TRIUMPH ADLER INFORMATIONS SYSTEME

Allgemeine Einführung zum Mikroprogramm

Das Mikroprogramm unterteilt sich in Steuer- und Interpretativ-Mikroprogramm.
Das Steuer-Mikroprogramm übernimmt alle Verwaltungsaufgaben und kontrolliert die Maschinenzustände, hierzu gehören:

Einschaltprogramm
Unterbrechungsprogramm
Betriebssystemroutinen

Das Interpretativ-Mikroprogramm verwirklicht die vorgeschlagene

TRIASS * - Sprache mit dazugehörigen Ablaufroutinen.

Es ist in zwei Abschnitte zu unterteilen:

Aufrufphase
Befehlsausführungsphase

Obwohl Steuer- und interpretatives Mikroprogramm eine Einheit bilden, sollten die Programme so modular aufgebaut sein, daß Ausbau und Änderungen in einem Teil sich auf andere Programmteile nicht auswirken. Diese Forderung steht im Widerspruch zur optimalen Programmierung (geringe Befehlsanzahl). So muß für jedes Mikroprogramm ein Kompromiß gefunden werden, der nach den speziellen Aufgabenbereich ausgelegt ist.

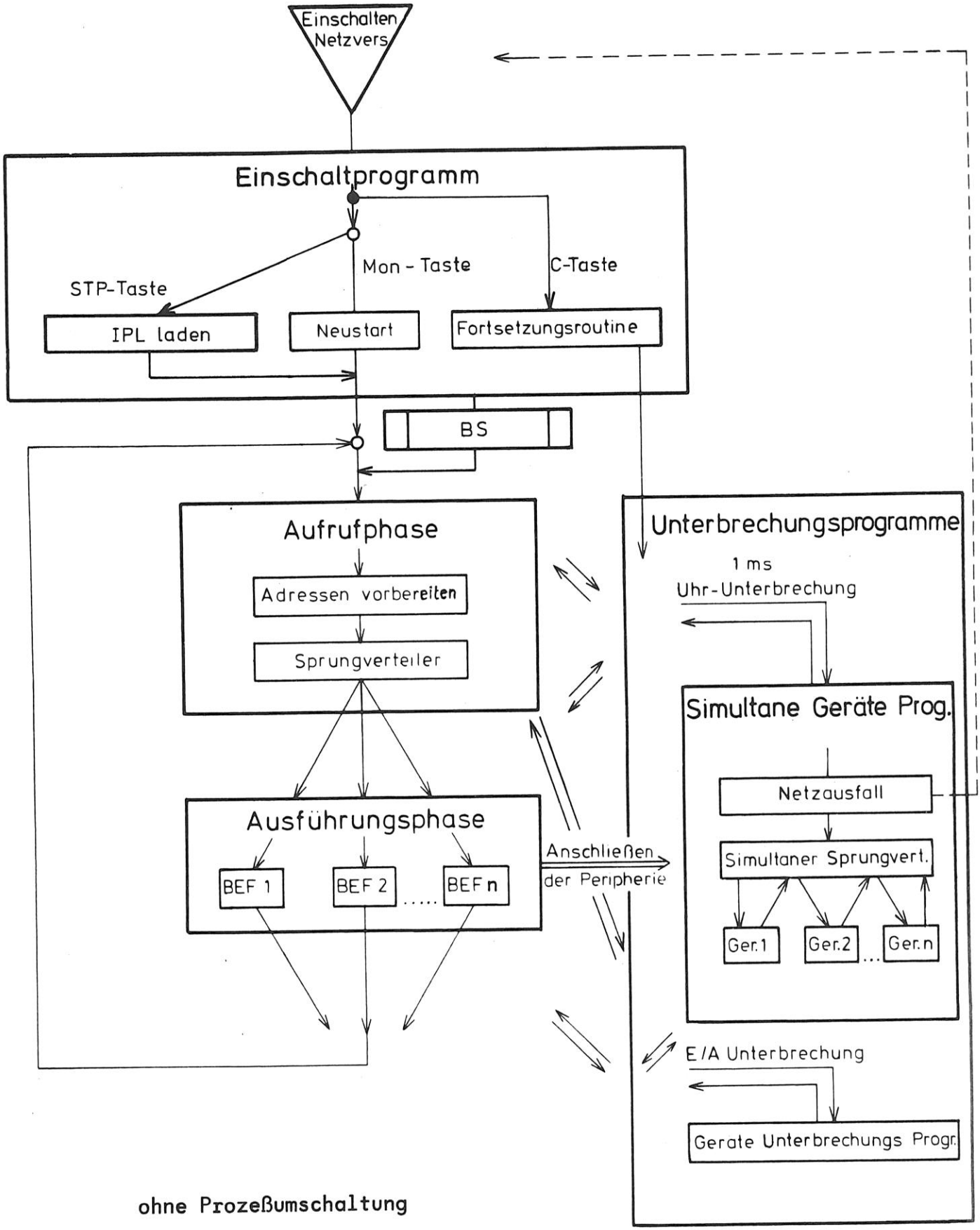
* TRIASS - TRIUMPH - ASSEMBLER

Es sind dabei folgende Faktoren u.a. zu beachten:

geringe Anzahl von Mikro-Befehlen
komplexe Anwender-Sprache
sichere Peripherie
