der AC für das nächste Programmteilstück frei wird. Deshalb endet jedes Programmteilstück mit dem Befehl TA.
Die hinter dem Befehl stehende Adresse gibt an, auf welche A-RAM-Zelle (und damit auf welchen Ausgang) der AC-Inhalt gegeben wird, z. B.

```
TA  K 17
```

Das Ergebnis wird wahr ausgegeben. Wird es nicht wahr benötigt, ist Zwischenspeicherung anzuwenden, wozu hier die gleiche A-RAM-Zelle benutzt werden kann, z. B.

```
    ⋮
TA  K 17  (Zwischenergebnis)
LAC K 17
TA  K 17  (Endergebnis)
```

2.2. Logische Verknüpfungen

Die eigentliche Verarbeitung der Daten erfolgt mit Hilfe der log. Verknüpfungen UND und ODER.
Es ist zu unterscheiden nach

- Art der Verknüpfung (UND / ODER)
- Verknüpfung mit E-Ram oder A-RAM
- Verknüpfung mit dem wahren oder nicht wahren Wert

Somit ergeben sich die Befehle:

UE, UEC, UA, UAC, OE, OEC, OA, OAC.

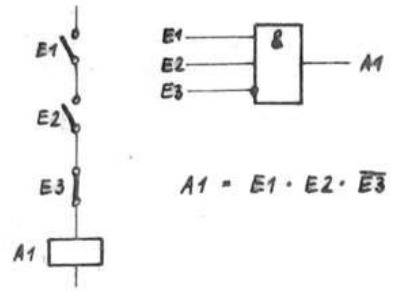
Die Ausführung erfolgt so, daß der bestehende AC-Inhalt mit dem wahren bzw. nicht wahren Wert der angesprochenen Zelle im E-RAM bzw. A-RAM logisch verknüpft wird.
Das Ergebnis wird wiederum im AC abgelegt.

Beispiel: UEC S 14 // S 14 ≙ Adr. 007

Der AC-Inhalt wird mit dem Komplement des Wertes der Zelle 007 im E-Speicher UND - verknüpft.
(Welche Informationen welchen Adressen zugeordnet werden, wird bei der Projektierung der PS 2000 ermittelt und spielt bei der Programmierung mit Namen keine Rolle).

2.2. Bl. 2

Beispiel 1:



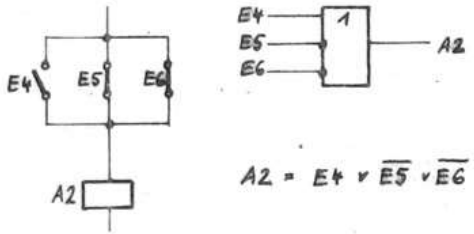
Programm:

LE	E1
UE	E2
UEC	E3
TA	A1

oder:

LEC	E3
UE	E2
UE	E1
TA	A1

Beispiel 2:



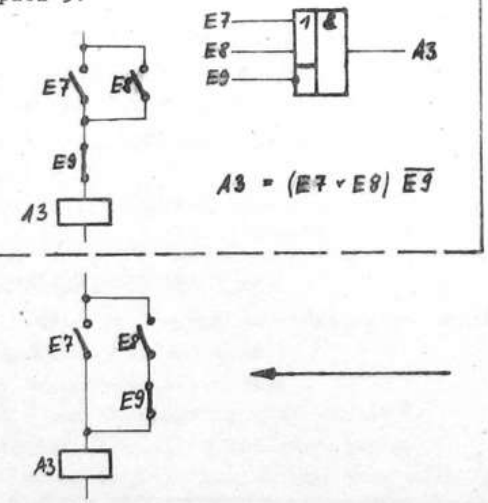
Programm:

LE	E4
OEC	E5
OEC	E6
TA	A2

oder:

LEC	E6
OEC	E5
OE	E4
TA	A2

Beispiel 3:



Programm:

LE	E7
OE	E8
UEC	E9
TA	A3

Beachte:

LEC	E9
UE	E8
OE	E7
TA	A3

Falsch! Dieses Programm entspricht nebenstehender Funktion!

2.3. Zusatzbefehle

Um die Programmierung zu erleichtern, wurden einige zusätzliche Befehle eingeführt.

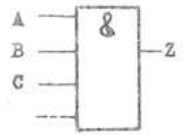
2.3.1. Befehl KO

KO ≙ keine Operation
Kodierung "H" in allen Bitstellen eines Wortes (Befehls- und Adressenteil), vgl. Tabelle 1.

Es erfolgt bei mit KO belegten Worten von der Steuerung keine Operation unter Beibehaltung sämtlicher Belegungen in den Speichern.

Anwendung:

1) Erstprogrammierung: Anlegen einer Reserve an bestimmten Stellen für später zu erwartende Erweiterungen bzw. Änderungen. Damit kann umständlichen Programmverschiebungen vorgebeugt werden.
   z. B. wortweise Reserve



LE	A
UEC	B
UE	C
KO	
TA	Z

} Platz für Erweiterung

2) Änderungen in bereits kodierten PROMS

z. B.

alt		neu				
LE	A	LE	A		LE	A
UEC	B	KO			UE	C
UE	C	UE	C	≙	TA	Z
UE	D	KO				
TA	Z	TA	Z			

2.3.2. Befehl NUL

NUL ≙ Leerwort

Kodierung "L" in allen Bitstellen eines Wortes. Damit können später einzufügende Worte direkt in den PROM geschrieben werden, ohne ihn vorher zu löschen.

Anwendung: komplexweise Reserve

```
    .
    .
    .
TA   M      Ende eines Programmteiles
NUL
NUL
 .          Platz für Erweiterungen
 .
 .
NUL
LE   M      Beginn eines Programmteiles
 .
 .
 .
```

Hinweis: Die Kodierung "L" in allen Bitstellen entspricht gleichzeitig der Operation UE mit der Adresse (ØØØ)8. Da diese Adresse in der Regel mit einer Information belegt ist, darf der Befehl NUL nur zwischen Programmteilen, nicht innerhalb eines Programmteiles angewandt werden.

2.3.3. Befehl END

END = Programmende
Mit Erkennung von END bricht die Steuerung die Phase 2 (Verarbeitungsphase) ab und geht zur Phase 3 (Ausgabe) über. Dadurch tritt eine Verkürzung der Zykluszeit (T_z) ein.
z. B. ohne END:
$T_z \approx 18$ ms (auch wenn nur 1k Worte progr. sind)
mit END nach 1 k Worten:
$T_z \approx 6$ ms

2.3.4. Hauptregister (H-Register)

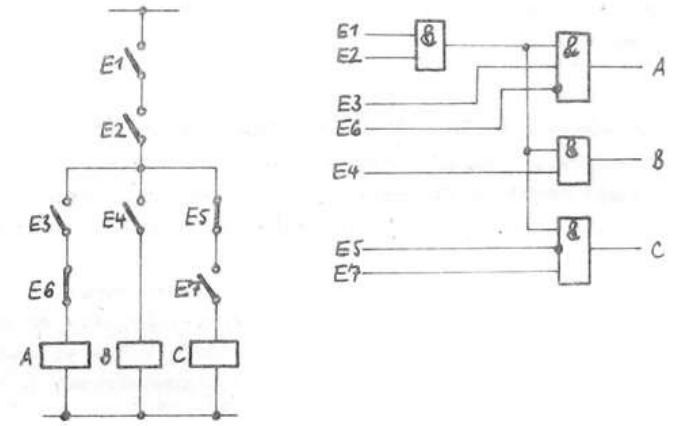
HS = Hauptregister aktivieren
HR = Hauptregister löschen

Das H-Register kann zur Programmvereinfachung (Einsparung von Worten) benutzt werden, wenn in einem logischen Komplex eine oder mehrere übergeordnete Variable auftreten, die mit den übrigen Teilzweigen konjunktiv verknüpft sind.

Folgendes Prinzip liegt zugrunde:

1) alle Variable der übergeordneten Funktion verknüpfen (Ergebnis steht im AC), mit dem Befehl HS wird der AC-Inhalt in das H-Register überführt.
2) Programmieren der verbleibenden Teilzweige, (bei jeder Ausgabe mit TA wird das Ergebnis mit dem H-Registerinhalt konjunktiv verknüpft).
3) Nach Programmierung des log. Komplexes ist die Wirkung des H-Registers durch den Befehl HR aufzuheben.

Beispiel:



Programm:

ohne H-Register		mit H-Register[1]
	oder	
LE E1	LE E1	LE E1
UE E2	UE E2	UE E2
UE E3	TA X	HS
UEC E6	UE E3	LE E3
TA A	UEC E6	UEC E6
LE E1	TA A	TA A
UE E2	LA X	LE E4
UE E4	UE E4	TA B
TA B	TA B	LEC E5
LE E1	LA X	UE E7
UE E2	UEC E5	TA C
UEC E5	UE E7	HR
UE E7	TA C	
TA C		

Hinweis:

1) Bei größeren Komplexen tritt eine wesentliche Verkürzung des Programms ein.

2.3.6. Register für direkte Eingabe/Ausgabe (D-Register)

Bei einer programmierbaren Steuerung werden an Signalbreite und - frequenz der externen Informationen Anforderungen gestellt, die sich aus der Zykluszeit T_z ableiten lassen. (Berechnung der T_z nach 1.1.).
Da die Informationseingabe nur zur Phase 1 eines jeden Zyklusses erfolgt, muß gewährleistet werden, daß die Signalbreite so groß ist, daß das Signal mit Sicherheit erfaßt wird ($> T_z$). Ebenso darf die Signalfrequenz nur so hoch sein, daß jeder Signalwechsel erfaßt wird.

Die Ausgabe der errechneten Informationen erfolgt erst zur Phase 3. Damit entsteht praktisch eine Totzeit zwischen dem auslösenden Moment und der Wirkung der Steuerung. Diese Tatsache ist bei Vorgängen, bei denen es zeitliche Probleme gibt, zu berücksichtigen.

Diese Probleme können in gewissen Grenzen mit dem D-Register gelöst werden. Das Prinzip besteht darin, daß in Abweichung von der normalen Wirkungsweise während der Phase 2 direkte Eingaben vom Prozeß und direkte Ausgaben an den Prozeß möglich sind.
Diese direkte Eingabe oder Ausgabe einer Information erfolgt nur bei gesetztem D-Register in dem Augenblick der Phase 2, in dem das Wort mit der Adresse dieser Information abgearbeitet wird.

DS $\hat{=}$ D-Register setzen

DR $\hat{=}$ D-Register rücksetzen

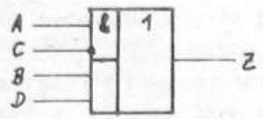
Bei gesetztem D-Register haben die Befehle LE und TA eine veränderte Bedeutung. Die Befehle LEC, LA, LAC sind nicht zulässig.

Wirkung	Befehl/Adr.	Bedeutung
direkte Eingabe	LE xxx	Laden des Wertes der Adresse xxx vom Prozeß in die entspr. Zelle des E-RAM
direkte Ausgabe	TA xxx	Laden des Wertes der Adresse xxx aus der entspr. Zelle des A-RAM an den Prozeß

Die direkte E. oder A. genügt zur Lösung des Problems noch nicht. Der Programmteilkomplex, in dem die kritischen Informationen verarbeitet werden, ist so oft in einem bestimmten Abstand zu programmieren, bis das Zeitproblem kompensiert ist (vgl. Beispiel). Damit geht der zeitliche Vorteil zu Lasten der Programmlänge.
Für den Teilkomplex ist wie folgt vorzugehen:

1) Setzen des D-Registers, Eingabe der krit. Informationen, Löschen des D-Registers
2) Normale Programmierung der log. Verknüpfungen des Teilkomplexes
3) Setzen des D-Registers, Ausgabe der krit. Informationen, Löschen des D-Registers.