einfache Ausbaufähigkeit der Peripherie
bequeme Bedienung

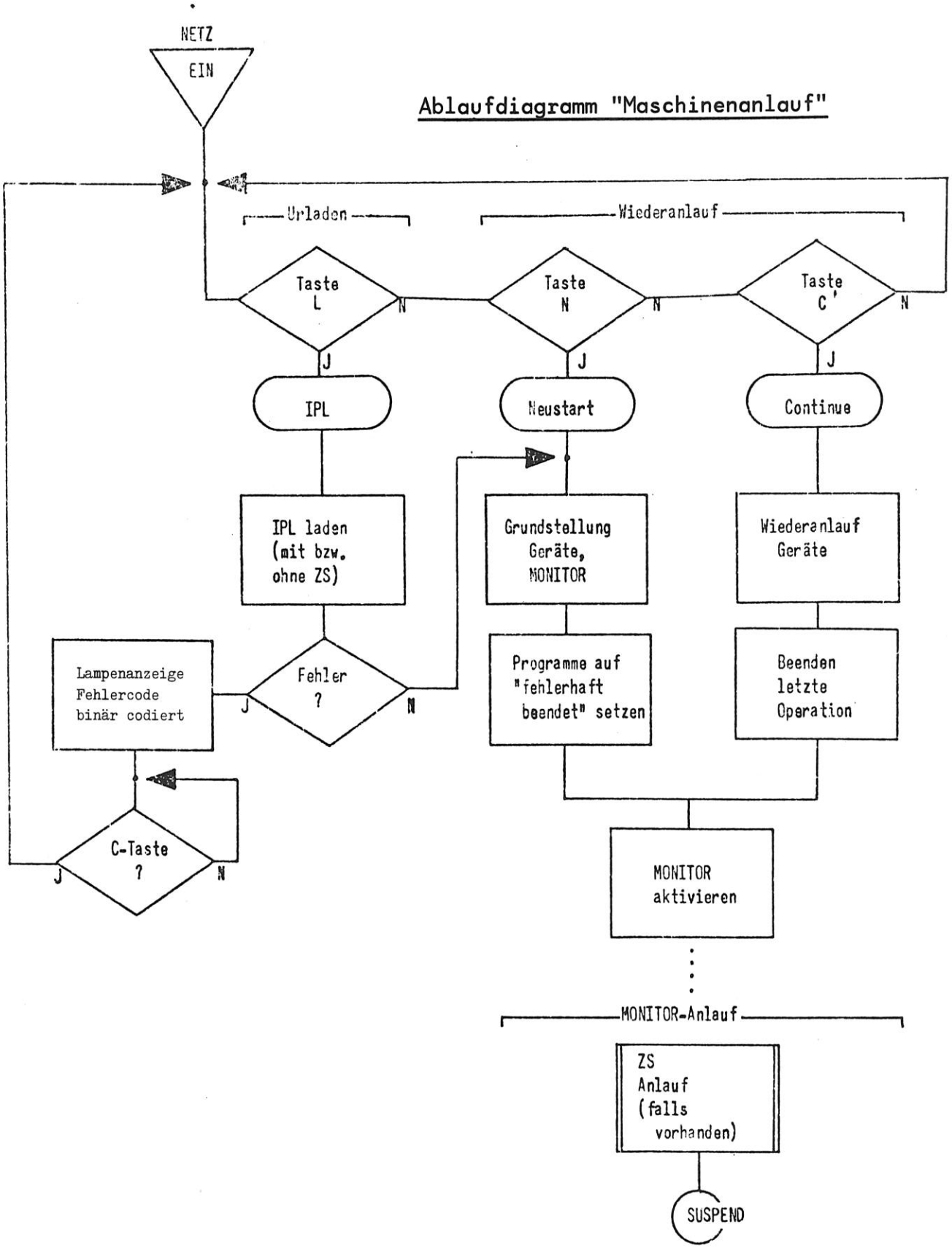
Die TRIASS-Befehle der Datenstation sind nach den genannten Gesichtspunkten realisiert worden, wobei eine Unterteilung in Standard- und Peripherie-Mikroprogramm entsprechend der Aufteilung im Standardbefehlssatz der TRIASS-Sprache vorgenommen wurde.
Im Standard-Mikroprogramm sind alle TRIASS-Befehle verwirklicht, die zur Grundkonfiguration der Datenstation gehören.
Charakterristisch für das Standard-Mikroprogramm ist eine Optimierung durch weitgehende Verschachtelung der interpretierten TRIASS-Befehle.
Die Peripherie-Geräteprogramme hingegen wurden so modular gehalten, daß eine beliebige Gerätezusammenstellung möglich ist. Ein jedes Geräteprogramm ist physikalisch in einem zusammenhängenden Speicherbereich untergebracht. Dieser Bereich wird Gerätemodul genannt.
Der Speicher ist für 32 K Festspeicher und 32 K Lebendspeicher organisiert, wobei von den 32 K Lebendspeicher "3 K-Byte" für die interne Verwaltung benötigt werden.
Diese "3 K" Lebendspeicher sind in "1 K" Scratchpad und "1 K" Arbeitspuffer für die verschiedenen Geräte, sowie "1 K" für Prozeßverwaltung unterteilt.

Mikroprogrammablauf



ohne Prozeßumschaltung

Ablaufdiagramm "Maschinenanlauf"



Anlaufroutine bis "TSP"-Lampe

0. 0. 0. 0	0. 12. 0. 4 0000 1100 0000 0100 (A)→S+N N = 0.0.0.4	hardewaremäßig kommt 4 auf R-Bus
0. 0. 0. 4	1. 4. 7. 15 0001 0100 0111 1111 BR→(P+N) N = 0.0.7.14	In Zelle 0.0.7.14 steht der Inhalt 1.0.14.2
1. 0.14. 2	8. 0. 1. 15 1000 0000 0001 1111 N→A N = 0.0.1.15	[1.15→A]
1. 0.14. 4	0. 12. 9. 8 0000 1100 1001 1000 (A)→S+N N = 0.0.9.8	1.15→8.0.9.8 0.0.1.15
1. 0.14. 6	8. 4. 0. 6 1000 0100 0000 0110 $(N)_Z$→A N = 0.0.0.6	6→A
1. 0.14. 8	0. 15. 15. 0 0000 1111 1111 0000 (A)→S+N N = 0.3.15.0	6→8.3.15.0 0.0.0.6
1. 0.14.10	8. 0. 0. 0 1000 0000 0000 0000 N→A N = 0.0.0.0	Löschen Akku
1. 0.14.12	0. 15. 13. 10 0000 1111 1101 1010 (A)→S+N N = 0.3.13.10	Löschen 8.0.13.10 0.0.0.0
1. 0.14.14	8. 0. 8. 0 1000 0000 1000 0000 N→A N = 0.0.8.0	0.0.8.0→A
1. 0.15. 0	0. 15. 13. 4 0000 1111 1101 0100 (A)→S+N N = 0.3.13.4	0.0.8.0→8.3.13.4 0.0.8.0
1. 0.15. 2	8. 15. 13. 10 1000 1100 1001 1010 (S+N)→A N = 0.0.9.10	8.0.9.10 0.0.8.0 → A Diese Zelle wird durch BS geladen.
1. 0.15. 4	9. 0. 11. 0 1001 0000 1011 0000 (A) N→A N = 0.0.11.0	0000 0000 1000 0000 = A ^ 00 1011 0000 = N 0000 0000 1000 0000 = A entspricht der Lampe "TSP"
1. 0.15. 6	12. 8. 9. 9 1100 1000 1001 1001 IO((S+N))$_B$↔A N = 0.0.9.8	8:0.9.8 0000 0000 | 0001 | 1 | 111 Zeile7 Output I/O Kartenadr.
1. 0.15. 8	12. 2. 0. 5 1100 0010 0000 0101 Sonderb. N = 0.2.0.4	(A) Rechtsshift um 1 bit Netzausfall→C