Beispiel:



Gesamtprogrammlänge:
1,5 K, mit END abgeschlossen
($T_Z \approx 8$ ms)

A, B, Z krit. Informationen
(Signalbreite $T_1 \approx 4$ ms)

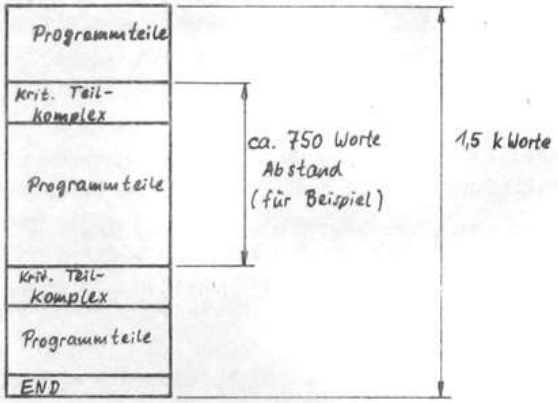
Programm: Teilkomplex

```
DS
LE  A
LE  B
DR
```
} Laden der Werte von A, B unmittelbar vom Prozeß in den E-Speicher

```
LE  A
UEC C
DE  D
OE  B
TA  Z
```
} log. Verknüpfung und Abspeichern des Ergebnisses im A-Speicher

```
DS
TA  Z
DR
```
} Ausgabe von Z unmittelbar an den Prozeß

Die Einordnung des krit. Teilkomplexes in das Gesamtprogramm hat im Beispiel zweimal zu erfolgen:



Hinweis: 1) steht der krit. Teilkomplex am Programmanfang, so kann die direkte Eingabe entfallen.

2) steht der krit. Teilkomplex am Programmende, so kann die direkte Ausgabe entfallen.

3. Verwendung von Zwischenspeichern

3.1. Definition

Unter Zwischenspeicher soll ein Speicherelement verstanden werden, das bestimmte Ergebnisse oder Teilergebnisse aufnimmt, die über einen bestimmten Zeitraum aufbewahrt werden müssen.
Dabei handelt es sich ausschließlich um Speicherzellen im A-RAM, die nicht durch Ausgangs-Prozeßelemente oder Sonderbausteineingänge belegt sind. Somit sind die Zwischenspeicherinhalte nur steuerungsintern durch das Programm abfragbar. Die konkreten Zellen werden bei der Projektierung der PS 2000 zugeordnet. Es genügt, im Programm symbolisch mit Namen zu operieren.

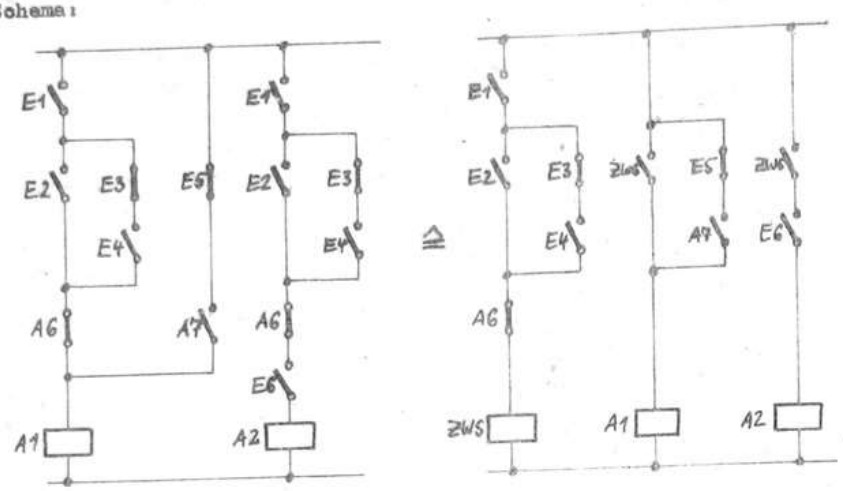
Zwischenspeicher werden verwendet

1. als Programmierhilfe
2. zur Vereinfachung der Funktionsdarstellung

Es ist zu beachten, daß die Speicherwirkung nur über diejenigen Programmzyklen erhalten bleibt, in denen für die Erregung der Speicherzelle auch "H" errechnet wird.
Soll eine Speicherwirkung auch bei verschwindender Erregung erzielt werden, so muß die "Selbsthaltung" ausdrücklich programmiert werden (vgl. Abschnitt 6.2)

3.2. Vereinfachung der Funktionsdarstellung

Schema:



≙

Programm:

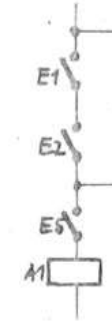
LEC	E3	LEC	E3	
UE	E4	UE	E4	
OE	E2	OE	E2	
UE	E1	UE	E1	
UAC	A6	UAC	A6	5
TA	A1	TA	ZWS	
LEC	E5	LEC	E5	
UA	A7	UA	A7	
OA	A1	OA	ZWS	
TA	A1	TA	A1	10
LEC	E3	LA	ZWS	
UE	E4	UE	E6	
OE	E2	TA	A2	
UE	E1			
UAC	A6			15
UE	E6			
TA	A2			

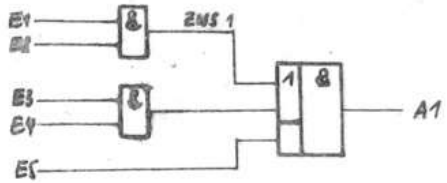
Hinweis: Durch Zwischenspeicherung bei gleichen, mehrfach auftretenden Teilschaltungen tritt eine Vereinfachung im Programm auf. Diese Zwischenspeicher ergeben sich meist schon beim Entwurf der Schaltung.

3.3. Programmierhilfe

Enthalten disjunktiv verknüpfte Zweige zwei oder mehr Elemente, so ist eine Zwischenspeicherung notwendig.

Schema:

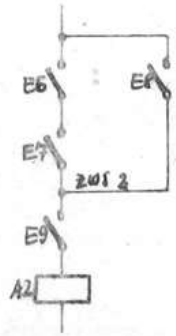




Programm:	**LE**	**E1**
	UE	**E2**
	TA	**ZWS 1**
	LE	**E3**
	UE	**E4**
	OA	**ZWS 1**
	UE	**E5**
	TA	**A1**

Sonderfall:

Befindet sich in einem Zweig nur 1 Element, so kann die Zwischenspeicherung durch geeignete Programmierreihenfolge umgangen werden.

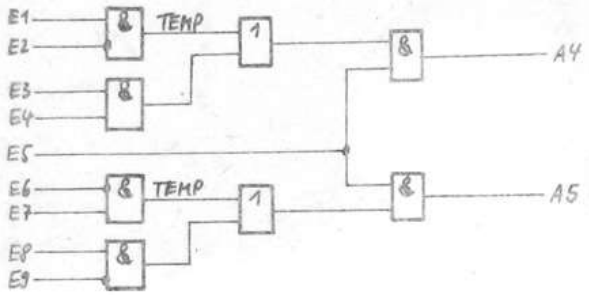


LE	**E6**	**aber:**	**LE**	**E8**
UE	**E7**		**TA**	**ZWS 2**
OE	**E8**		**LE**	**E6**
UE	**E9**		**UE**	**E7**
TA	**A2**		**OA**	**ZWS 2**
			UE	**E9**
			TA	**A2**

unzweckmäßige Programmierung !

3.4. Temporäre Zwischenspeicher

Unter temporären Zwischenspeicher werden Speicherelemente verstanden, in denen Ergebnisse nur kurzzeitig, und zwar nur für Bruchteile eines Programmzyklusses aufbewahrt werden müssen. Daraus ergibt sich, daß die gleiche Speicherzelle für mehrere voneinander unabhängige Zwecke verwendet werden kann. Die Anwendung ist günstig als Programmierhilfe gemäß Absehnitt 3.3.
Beispiel:



Programm:

LE	E1		
UEC	E2		
TA	TEMP	——→	Zwischenspeichern
LE	E3		
UE	E4		
OA	TEMP	——→	ab hier ist Zwischenspeicher frei
UE	E5		und kann neu genutzt werden
TA	A4		
LEC	E6		
UE	E7		
TA	TEMP	——→	2. Verwendung dergleichen Speicherzelle
LE	E8		
UEC	E9		
OA	TEMP	——→	ab hier ist Zwischenspeicher wieder frei
UE	E5		
TA	A5		

3.5. Bemerkungen zum Befehl TEC

Nach der Berechnung wird das Ergebnis der interessierenden Größe normalerweise in den A-RAM transportiert (Befehl TA), von wo aus es im weiteren Verlauf der Phase 2 abgerufen und weiterverknüpft und zur Phase 3 ausgegeben wird. Dabei ist es unerheblich, ob es sich um eine Prozeßgröße oder um einen Zwischenspeicher handelt.

In Abschnitt 3.1. wurde bereits erläutert, daß erforderliche Zwischenspeicher in A-RAM-Zellen untergebracht werden, die nicht durch Prozeßgrößen belegt sind. In der Praxis reicht die Zahl der möglichen Zwischenspeicher auch völlig aus. Sollten dennoch zusätzliche Zwischenspeicher erforderlich sein, so ist es möglich, diejenigen Zellen des E-RAM damit zu belegen, die nicht durch Eingangs-Prozeßelemente oder Sonderbausteinausgänge belegt sind.
Nach der Berechnung wird das zwischenzuspeichernde Ergebnis mit dem Befehl TEC in die angesprochene E-RAM-Zelle transportiert und steht dann dort für weitere Verknüpfungen zur Verfügung.

Beachte:

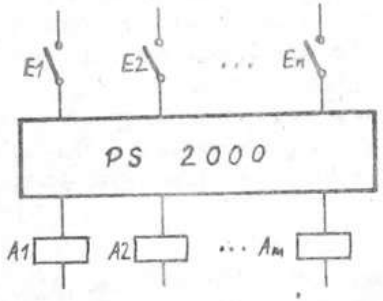
1) Zur Phase 1 werden die LOW/HIGH-Zustände aller möglichen Eingänge in den E-RAM übertragen, unabhängig davon, ob Prozeßelemente oder Sonderbausteine angeschlossen sind oder nicht. Das heißt, daß auch die o.g. E-RAM-Zelle überschrieben werden. Das bedeutet wiederum, daß E-RAM-Zellen als Zwischenspeicher nur über max. einen Programmzyklus verwendet werden können (also nur als temporäre Zwischenspeicher).

2) Aus internen Gründen ist der Transport aus dem AC in den E-RAM nur mit dem Befehl TEC möglich. Das heißt, daß im E-RAM das Komplement des errechneten Wertes steht. Bei einer Weiterverknüpfung ist dann wiederum mit Komplementbefehlen (z.B. UEC , OEC) zu arbeiten, wenn die Verknüpfung wahr in bezug auf den ursprünglichen Wert erfolgen soll.

4. Einfluß der externen Informationen auf die Programmierung

4.1. Prinzip

Zwischen der Art der extern an die PS 2000 angeschlossenen Informationen (Eingänge) und ihrer Verwendung in der zu realisierenden Funktion besteht ein Zusammenhang, der bei der Programmierung zu beachten ist.
Ein wesentliches Merkmal einer PS ist es, daß alle Prozeßinformationen parallel und unabhängig voneinander an die Steuerung angeschlossen sind.



Die Verknüpfung der Prozeßinformationen erfolgt in der Steuerung. Daraus und aus der digitalen Verarbeitung ergibt sich, daß

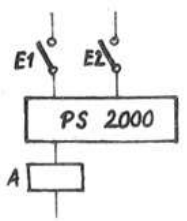
- jedes Eingabeprozeßelement nur 2 Zustände (LOW/HIGH) liefern muß. Damit werden z.B. Doppelkontakte, Wechsler, Mehrebenenwahlschalter u.a. überflüssig. In allen Fällen genügt ein einfacher Schließer (bei Kontaktbauelementen).
- es im Prinzip gleich ist, ob das angeschlossene Prozeßelement in Grundstellung ein wahres oder ein nicht wahres Signal liefert. Günstig ist allerdings der Normalfall (vgl. Absohn. 4.2.).

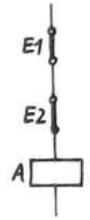
Die folgenden Ausführungen erfolgen in der Kontaktdarstellung, weil hier die Zusammenhänge am deutlichsten darstellbar sind.

4.2. Schließer

Normalfall! Die externe Information ist als Schließer (Grundstellung) an die PS 2000 angeschlossen.