Dieses Programm sorgt für den ordnungsmäßigen Ablauf der Anlage nach dem Einschalten, nach einem Netzausfall und nach einem Ladevorgang. Vom Operator können drei verschiedene Routinen angewählt werden.
Jeder Routine ist eine Taste zugeordnet.

A	Fortsetzungsroutine (continue)	C	- Taste
B	Neustartroutine	MON	- Taste
C	IPL * - Routine	STP	- Taste

A Fortsetzungsroutine

Sie wird nach einem Netzausfall angesprochen. Dieses Programm prüft und regeneriert alle notwendigen Parameter für die Fortsetzung des Programmes (z.B. Parameter von gestörten Peripherieausführungen). Die Fortsetzungsroutine durchläuft sämtliche Geräte-Anlaufprogramme, beginnend mit dem Tastatur-Anlaufprogramm.
Bei der Mikroprogrammierung ist auf die Netzausfallfrage in regelmäßigen Zeitabständen zu achten. Die simultanen Geräteanlaufroutinen starten erst mit der Freigabe der 1-ms-Uhr. Zu beachten ist, daß während die Geräte wieder anlaufen, ein erneuter Netzausfall eintreten kann.
Mit der C-Taste erfolgt die Fortsetzung des Anwender-Programmes an der unterbrochenen Stelle.

* IPL - Initial Program Loading (Einleitendes Programmladen)

B Neustartroutine

Sie bringt die Anlage in eine definierte Grundstellung (d.h. Befehlszähler, interne Arbeitszellen, Statusbyte, Simultanzeiger usw., werden mit Anfangswerten geladen). Anschließend wird in das Betriebssystem verzweigt.

C IPL

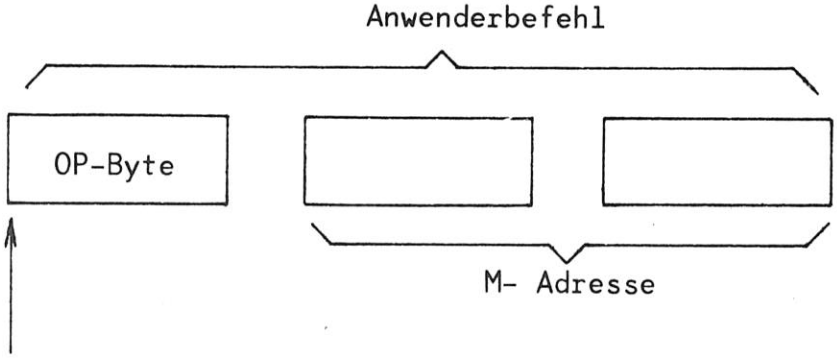
Die STP-Taste bewirkt die Verzweigung in eine spez. Laderoutine (IPL" Laden), danach Verzweigung in die Umlaufroutine und in das Betriebssystem.

D Internfehlerroutine

Auftretende Internfehler in Anwenderprozessen, werden als Systemmeldungen in der Systemzeile abgebildet. Die Meldungen sind im Nachrichtenkopf mit "SYS" gekennzeichnet und müssen vom Bediener quittiert werden.
Für Serviceunterstützung wird zusätzlich in der SC-Zelle XIFMIA (8.0.7.2), die Adresse des ausführenden Mikrobefehles abgespeichert, der den Internfehler erkannte.
Internfehler im Monitorprogramm können nur mit Hilfe des Testtableaus lokalisiert werden.
Die Nummer des Internfehlers steht in XINTF (8.0.6.14) und die absolute Adresse des fehlerhaften Befehls in XICF (8.0.7.0).

Die Aufrufphase hat die Aufgabe, jeden Anwenderbefehl aufzurufen, um ihn für die Verarbeitung durch das Mikro aufzubereiten.
Wenn der Anwenderbefehl genau seiner Befehlsgruppe zugeordnet ist, dann verzweigt das Programm in die Befehlsausführungsphase. Die Einzelschritte der Aufrufphase sind nachstehend aufgelistet.

a) Instruction-Counter und Byte-Counter werden auf das OP-Byte des Anwenderbefehls gesetzt.
OP-Byte wird nach Akku gebracht.



Instruction-Counter bleibt während der Abarbeitung des gesamten Befehls hier stehen.
Byte-Counter zählt nach Abarbeitung jeden bytes um 1 nach rechts weiter.

b) Prüfung ob OP - byte = 0 (NOP)

b1) Prüfung ob Branch - bzw. Registerbefehl (0 - 12)

Prüfung ob allgemeiner Befehl (13)

" " arithmetischer Befehl (14)

" " I/O-Befehl (15)

c) Entscheidung Sprung- oder Registerbefehl

d) Anwenderprogramm - Startadresse nach Akku laden

e) Errechnen der echten Register-Anfangsadresse

f) Prüfung, ob Befehl ohne Adressteil

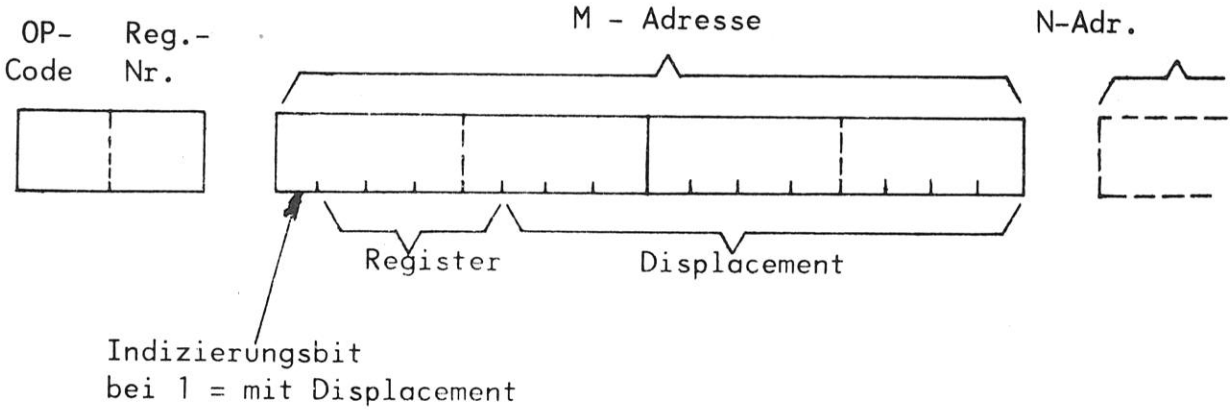
g) Hohes byte der M-Adresse kommt nach Akku

h) Prüfen ob direkt oder indirekt adressiert ist.

i) Register nach Arbeitszelle laden

j) Register + Anwenderprogramm-Anfangsadresse errechnen

k) Inhalt Register + Displacement der hohen M-Adresse errechnen



1) Inhalt der niederen Adresse abspeichern
   Registerinhalt + Displacement-Inhalt abspeichern