Beispiel: 1) Verwendung als Schließer 2) Verwendung als Öffner





Programme:

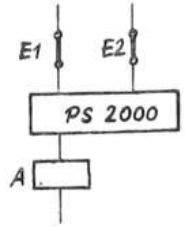
LE	E1	LEC	E1
UE	E2	UEC	E2
TA	A	TA	A

Auf diesen Normalfall werden im Programmierhandbuch alle Fälle zurückgeführt, sofern nicht ausdrücklich andere angegeben.

4.3. Öffner

Die externe Information ist als Öffner (Grundstellung) an die PS 2000 angeschlossen.

Beispiel: 1) Verwendung als Schließer 2) Verwendung als Öffner



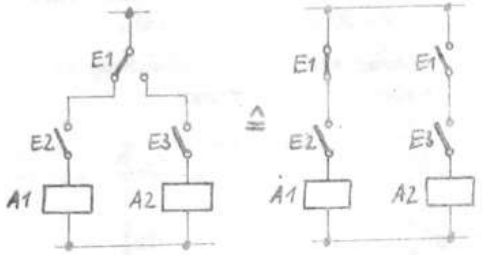
Programme:

LEC	E1	LE	E1
UEC	E2	UE	E2
TA	A	TA	A

4.4. Wechsler

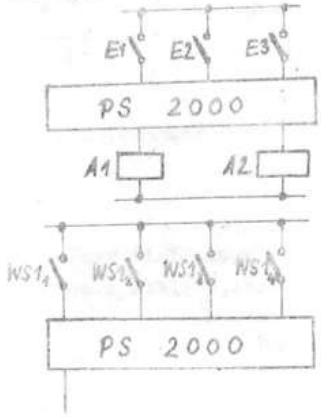
Ein Wechslerkontakt ist stets auf einen Schließer und einen Öffner zurückführbar.

Beispiel:

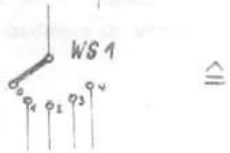


LEC	E1
UE	E2
TA	A1
LE	E1
UE	E3
TA	A2

Damit genügt der Anschluß von E1 an die Steuerung über nur 1 Kanal (Adresse) mittels Schließer (oder Öffner).



4.5. Wahlschalter



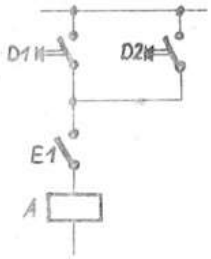
≙

Wahlschalter werden auf Schließer zurückgeführt. Jede Wahlschalterstellung, die eine Funktion ausübt, belegt an der Steuerung einen Kanal (eine Adresse). Damit erfolgt die Programmierung nach der bereits erläuterten Grundsätzen.

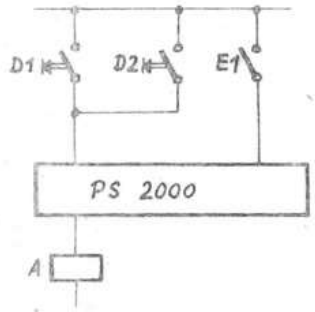
4.6. Verkettete Eingangsinformationen

Externe Informationen, die **immer** fest miteinander verknüpft sind und **nicht einzeln** in ihrer Funktion benötigt werden, können als **eine Information** betrachtet werden. Sie belegen dann auch nur 1 Kanal. Der Verknüpfungsgrad ist beliebig.

Beispiel:



Annahme: D1, D2 fest disj. verknüpft



Programm

LE	(D1vD2)
UE	E1
TA	A

Durch diese Maßnahme können Peripheriebausteine und evtl. Verkabelung eingespart werden, wenn die verketteten Prozeßelemente räumlich nahe zueinander angeordnet sind.

5. Programmierung von Funktionen unter Verwendung von Sonderbausteinen

5.1. Prinzip

Sonderbausteine sind Kartenbaugruppen mit einem fest vorgegebenen Funktionsinhalt, z. B.

- Zeitstufen, analog
- Zeitstufen, digital
- Zähler, dezimal
- Schieberegister
- Haftspeicher

Sie können anstelle eines Peripheriebausteins an beliebiger Stelle in der Steuerung PS 2000 plaziert werden.
Der Verkehr zu und von den Sonderbausteinen erfolgt ausschließlich intern über das Programm.
Dadurch, daß die auf dem S. realisierte Funktion nicht noch einmal programmiert werden muß, tritt eine erhebliche Programmverkürzung ein.

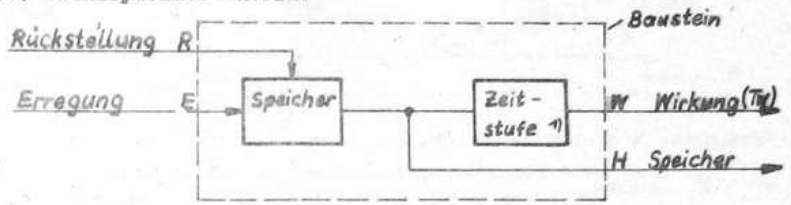
Voraussetzungen für die Anwendung sind:

- daß die auf dem S. realisierte Funktion auch der gewollten Funktion entspricht,
- daß der Platz in der PS 2000 ausreicht, um Sonderbausteine zu stecken.

Die Stellung der Sonderbausteine zu den übrigen Funktionseinheiten der PS 2000 ist aus dem Blockschaltbild (Anhang, Bild 1) erkennbar. Die Eingänge der Sonderbausteine werden über die Ausgänge der PS 2000 mit den notwendigen Informationen versorgt. Die Wirkungen an den Ausgängen der Sonderbausteine können über die Eingänge der PS 2000 abgefragt werden. Diese Tatsache ist besonders bei der Projektierung (Adressenfestlegung) und bei der Programmierung (Befehlsauswahl: Verknüpfung mit dem E-Bereich oder dem A-Bereich) zu beachten!

5.2. Zeitstufe, analog

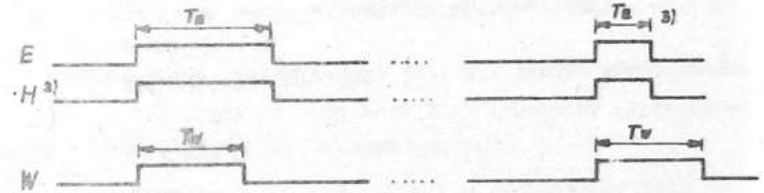
5.2.1. Prinzipieller Aufbau:



Hinweise: 1) mit Erholzeit: Zeitstufe 1stufig, Typ TAE
ohne Erholzeit: Zeitstufe 2stufig, Typ TA
Die gewünschte Zeit Tw wird durch entsprechende RC-Bestückung erzielt. Sie wird fest eingestellt.

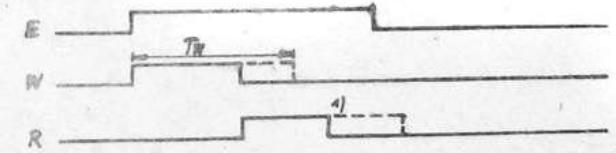
5.2.2. Prinzipielle Funktionsweise:

1) Normale Anwendung



Hinweise: 2) H ist näherungsweise mit dem Verlauf von E identisch (wird zusätzlich von Rückflanke eines internen Taktes gesteuert).
3) Tw läuft unabhängig von T_E ab, entscheidend ist die L/H-Flanke am Speicher.

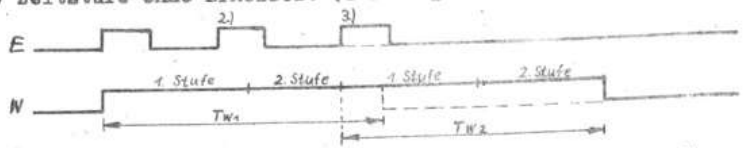
2) Rücksetzen



Hinweis: 1) Trotz noch verhand. E erfolgt keine neue Erregegung der Zeitstufe, da diese nur auf eine L/H-Flanke anspricht.

5.2.2. Bl. 2

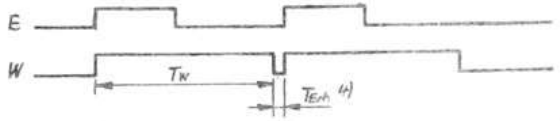
3) Zeitstufe ohne Erholzeit (2 stufig)



Hinweise: 2) Keine neue Erregung, wenn 1. Stufe noch läuft.

3) Neue Erregung, wenn diese in den Lauf der 2. Stufe fällt. Tw wird von Erregung an voll abgearbeitet.

4) Zeitstufe mit Erholzeit (1 stufig)



4) Es muß eine T_{Erh} = 25 ... 250 ms (je nach RC-Bestückung) garantiert werden, sonst wird die nachfolgende Tw verfälscht.

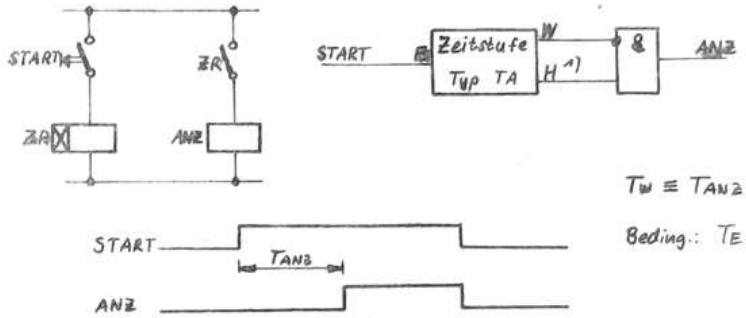
5.2.3. Anwendung

Beispiel 1: Monostab: Multivibrator

Programm:	LE(A)	START	} Zeitstufe anstoßen
	TA	E	
	LE	W	} Wirkung abfragen
	TA	ZEIT	

5.2.3. Bl. 2

Beispiel 2: Zeitverzögerung (Anzugverzögerung)

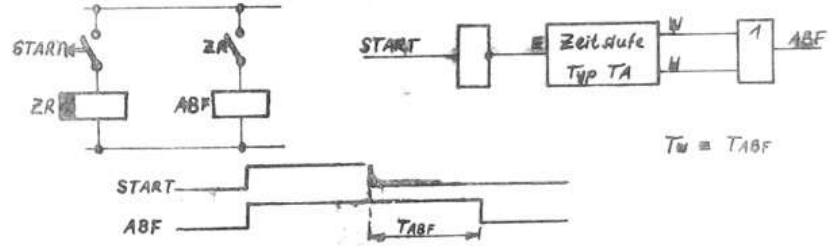


Programm:	LE	START	} Zeitstufe anstoßen
	TA	E	
	LE	H 1) 2)	} Auswertung
	UEC	W	
	TA	ANZ	

Hinweis: 1) Obwohl E und H annähernd im Verlauf identisch sind, ist H aus verarbeitungstechnischen Gründen zu verwenden.

2) Ausgänge der Zeitstufe sind für die PS 2000 Eingänge!

Beispiel 3: Zeitdehnung (Abfallverzögerung)

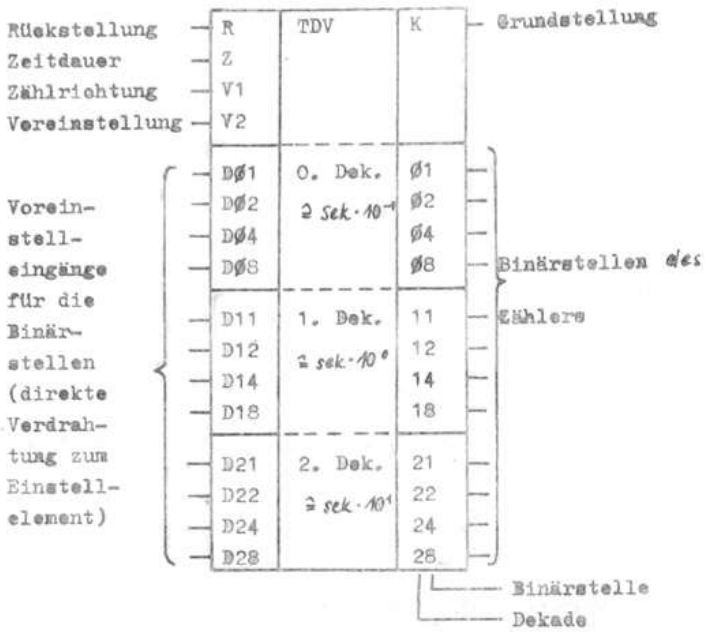


Programm:	LEC	START	} Zeitstufe anstoßen
	TA	E	
	LEC	H 1)	} Auswertung
	OE	W	
	TA	ABF	

Hinweise: 1) Wie Anwendungsbeispiel 2

5.3. Zeitstufe, digital, mit Voreinstellung

5.3.1. Prinzipieller Aufbau



Der Baustein TDV besteht aus einem Taktgenerator 10 Hz (kleinste Einheit = Zehntelsekunden) und einem Zähler (3 Dekaden).

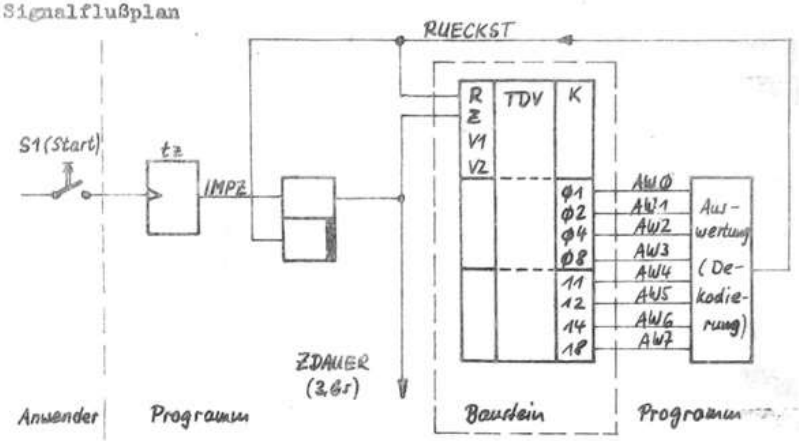
5.3.2. Wirkungsweise

Der Zähler erhält Impulse vom Generator, solange an Z H liegt.
Zählrichtung V1: L ≙ vorwärts
H ≙ rückwärts
Rückstellung zur Grundstellung (alle Bitstellen haben L)
durch H an R.
In Grundstellung steht K zusätzlich auf H.
Übernahme der Voreinstellwerte durch H an V2. In nichtbeschaltete Voreinstelleingänge wird L übernommen. Die Auswertung des Zählerstandes erfolgt über die Ausgänge Ø1 bis 28.

5.3.3. Anwendungsbeispiel 1

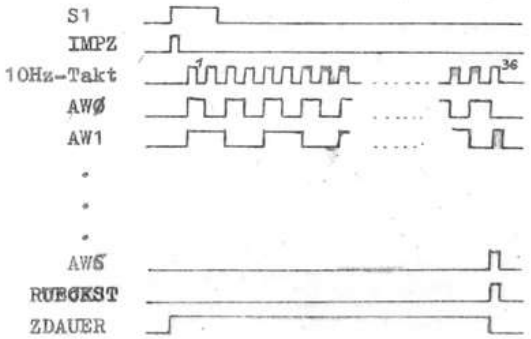
Zeitstufe mit fest eingestellter Zeit (z. B. 3,6 s), keine Voreinstellung notwendig.

Signalflußplan



3,6 sek. ≙ $\overline{AW7} \cdot \overline{AW6} \cdot AW5 \cdot AW4 \cdot \overline{AW3} \cdot AW2 \cdot AW1 \cdot \overline{AWØ}$

Taktdiagramm:



5.3.3. Bl. 2

Programm:

```
LEC   S1      }
OA    IMPB    }
UAC   IMPZ    } Impuls-
TA    IMPB    } schaltung 1)
UE    S1      }
TA    IMPZ    }

                LE    AW5   }
                UE    AW4   }
                UEC   AW3   } Auswertung 2)
                UE    AW2   }
                UE    AW1   }
                UEC   AW0   }
                TA    RUECKST }

                LA    IMPZ    }
                OA    ZDAUER  } Speicher
                UAC   RUECKST }
                TA    ZDAUER  }
```

Hinweise: 1) Die Impulsschaltung schafft definierte Startverhältnisse, d.h. Unabhängigkeit von der Betätigungsdauer von S1 (vgl. Abschnitt 6.3.)

2) Auswertung (Dekodierung) nur bis zur höchsten Stelle mit wahrem Wert programmieren.

3) Erläuterung zur Adressenzuordnung (Projektierung):

Name	Verwendung
S1	Prozeßelement (Eingang)
IMPB, IMPZ	Zwischenspeicher
AW0–AW5	Sonderbausteinausgänge ≙ Steuerungseingänge
RUECKST, ZDAUER	Sonderbausteineingänge ≙ Steuerungsausgänge (gleichzeitige Nutzung als Zwischenspeicher)

Die gewünschte Zeitwirkung kann an ZDAUER abgefragt werden.

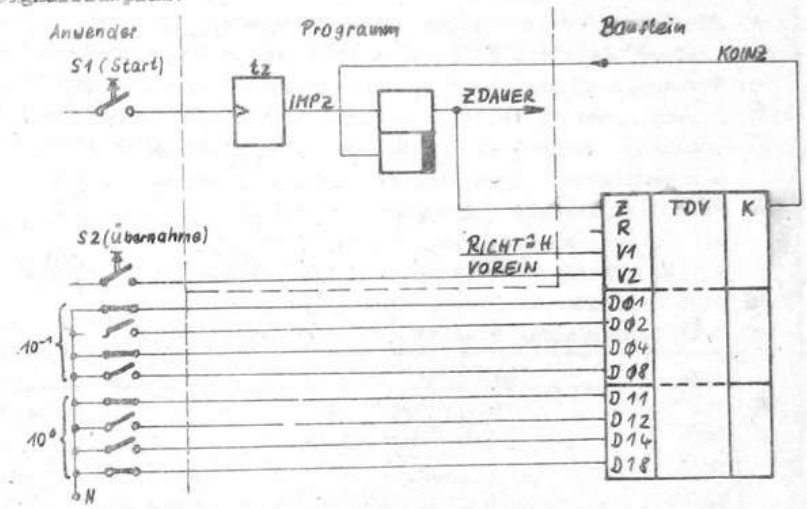
4) Das geschilderte Beispiel ohne Voreinstellung wird man im Normalfall mit Sonderbausteinen Typ TA bzw. TAE realisieren, sofern keine anderen Gründe, wie Genauigkeit u.a. vorliegen. Diese Typen sind wesentlich kostengünstiger.

5.3.4. Anwendungsbeispiel 2

Zeitstufe, Zeitwert variabel über Voreinstellung

(an den Vorwahlschaltern sind z.B. 9,5 sek. eingestellt)

Signalflußplan:



Wirkungsweise:

Einschreiben des voreingestellten Wertes mit S2. Durch S1 starten, damit setzen des Speichers (Zeitbeginn). Zähler zählt rückwärts (H an V1) bis in Grundstellung. Dann liegt H an K und Speicher wird rückgesetzt (Zeitende).

Programm:

```
LE    X       }
OEC   X       } Zählrichtung 2)
TA    RICHT   }
LE    S2      }
TA    VOREIN  } Voreinstellung 3)

                LEC   S1      }
                OA    IMPB    }
                UAC   IMPZ    } Impuls-
                TA    IMPB    } schaltung 1)
                UE    S1      }
                TA    IMPZ    }

                (LA   IMPZ)   }
                OA    ZDAUER  }
                UEC   KOINZ   } Speicher
                TA    ZDAUER  }
```

5.3.4. Blatt 2

Hinweise:

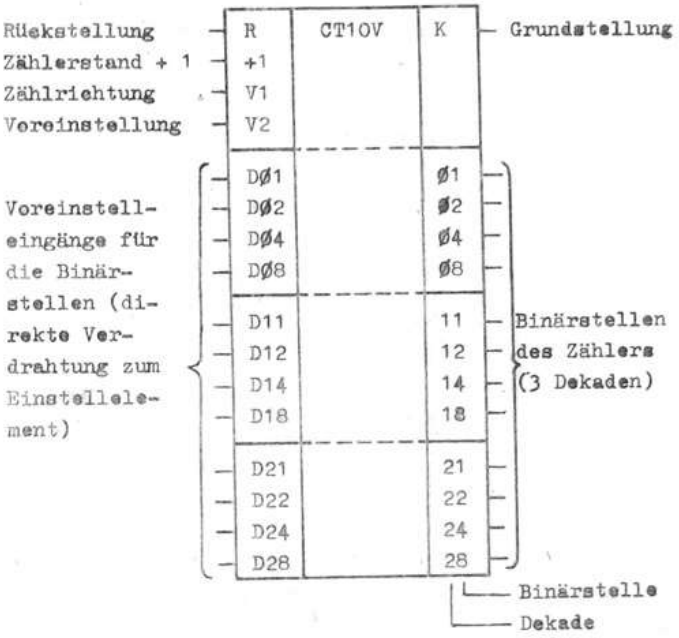
1) wie 5.3.3. Hinweis 1
2) Für X ist jede beliebige Größe einsetzbar (LE X, OEC X liefert immer ein H), die im Programm vorkommt.
3) Der Anschluß der Elemente zur Voreinstellung erfolgt direkt (über Verteiler und Kabel zum X4 des Sonderbausteines). Deshalb sind die Voreinstelleingänge nicht zu programmieren und es sind dafür keine Adressen (bzw. Namen) zu vergeben. Programmiert werden muß lediglich der Übernahmeimpuls zur Voreinstellung.
   Nichtangeschlossene Elemente (z. B. Dekade 10^1) bewirken eine Übernahme von L.
4) Erläuterung zur Adressenzuordnung (Projektierung):

Name	Verwendung
S1, S2	Prozeßelemente
IMPB, IMPZ	Zwischenspeicher
KOINZ	Sonderbausteinausgang ≙ Steuerungseingang
RICHT, VOREIN, ZDAUER	Sonderbausteineingänge ≙ Steuerungsausgänge (ZDAUER wird zusätzl. als Zwischenspeicher genutzt)

Die gewünschte Zeitwirkung kann an ZDAUER abgefragt werden.

5.4. Zähler, dezimal, mit und ohne Voreinstellung

5.4.1. Prinzipieller Aufbau



Beim Typ CT 10 (ohne Voreinstellung) entfallen die Voreinstelleingänge DØ1 bis D28 und der Steuereingang V2.

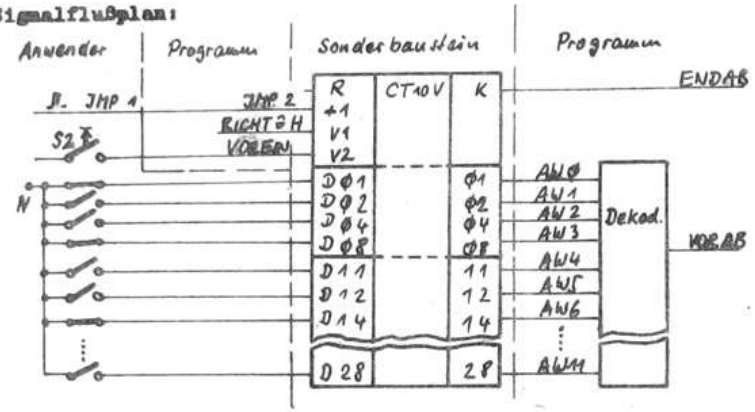
5.4.2. Wirkungsweise

Die an +1 ankommenden Impulse erhöhen oder erniedrigen den Zählerstand jeweils um 1.
Alle anderen Ein- und Ausgänge sind in Abschnitt 5.3.2. beschrieben.

5.4.3. Anwendungsbeispiel

Ein Zähler soll als Teil einer Wegsteuerung eingesetzt werden. Es erfolgt durch S2 eine Voreinstellung des zu verfahrenden Weges. Während der Bewegung sollen entstehende Impulse rückwärts gezählt werden. Der Zählerstand 006 soll zur Vorabschaltung (z. B. von Eilgang auf Schleichgang) und der Zählerstand 000 zur Endabschaltung führen. Dargestellt ist nur die im Zusammenhang mit dem Zähler interessierende Schaltung.