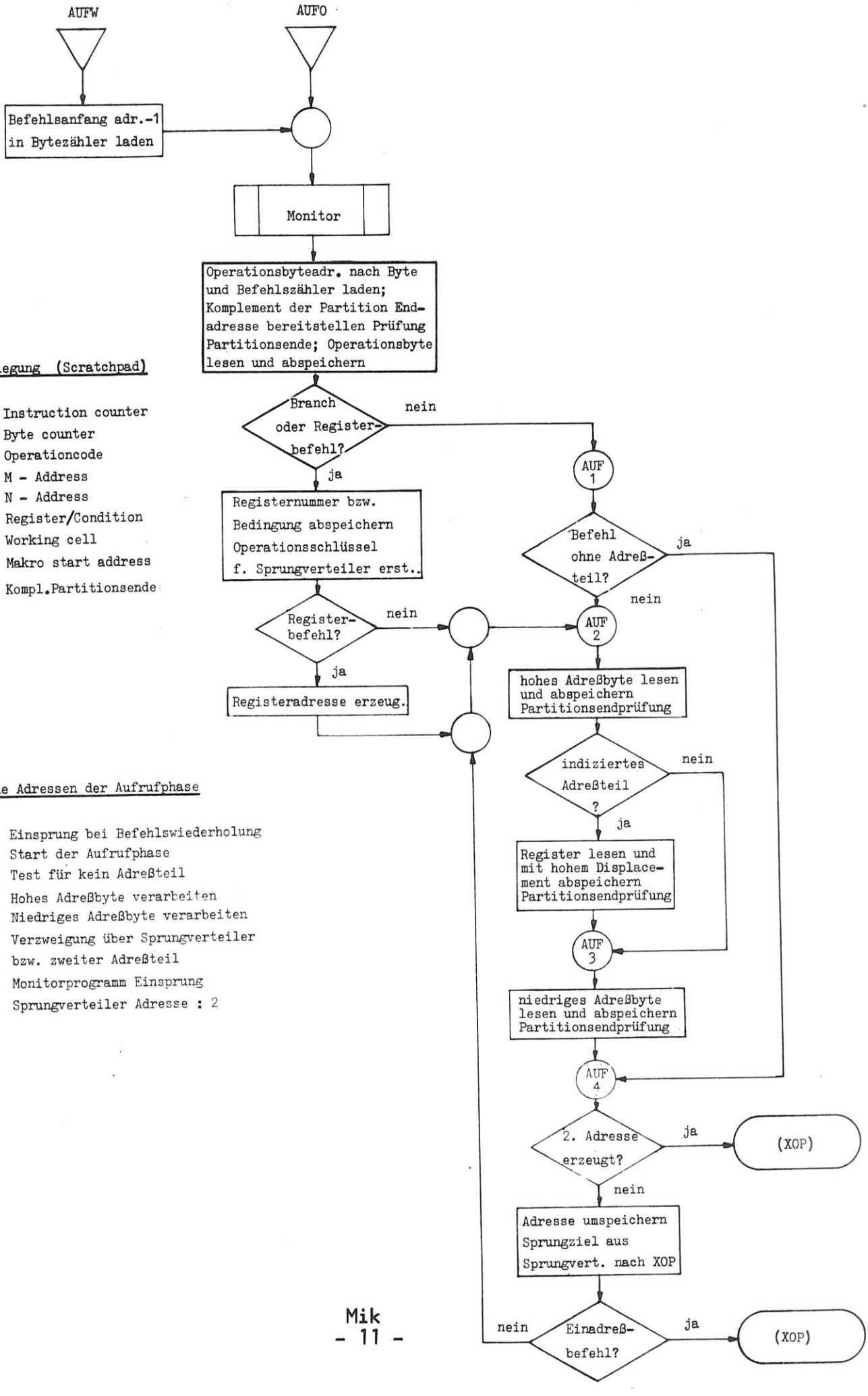
m) Sprungverteileradresse + doppelter OP-Code verweist nach fest vereinbarten Zellen

n) Ist das 1. bit dieser Zellen ungerade, handelt es sich um einen 1-Adressbefehl (nur M-Adresse). Ist das 1. bit dieser Zellen gerade, muß die N-Adresse noch hinzuaddiert werden, anschließend wird der Befehl ausgeführt.
   Im ersten Fall (ungerade) erfolgt sofort der Sprung in die Ausführungsphase.