Signalflußplan:



Programm:

```
LE    X        }
GEC   X        } Zählricht.
TA    RICHT    }
LE    S2       } Voreinstellung
TA    VOREIN   }
LE    IMP1
TA    IMP2
```

```
LEC   AW11
UEC   AW1Ø
UEC   AW9
  .
  .
  .
UEC   AW3
UEC   AW2
UEC   AW1
UEC   AWØ
TA    VORAB
```

Hinweis zur Adressenzuordnung:

IMP1, S2 ≙ Eingänge vom Prozeß

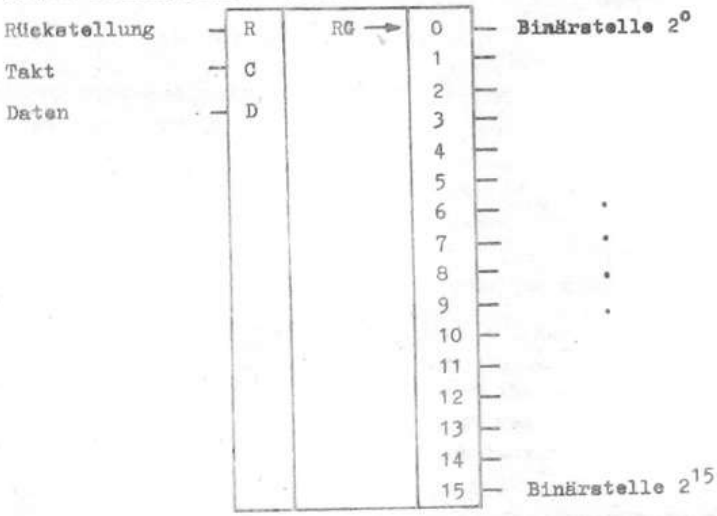
IMP2, RICHT, VOREIN ≙ Sonderbausteineingänge/Steuerungsausgänge

ENDAB, AWØ bis AW11 ≙ Sonderbausteinausgänge/Steuerungseingänge

VORAB ≙ Zwischenspeicher

5.5. Schieberegister

5.5.1. Prinzipieller Aufbau



5.5.2. Wirkungsweise

Die an D anstehende L/H-Wertigkeit wird mit der Vorderflanke des Taktes in die Binärstelle 2^0 übernommen und gleichzeitig erfolgt die Übernahme der Wertigkeiten der Binärstellen in die nächst höhere. Der Registerstand kann an allen Binärstellen abgefragt werden. Ein H an Rückstellung stellt alle Binärstellen auf L.

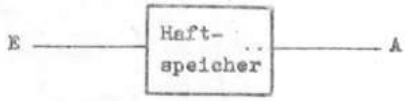
5.5.3. Anwendungsbeispiel

Auf Beispiele kann hier verzichtet werden. Die grundsätzliche Handhabung der Bausteineingänge sowie der Bausteinausgänge ist den Abschnitten 5.3.3. und 5.4.3. vergleichbar.

5.6. Haftspeicher

5.6.1. Prinzipieller Aufbau

Haftspeicher sind Speicherzellen, die bei Spannungsausfall ihren Wert behalten. Sie sind auf einem Sonderbaustein untergebracht.



Sie besitzen Eingang und Ausgang.

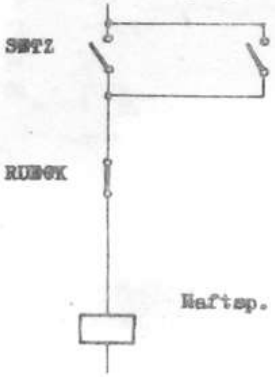
5.6.2. Wirkungsweise

An E ist der Wert (L oder H) anzulegen. Dieser Wert steht nach dem Wiedereinschalten der PS 2000 noch an A an und kann von dort zur weiteren Verarbeitung herangezogen werden.

5.6.3. Programmierung

Die Programmierung ist der eines Speichers gemäß Abschnitt 6.2. ähnlich, d. h. die Selbsthaltewirkung ist ausdrücklich zu programmieren.

Folgendes Prinzip ist anzuwenden (Setz- und Rücksetzbedingung werden als Eingänge angenommen):




```
LE   A
OE   SETZ
UEC  RUECK
TA   E
LE   A
.    weitere Aus-
.    wertung des
.    Haftspeicher-
     ausganges
```


6. Programmierung von Funktionen ausschließlich mittels Programm

6.1. Prinzip

Im folgenden sollen Programme von Funktionskomplexen vorgestellt werden, die häufig vorkommen dürften, wie z. B.

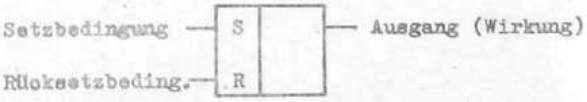
- Speicher
- Impulsgeneratoren
- Vergleicher
- Dekoder
- Zähler
- Schieberegister

Ziel ist es, Programmentwicklungsarbeit einzusparen, indem man das Programm übernimmt und lediglich die Namen der Variablen entsprechend verändert.
Weiterhin werden Hinweise gegeben, die bei individueller Programmentwicklung zu beachten sind.
Im Gegensatz zu der Verwendung von Sonderbausteinen wird der gesamte Funktionsinhalt des Komplexes mittels Programm realisiert. Dadurch werden Sonderbausteine eingespart und damit der Platzbedarf verringert. Die Länge des Programms erhöht sich jedoch. Für die auftretenden Größen (z.B. Binärstellen u.a.) werden Zwischenspeicher verwendet.

6.2. Speicherfunktion

6.2.1. Grundsätze

Es soll mittels Programm eine Wirkung erzielt werden, die der eines Flip-Flops identisch ist.



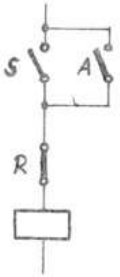
Das wesentlichste Merkmal ist, daß die Speicherwirkung auch dann noch anhalten soll, wenn die Setzbed. nicht mehr erfüllt ist. Erst die erfüllte Rücksetzbed. führt eine Zustandsänderung herbei.

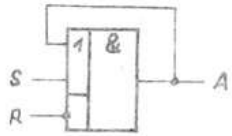
Damit genügt es nicht, einen Zwischenspeicher (Zelle im A-RAM) zu verwenden (dieser würde in der Phase 2 sofort mit "L" überschrieben werden, sobald die Setzbedingung nicht mehr erfüllt ist), sondern die Speicherwirkung muß ausdrücklich programmiert werden.
Für jede Speicherwirkung ist eine A-RAM-Zelle (Adresse) zu projektieren. Dabei ist es gleich, ob die Speicherwirkung extern (an die A-RAM-Zelle ist eine Prozeßelement angeschlossen) oder intern (kein Prozeßelement ist angeschlossen ⟶ Zwischenspeicher) benötigt wird.

Die Programmierung der Speicherfunktion ist leicht anhand von kontaktbehafteten Selbsthaltesschaltungen ableitbar (ungetaktete Speicher).
Im Prinzip ist es auch möglich, getaktete Speicher nachzubilden. Der Takt ist dann extern oder vom Programm bereitzustellen und mit dem eigentlichen R- oder S-Flip-Flop entsprechend zu verknüpfen.

6.2.2. R-Flip-Flop

≙ Speicher dominierend rücksetzen
Schaltung:

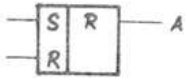




Wertetabelle:

S	R	A
L	L	Zust. bleibt erhalten
L	H	L
H	L	H
H	H	L

Symbol:

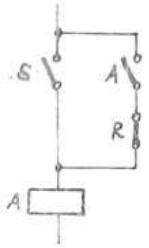


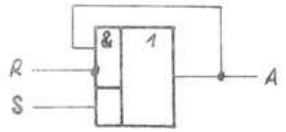
Programm:

LE	S
OA	A
UEC	R
TA	A

6.2.3. S-Flip-Flop

≙ Speicher dominierend setzen
Schaltung:

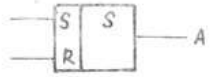




Wertetabelle:

S	R	A
L	L	Zust. bleibt erhalten
L	H	L
H	L	H
H	H	H

Symbol:

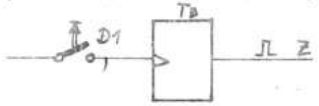


Programm:

LA	A
UEC	R
OE	S
TA	A

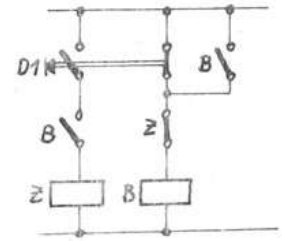
6.3. Monostab. Multivibrator

Der mon. Multiv. soll bei Anlegen einer L/H-Flanke einen Impuls definierter Länge abgeben. Diese Länge soll der Zykluszeit (≈ 18 ms bei 4 k Worten) entsprechen.



Der nächste Impuls soll erst bei der nächsten L/H-Flanke abgegeben werden, unabhängig davon wie lange D1 betätigt war.

Schaltung:



Programm:

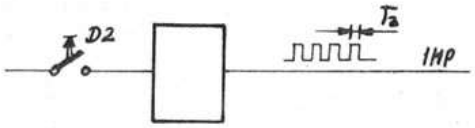
LEC	D1
OA	B
UAC	Z
TA	B
UE	D1
TA	Z

Hinweis: B ist als Zwischenspeicher zu projektieren. Z nur dann, wenn es nicht extern benötigt wird. Der Impuls ist an Z abfragbar.

Anwendung: Die Schaltung schafft definierte Verhältnisse an einem Eingang (Unabhängigkeit von der Betätigungsdauer von D1). Sie ist bevorzugt bei Zählschaltungen einzusetzen, damit die ständig ablaufenden Programmzyklen der Steuerung keine Fehlzählungen verursachen.

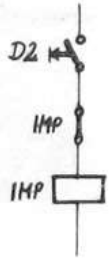
6.4. Astabiler Multivibrator

Der astab. Multiv. soll eine Impulsfolge abgeben, solange der Taster D2 betätigt wird. Die Impulsbreite soll der Zykluszeit entsprechen.



Die Schaltung kann auf eine Selbstunterbrechung zurückgeführt werden, die sich aus dem stetig wiederkehrenden Programmzyklus der PS 2000 ergibt.

Schaltung:



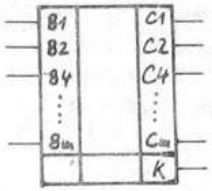
Programm:

LE	D2
UAC	IMP
TA	IMP

IMP ≙ Zwischenspeicher

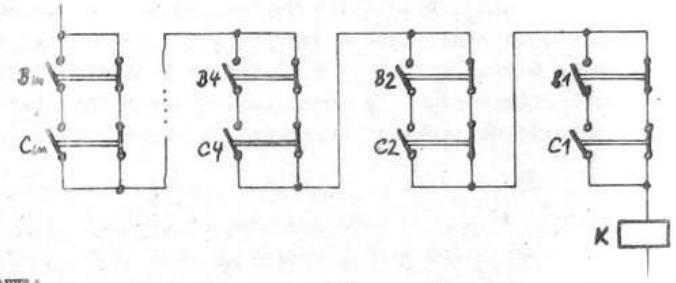
6.5. Vergleicher

Es sollen 2 binärkodierte Größen B und C mit je m Binärstellen miteinander verglichen werden. Bei Gleichheit soll an K H-Signal abgegeben werden.



$K = (B1C1 \vee \overline{B1}\,\overline{C1}) \cdot (B2C2 \vee \overline{B2}\,\overline{C2}) \cdots (BmCm \vee \overline{Bm}\,\overline{Cm})$

Schaltung:

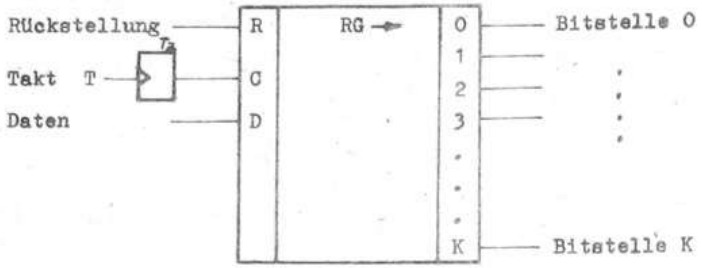


Programm:

LE	Bm		LE	B2	
UE	Cm		UE	C2	
TA	X1	Binärstelle m	TA	X1	
LEC	Bm		LEC	B2	
UEC	Cm		UEC	C2	Binärstelle 2
OA	X1		OA	X1	
TA	X2		UA	X2	
.			TA	X2	
.					
.					
LE	B4		LE	B1	
UE	C4		UE	C1	
TA	X1	Binärstelle 4	TA	X1	
LEC	B4		LEC	B1	Binärstelle 1
UEC	C4		UEC	C1	
OA	X1		OA	X1	X1, X2 ≙ temp. Zwischenspeicher
UA	X2		UA	X2	
TA	X2		TA	K	

6.6. Schieberegister

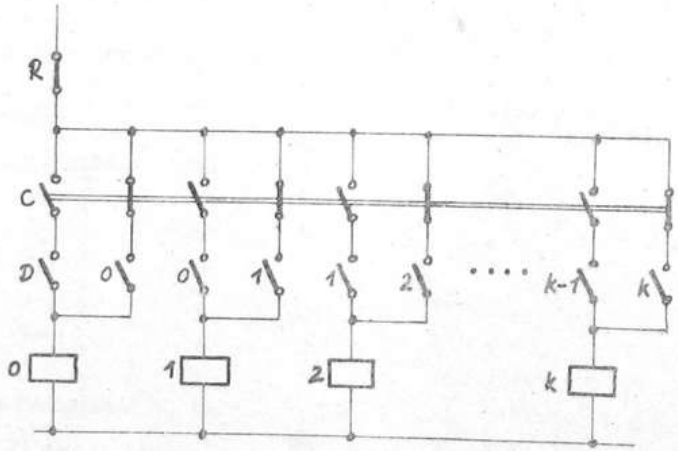
1. Blockschaltbild:



2. Wirkungsweise

Ausgelöst durch die L/H-Flanke des Taktes erzeugt der monostab. Multiv. (vgl. 6.3.) einen Impuls der Länge T_z. Damit wird die an D anstehende Wertigkeit (L oder H) in die Bitstelle 0 übernommen. Gleichzeitig werden die Wertigkeiten der Bitstellen in die nächst höheren übertragen. Rückstellung durch H an R.

3. Schaltung



4. Programm

```
LAC   R   } Rückstellung
BE        }
LAC   T   }
OA    B   } Monostab.
UAC   C   } Multivibr.
TA    B   } nach 6.3.
UA    T   }
TA    C   }
(LA   C)  }
UA    K-1 }
TA    X   } Bitstelle K
LAC   C   }
UA    K   }
OA    X   }
TA    K   }
 .        }
 .        }
 .        }
 .        }
```

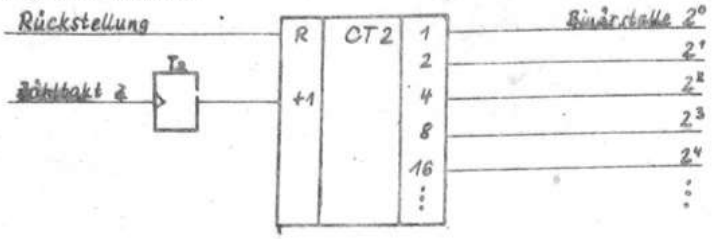
```
 .
 .
 .
LA    C   }
UA    0   }
TA    X   } Bit-stelle 1
LAC   C   }
UA    1   }
OA    X   }
TA    1   }
LA    C   }
UA    D   }
TA    X   }
LAC   C   } Bit-stelle 0
UA    0   }
OA    X   }
TA    0   }
HR
```

5. Hinweise:

1) Programmierreihenfolge einhalten:
   Rückstellung, Mon. Multivibr., Bitstellen von der höchsten zur niedrigsten.
2) H-Register anwenden (vgl. 2.3.3.)
3) R, T und D wurden als Zwischenspeicher angenommen.
   Sie können auch externe Größen des Anwenders sein (E-Befehle anwenden!)
   X ≙ temporärer Zwischenspeicher
   C, B, 0, 1, 2 ... K ≙ Zwischenspeicher

6.7 Binärzähler (vorwärts)

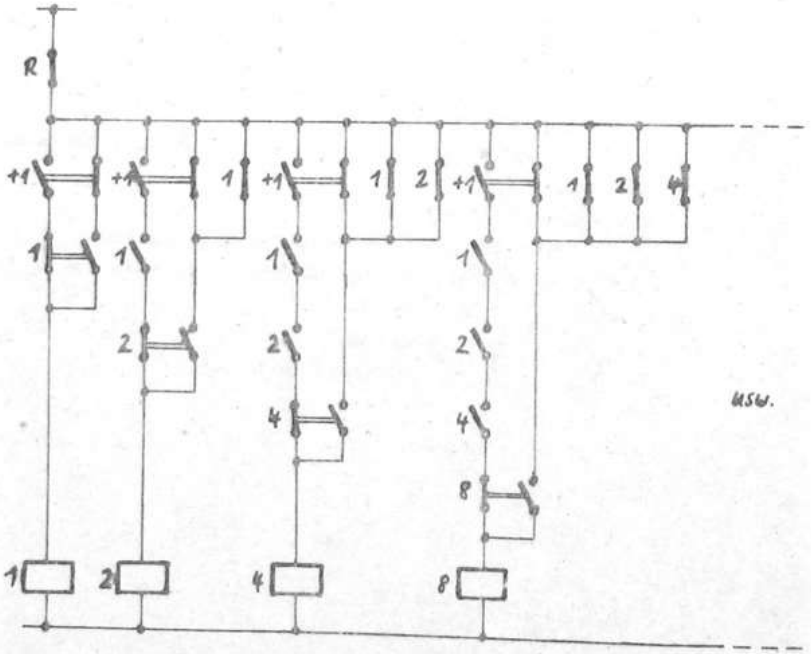
1. Blockschaltbild



2. Wirkungsweise

Ausgelöst durch die L/H - Flanke des Zähltaktes Z erzeugt der monostab. Multivibr. (vgl. 6.3.) einen Impuls der Länge Ts. Mit jedem dieser Impulse erhöht sich der Zählerstand um 1 (Binärcode).
Rückstellung durch H an R.

3. Schaltung



4. Programm

```
LAC   R    } Rückstellung
HS         }
LAC   Z    }
OA    B    } Monostab.
UAC   +1   } Multivibr.
TA    B    }
UA    Z    }
TA    +1   }
.
.
.
(LA   +1)  }
UA    1    }
UA    2    }
UAC   4    }
TA    X    }
LAC   +1   } Binärstelle 2²
OAC   1    }
OAC   2    }
UA    4    }
OA    X    }
TA    4    }
```

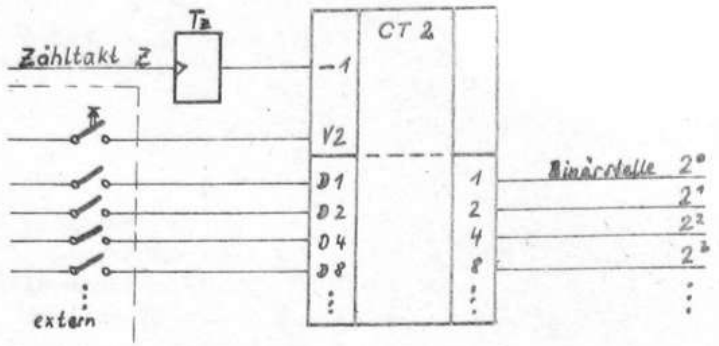
```
LA    +1   }
UA    1    }
UAC   2    }
TA    X    }
LAC   +1   } Binärstelle 2¹
OAC   1    }
UA    2    }
OA    X    }
TA    2    }
LA    +1   }
UAC   1    }
TA    X    } Binärstelle 2°
LAC   +1   }
UA    1    }
OA    X    }
TA    1    }
HR
```

5. Hinweise

1) Programmierreihenfolge einhalten:
   Rückstellung, Mon. Multivibrator, Binärstellen von der höchsten zur niedrigsten.
2) H-Register anwenden
3) R, Z wurden als Zwischenspeicher angenommen. Sie können auch externe Größen des Anwenders sein (E-Befehle anwenden!).
   X ≙ temporärer Zwischenspeicher
   +1, B, 1, 2, 4, 8 ... ≙ Zwischenspeicher
4) Der Binärzähler ist beliebig erweiterbar. Das formale Bildungsgesetz ist aus der Schaltung erkennbar. Die Wortanzahl erhöht sich mit jeder weiteren Stufe um 2 pro Stufe.

6.8 Binärzähler, rückwärts, mit externer Voreinstellung

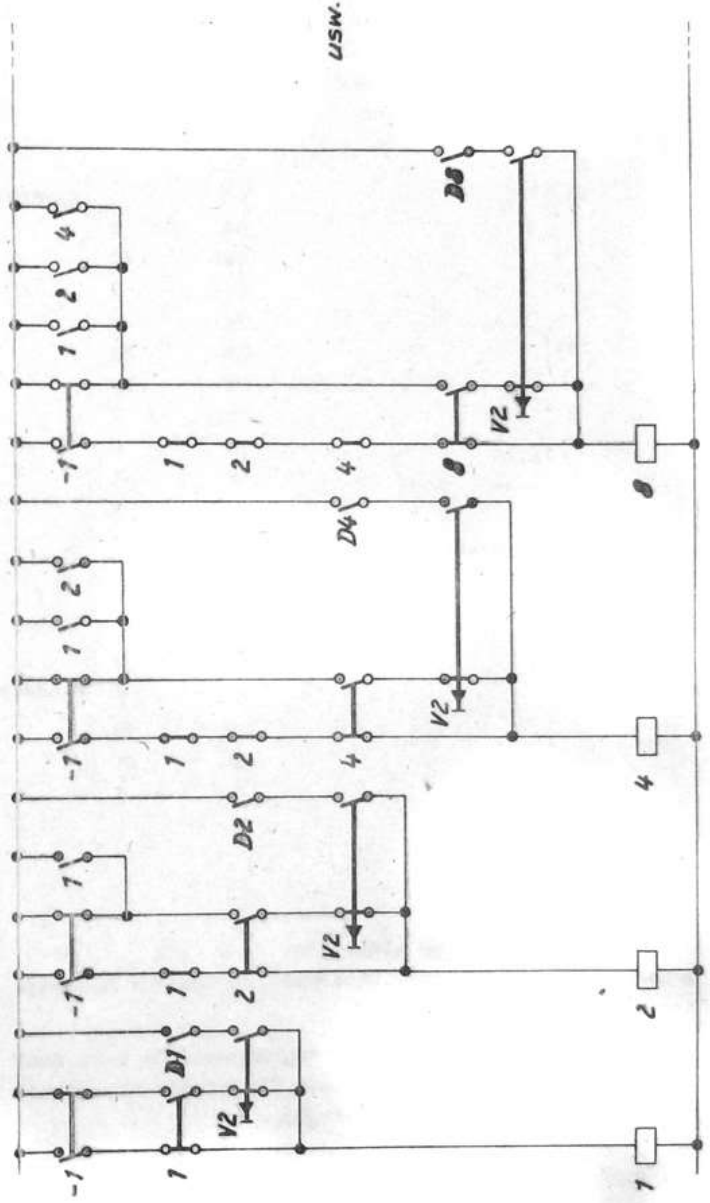
1. Blockschaltbild



2. Wirkungsweise

Mit einem H an V2 (Betätigen des Tasters) werden die voreingestellten Werte in den Zähler übernommen.
Ausgelöst durch die L/H - Flanke des Zähltaktes Z erzeugt der monostab. Multivibr. (vgl. 6.3) einen Impuls der Länge Tz. Mit jedem dieser Impulse erniedrigt sich der Zählerstand um 1 (Binärcode).