Befehls - Ausführungsphase

Die Ausführungsphase der TRIASS-Befehle schließt an die Aufrufphase an und führt die eigentliche Bearbeitung durch.
Abgesehen von den Peripherie-Befehlen, die simultan beendet werden, ist nach der Ausführungsphase der TRIASS-Befehl abgeschlossen.
Es folgt der Aufruf des nächsten TRIASS-Befehls.
Die Ausführungsphase eines Befehls kann durch die verschiedenen Interrupt-Einrichtungen unterbrochen werden.

Programm - Ablaufplan : Aufrufphase



Speicherbelegung (Scratchpad)

XIC	=	Instruction counter
XBC	=	Byte counter
XOP	=	Operationcode
XM	=	M - Address
XN	=	N - Address
XRC	=	Register/Condition
XWK	=	Working cell
XMST	=	Makro start address
XPATED	=	Kompl.Partitionsende

Symbolische Adressen der Aufrufphase

AUFW	=	Einsprung bei Befehlswiederholung
AUFO	=	Start der Aufrufphase
AUF1	=	Test für kein Adreßteil
AUF2	=	Hohes Adreßbyte verarbeiten
AUF3	=	Niedriges Adreßbyte verarbeiten
AUF4	=	Verzweigung über Sprungverteiler bzw. zweiter Adreßteil
MON	=	Monitorprogramm Einsprung
SV/2	=	Sprungverteiler Adresse : 2

Makrobefehlsliste Gruppenübersicht

Art	Code	Bef	Funktion	Benutzte Bytes
Branch	0, X	BC	Branch / Bedingung ⟶ M	OP | M
	3, X			OP | M
Register	4, X			OP | M
	12, X	CX	(R) ? M ⟶ ME, ML, MH	OP | M
Allgemein	13, 0	SM	setze Merker „n"	OP | F0
	13, 7	MVC	(N) ⟶ M, FL - Zeichen	OP | M | N | FL
	13,15			
arithmet.	14, 0	AP	(M) + (N) ⟶ M	OP | M | N
	14, 5	MVP	(N) ⟶ M	OP | M | N
	14, 8	IS	−(M) ⟶ M	OP | M
	14,15			
I/O-Bef.	15, 0	ACB	Eingabe Ziffern	OP | M | N | FF | FB
	15, 4	PUT	allgemeine Ausgabe	OP | M | FG | FL | F0
	15,15			

TRIUMPH ADLER INFORMATIONS SYSTEME

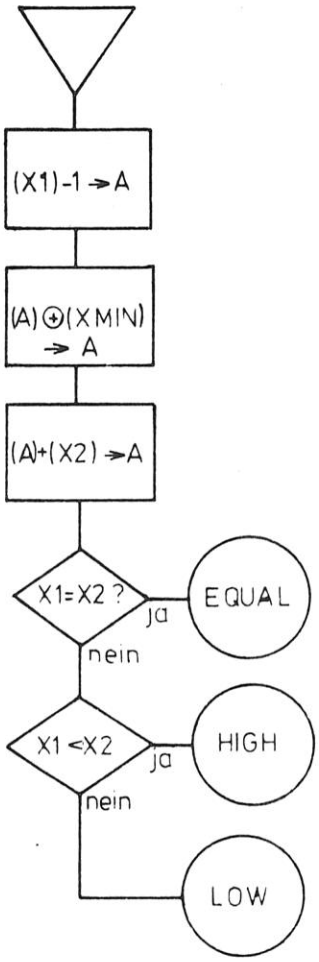
1. Vergleich zweier Zahlen

In den Speicherstellen X1 und X2 steht jeweils eine 16-bitige Dualzahl. Die Zahlenwerte sollen verglichen und eine Programmverzweigung, entsprechend den unten angegebenen Werten, durchgeführt werden.

X1	<	X2	BRANCH TO HIGH
X1	=	X2	BRANCH TO EQUAL
X1	>	X2	BRANCH TO LOW

Der Inhalt der Speicherstellen X1 und X2 soll erhalten bleiben. Eine Speicherzelle XMIN, in der alle Bits logisch 1 gesetzt sind, steht zur Verfügung.