3. Schaltung



4. Programm

LAC	Z	Monostab. Multivibrator	LA	-1	Binärstelle 2^1
OA	B		UAC	1	
UAC	-1		UAC	2	
TA	B		TA	X	
UA	Z		LAC	-1	
TA	-1		OA	1	
.			UA	2	
.			UEC	V2	
.			OA	X	
.			TA	X	
(LA	-1)	Binärstelle 2^2	LE	V2	
UAC	1		UE	D2	
UAC	2		OA	X	
UAC	4		TA	2	
TA	X		LA	-1	Binärstelle 2^4
LAC	-1		UAC	1	
OA	1		TA	X	
OA	2		LAC	-1	
UA	4		UA	1	
UEC	V2		UEC	V2	
OA	X		OA	X	
TA	X		TA	X	
LE	V2		LE	V2	
UE	D4		UE	D1	
OA	X		OA	X	
TA	4		TA	1	

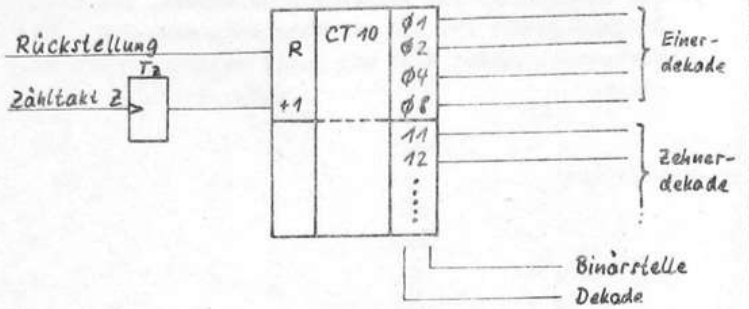
5. Hinweise

1. Programmierreihenfolge einhalten
   Monostab. Multivibrator, Binärstellen von der höchsten zur niedrigsten
2. Z wurden als Zwischenspeicher angenommen. Es kann auch externe Größe des Anwenders sein (E-Befehle anwenden!)
   V2, D1, D2, D4 ... ≙ Prozeßeingänge
   X ≙ temp. Zwischenspeicher
   -1, B, 1, 2, 4, 8 ≙ Zwischenspeicher

3. Der Binärzähler ist beliebig erweiterbar. Das formale Bildungsgesetz ist aus der Schaltung erkennbar. Die Wortanzahl erhöht sich mit jeder weiteren Stufe um 2 pro Stufe.

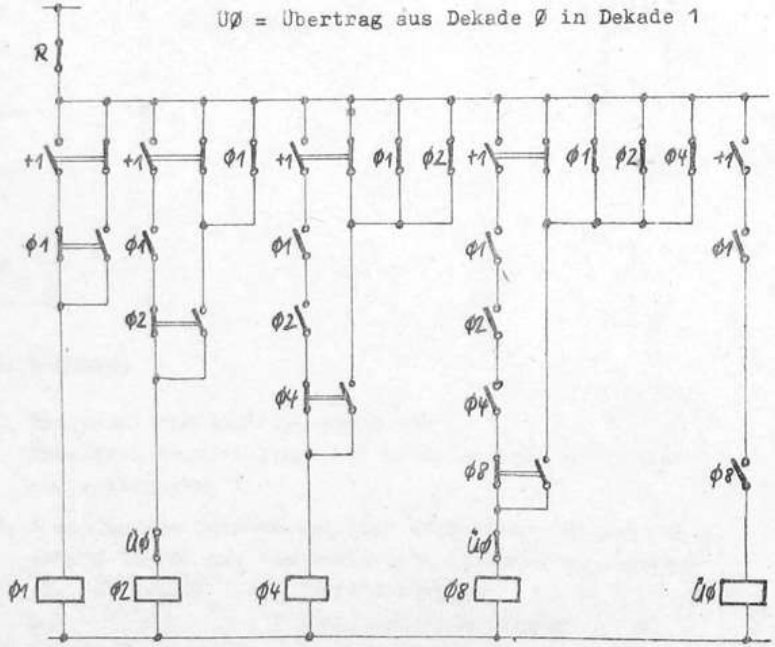
6.9 Binärkodierter Dezimalzähler (vorwärts)

1. Blockschaltbild



2. Wirkungsweise
wie Binärzähler nach 6.7, jedoch Übertrag nach Zählerstand 9 in die nächste Dekade und Rückstellung der auslösenden Dekade.

3. Schaltung der Einerdekade



4. Programm

LAC	Z	Monostab. Multivibrator	LA	+1	Binärstelle 2^2
OA	B		UA	Ø1	
UAC	+1		UA	Ø2	
TA	B		UAC	Ø4	
UA	Z		TA	X	
TA	+1		LAC	+1	
LAC	R	Rückstellung	OAC	Ø1	
NS			OAC	Ø2	
⋮		Höhere Dekaden	UA	Ø4	
			OA	X	
LA	+1	Übertrag aus Dekade Ø in Dekade 1	TA	Ø4	
UA	Ø1		LA	+1	Binärstelle 2^1
UA	Ø8		UA	Ø1	
TA	ÜØ		UAC	Ø2	
LA	+1	Binärstelle 2^3	TA	X	
UA	Ø1		LAC	+1	
UA	Ø2		OAC	Ø1	
UA	Ø4		UA	Ø2	
UAC	Ø8		OA	X	
TA	X		UAC	ÜØ	
LAC	+1		TA	Ø2	
OAC	Ø1		LA	+1	Binärstelle 1^0
OAC	Ø2		UAC	Ø1	
OAC	Ø4		TA	X	
UA	Ø8		LAC	+1	
OA	X		UA	Ø1	
UAC	ÜØ		OA	X	
TA	Ø8		TA	Ø1	
			HR		

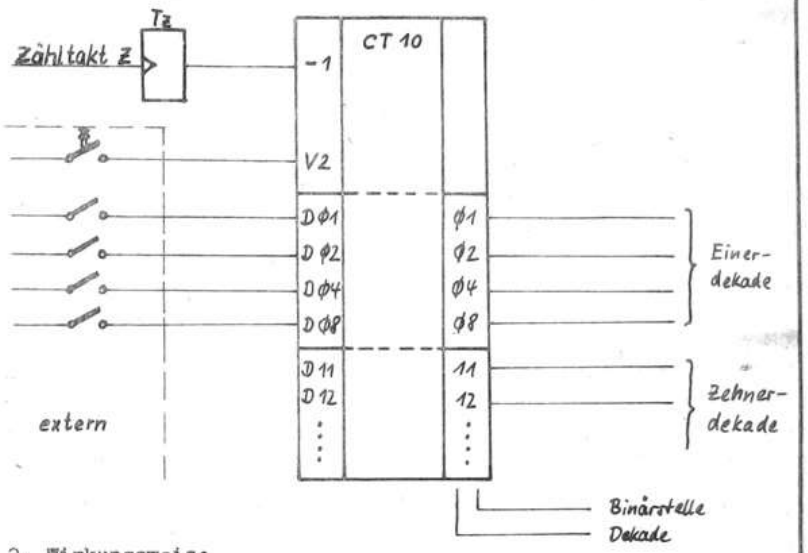
5. Hinweise

1. Programmierreihenfolge einhalten:
Monostab, Multivibr. Rückstell., höchste Dekade (Binärstellen in absteigender Reihenfolge), niedere Dekaden (Übertrag, Binärstellen in absteigender Reihenfolge.)

2. H-Register anwenden
3. R, Z wurden als Zwischenspeicher angenommen (auch als externe Größen möglich)
   X ≙ temp. Zwischenspeicher
   +1, B, Ø1, Ø2, Ø4, Ø8, ÜØ .... ≙ Zwischenspeicher
4. Die anderen Dekaden werden analog programmiert
   z. B. Zehnerdekade ⇒ Es ist zu ersetzen: +1 → ÜØ
   ÜØ → Ü1
   Ø1 → 11
   Ø2 → 12 usw.

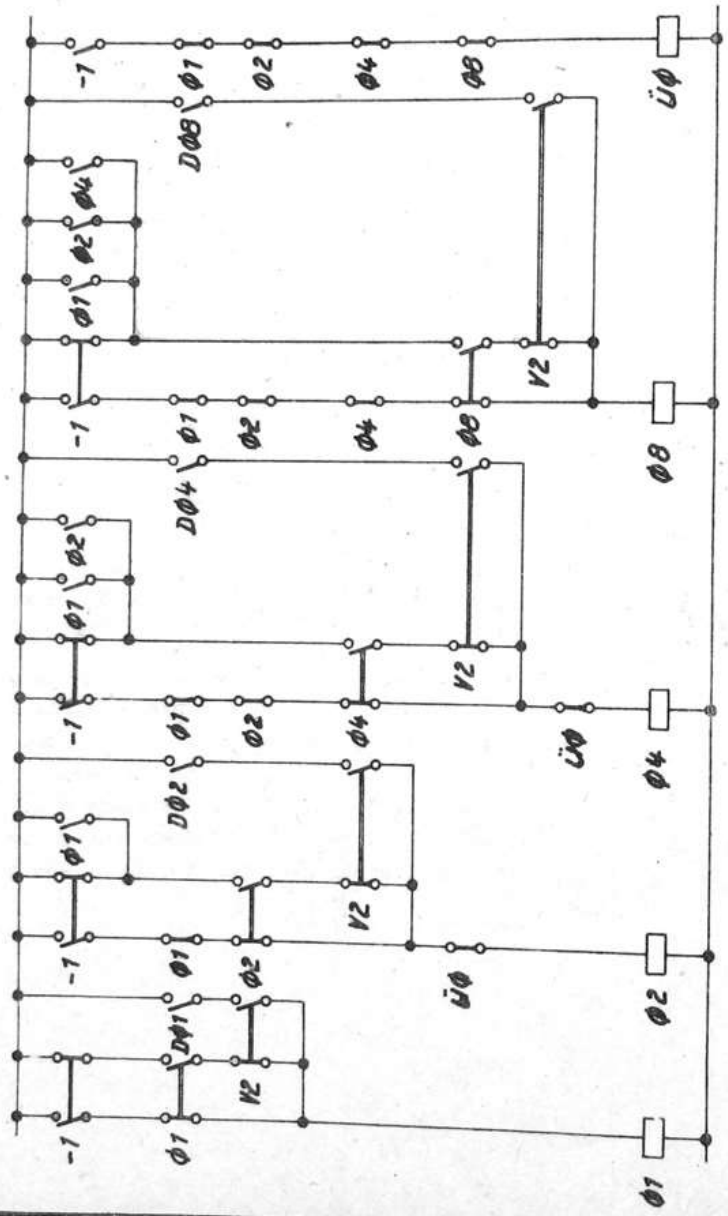
6.10 Binärkodierter Dezimalzähler (rückwärts) mit externer Voreinstellung

1. Blockschaltbild



2. Wirkungsweise
   Mit einem H an V2 werden die voreingestellten Werte in den Zähler übernommen. Ausgelöst durch die L/H - Flanke des Zähltaktes Z erzeugt der monostab. Multivibr. einen Impuls der Länge Tz. Mit jedem dieser Impulse erniedrigt sich der Zählerstand um 1. Nach dem Zählerstand Ø der Einerdekade erfolgt ein Übertrag. Der Stand der Zehnerdekade wird dabei um 1 verringert und die Einerdekade auf 9 gestellt.

3. Schaltung der Einerdekade



4. Programm

LAC	Z		⋮		Binärstelle 2^2
OA	B		⋮		
UAC	-1	Monostab. Mult.	LA	-1	
TA	B		UAC	Ø1	
UA	Z		UAC	Ø2	
TA	-1		TA	X	
⋮		Höhere Dekaden	LAC	-1	
(LA	-1		OA	Ø1	
UAC	Ø1		UA	Ø2	Binärstelle 2^1
UAC	Ø2	Übertrag aus	UEC	V2	
UAC	Ø4	Dekade Ø in	OA	X	
UAC	Ø8	Dekade 1	TA	X	
TA	Üø		LE	DØ2	
LA	-1		UE	V2	
UAC	Ø1		OA	X	
UAC	Ø2		UAC	Üø	
UAC	Ø4		TA	Ø2	
UAC	Ø8		LA	-1	
TA	X		UAC	Ø1	
LAC	-1		TA	X	
OA	Ø1		LAC	-1	
OA	Ø2		UA	Ø1	
OA	Ø4	Binärstelle 2^3	UEC	V2	Binärstelle 2^0
UA	Ø8		OA	X	
UEC	V2		TA	X	
OA	X		LE	DØ1	
TA	X		UE	V2	
LE	DØ8		OA	X	
UE	V2		TA	Ø1	
OA	X				
TA	Ø8				

5. Hinweise

1. Programmierreihenfolge einhalten:
   Monostab. Multiv., höchste Dekade (Binärstellen in absteigender Reihenfolge); nächst niedere Dekaden ⟶

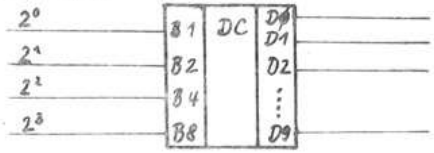
(Übertrag, Binärstellen in absteigender Reihenfolge).

2. Z wurde als Zwischensp. angenommen. Es kann auch ext. Größe des Anwenders sein (E-Befehle anwenden!)
   V2, DØ1, DØ2, DØ4, DØ8, ... ≙ Prozeßeingänge
   X ≙ temporärer Zwischensp.
   -1, B, Ø1, Ø2, Ø4, Ø8, Ü Ø... ≙ Zwischenspeicher

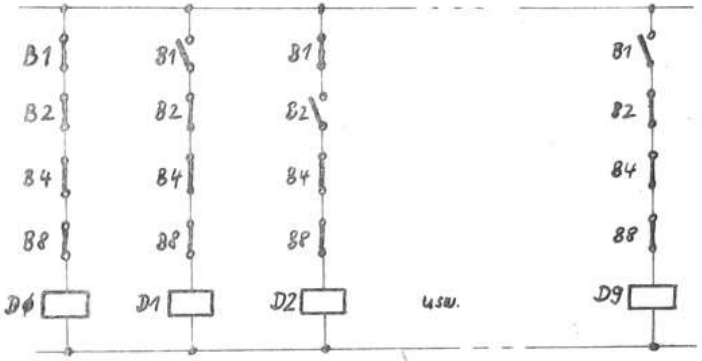
3. Die anderen Dekaden werden analog programmiert,
   z. B. Zehnerdekade: Es ist zu ersetzen: -1 ⟶ ÜØ
   ÜØ ⟶ U1
   Ø1 ⟶ 11
   Ø2 ⟶ 12 usw.

6.11 Dekoder BCD - Dezimal

1. Blockschaltbild



2. Schaltung



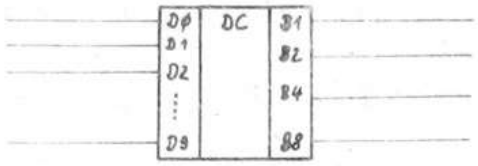
3. Programm

```
LEC  B1
UEC  B2
UEC  B4
UEC  B8
TA   DØ      } Ø

LE   B1
UEC  B2
UEC  B4
UEC  B8
TA   D1      } 1

LEC  B1
UE   B2
UEC  B4
UEC  B8
TA   D2      } 2

LE   B1
UEC  B2
UEC  B4
UE   B8
TA   D9      } 9
```

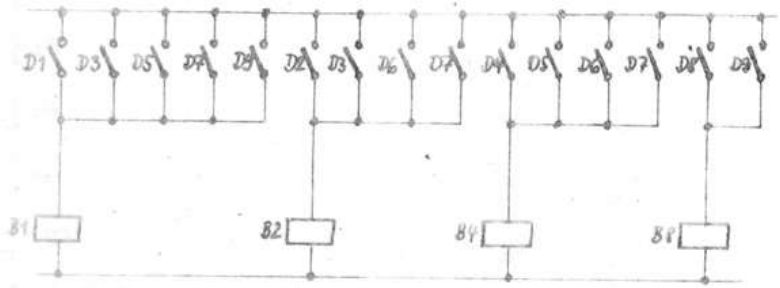
4. Hinweise
   Nur diejenigen Dezimalstellen programmieren, die benötigt werden. die Reihenfolge der Teilkomplexe ist beliebig.

6.12 Dekoder Dezimal - BCD

1. Blockschaltbild



2. Schaltung



3. Programm

```
LE  D1
OE  D3
OE  D5
OE  D7
OE  D9
TA  B1
LE  D2
OE  D3
OE  D6
OE  D7
TA  B2
LE  D4
OE  D5
OE  D6
OE  D7
TA  B4
LE  D8
OE  D9
TA  B8
```

4. Hinweise

Nur diejenigen Binärstellen Programmieren, die benötigt werden. Die Reihenfolge der Teilkomplexe ist beliebig. Der Eingang für die Dezimalzahl Ø kann entfallen.

7. Festwertspeicheränderungen

7.1. Löschen und neu programmieren

Der Inhalt der pROMS kann geändert werden, in dem diese Speicher mit UV-Licht bestrahlt werden. Anschließend ist eine erneute Programmierung möglich.

7.2. Überprogrammieren

Die mit HIGH belegten Stellen des Bitmusters sind nicht veränderbar (UV-Bestrahlung ausgenommen). Die mit LOW belegten Stellen nehmen jedoch bei Überprogrammieren mit HIGH diesen Wert an. Damit werden in einzelnen Fällen Programm – Änderungen möglich, ohne das Programm erst löschen zu müssen. Dabei ist es gleich, ob es sich um den Befehl oder die Adresse handelt.

Folgende Überschreibungen sind möglich:

Istwert (hex.)	Sollwert möglich (hex.)
0	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
1	1, 3, 5, 7, 9, B, D, F
2	2, 3, 6, 7, A, B, E, F
3	3, 7, B, F
4	4,5,6,7, C,D,E,F
5	5, 7, D, F
6	6,7, E,F
7	7, F
8	8, 9, A, B, C, D, E, F
9	9, B, D, F
A	A, B, E, F
B	B, F
C	C, D, E, F
D	D, F
E	E, F
F	F

Tabelle 1: Befehlsumfang und Kodierung

Befehlsart	Befehl	Name	Kode	Bedeutung
Transportbefehle	LE	NAME	4 X X X	Laden eines Wertes (wahr) aus dem E-Speicher } in den Akkumulator (AC)
	LEC	NAME	5 X X X	Laden eines Wertes (kompl.) aus dem E-Speicher } in den Akkumulator (AC)
	LA	NAME	C X X X	Laden eines Wertes (wahr) aus dem A-Speicher } in den Akkumulator (AC)
	LAC	NAME	D X X X	Laden eines Wertes (kompl.) aus dem A-Speicher } in den Akkumulator (AC)
	TEC	NAME	6 X X X	Transport des Akku-Inhaltes in den E-Speicher (kompl.) 1)
	TA	NAME	E X X X	Transport des Akku-Inhaltes in den A-Speicher (wahr)
Verknüpfungsbefehle	UE	NAME	Ø X X X	UND-Verknüpfung eines Wertes (wahr) aus dem E-Speicher } mit dem AC-Inhalt
	UEC	NAME	1 X X X	UND-Verknüpfung eines Wertes (kompl.) aus dem E-Speicher } mit dem AC-Inhalt
	UA	NAME	8 X X X	UND-Verknüpfung eines Wertes (wahr) aus dem A-Speicher } mit dem AC-Inhalt
	UAC	NAME	9 X X X	UND-Verknüpfung eines Wertes (kompl.) aus dem A-Speicher } mit dem AC-Inhalt
	OE	NAME	2 X X X	ODER-Verknüpfung eines Wertes (wahr) aus dem E-Speicher } mit dem AC-Inhalt
	OEC	NAME	3 X X X	ODER-Verknüpfung eines Wertes (kompl.) aus dem E-Speicher } mit dem AC-Inhalt
	OA	NAME	A X X X	ODER-Verknüpfung eines Wertes (wahr) aus dem A-Speicher } mit dem AC-Inhalt
	OAC	NAME	B X X X	ODER-Verknüpfung eines Wertes (kompl.) aus dem A-Speicher } mit dem AC-Inhalt
Zusatzbefehle	HR	–	F Ø Ø Ø	Rücksetzen des H-Registers } Register für die Hauptfunktion
	HS	–	F Ø Ø 1	Aktivieren des H-Registers } Register für die Hauptfunktion
	DR	–	F Ø Ø 2	Rücksetzen des D-Registers } Register für direkte Eingabe oder Ausgabe
	DS	–	F Ø Ø 3	Setzen des D-Registers } Register für direkte Eingabe oder Ausgabe
	END	–	F Ø Ø 4	Programmende
	KO	–	F 7 7 7	Keine Operation – Vollwert
	NUL	–	Ø Ø Ø Ø	Leerwort (nur bedingt anwendbar, vgl. 2.3.2)

Hinweise:
NAME ≙ Name der Eingangs- oder Ausgangsinformation
X X X ≙ oktale Adresse der Eingangs- oder der Ausgangsinformation
– ≙ keine Angabe
1) ≙ TEC-Befehl nur bedingt anwendbar (vgl. 3.5)

62

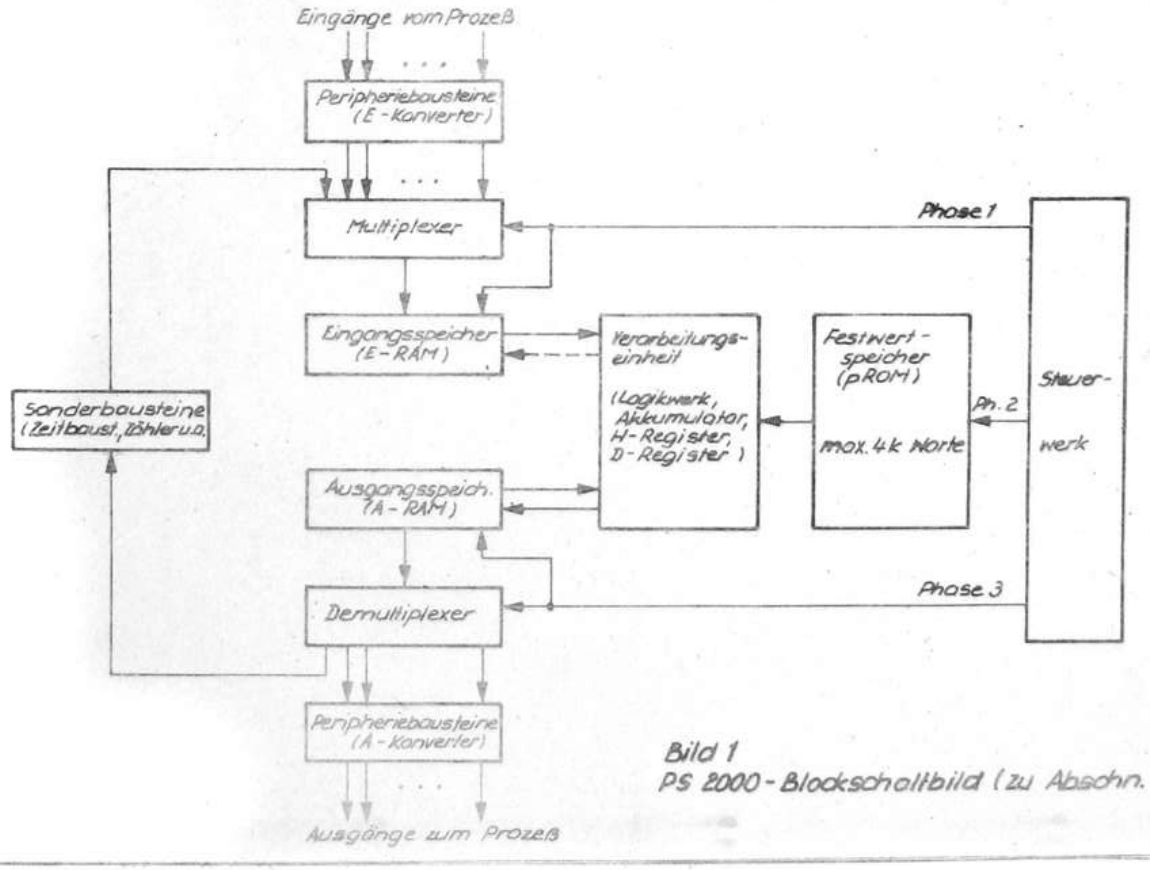


Bild 1
PS 2000 - Blockschaltbild (zu Abschn. 1.1.)

63