Assembler Schreibweise	Mathematische Schreibweise
DAX	(X1) - 1 ⟶ A
EAX	(A) + (XMIN) ⟶ A
AAX	(A) + (X2) ⟶ A
BE	BR/(A) = Ø ⟶ EQUAL
BC	BR/(C) = 1 ⟶ HIGH
BR	BR ⟶ LOW



Dekrementierung von X1 um 1 und in den Akku abspeichern

△ X = X2 - X1

Einerkomplement

Zweierkomplement

Verzweigung nach EQUAL bei Akku = 0

Verzweigung nach HIGH bei gesetztem Überlauf (C = 1)

unbedingter Sprung bei weder EQUAL noch HIGH

Unterteilung in 7 Gruppen

1. Gruppe <u>Sprungbefehle (Branchbefehle)</u> 1. Reihe

Diese sind unterteilt in:

bedingte	Sprünge z.B.	BR/(A) = 0 $\longrightarrow N_Z$	BE
unbedingte	Sprünge z.B.	BR $\longrightarrow N_Z$	BR
direkte	Sprünge z.B.	BR $\longrightarrow$ P+N	BR
	z.B.	BS $\longrightarrow N_Z$	BS (Unterprogr.Spr.)
indirekte	Sprünge z.B.	BR $\longrightarrow$ (S+N)	BRX
	z.B.	RT $\longrightarrow$ (S+N)	RTX (Rückspr. v. Unterpr. ins Hautpr.)

Der Oberbegriff für die nachfolgenden Gruppen (2-7) lautet Akkumulatorbefehle.

2. Gruppe <u>Literalbefehle (Direktoperandenbefehle)</u> 2. Reihe 1. Hälfte

Diese sind unterteilt in:

Akkumulatorbefehle		N $\longrightarrow$ A	LAD
logische Befehle	z.B.	(A) $\wedge$ N $\longrightarrow$ A	MAD
arithmetische Befehle	z.B.	(A) + N $\longrightarrow$ A, C	AAD

2. Hälfte

3. Gruppe Zeropagebefehle	z.B.	$(N)_Z \longrightarrow$ A	LA
4. Gruppe Currentpagebefehle	z.B.	(P+N) $\longrightarrow$ A	LA

5. Gruppe Scratchpad Page Befehle 3. u. 4. Reihe

Diese sind unterteilt in 2 Hauptgruppen, wobei die 1. Gruppe (3. Reihe) nach Akku und Scratchpad läuft, während die 2. Gruppe nur nach Akku läuft. Diese beiden Hauptgruppen unterteilen sich wieder in

direkte	Befehle z.B. (A) $\longrightarrow$ $S+N_L$	SML
indirekte	Befehle z.B. (A) $\longrightarrow$ (S+N)	SMI
logische	Befehle z.B. (A) $\wedge$ $(S+N)_L \longrightarrow$ A, $S+N_L$	MML
arithmetische	Befehle z.B. (A,C)+$(S+N)_L \longrightarrow$ A,C, $S+N_L$	CML

6. Gruppe IO - Befehle in der 4. Reihe 7.)

diese sind unterteilt in:

direkte	Befehle z.B. IO $(N)_L \longleftrightarrow$ A	IOL
indirekte	Befehle z.B. IO ((S+N)) $\longleftrightarrow$ A	IOI

7. Gruppe Sonderbefehl in der 2. Reihe

z.B. RS Ø $\longrightarrow$ A16 SY (Rechtsshiften)

TRIUMPH ADLER INFORMATIONS SYSTEME

Mikrobefehle

Operationscode Bit 11 1 / 16 15 14 13 12	Assembler-schreibweise	0 0 — Mathematische Schreibweise	Assembler-schreibweise	0 1 — Mathematische Schreibweise
0 0 0 0 0	BS	BS → N_Z 1)	BS	BS → P + N 1)
0 0 0 1 0	BR	BR → N_Z	BR	BR → P + N
0 0 1 0 0	BE	BR / (A) = 0 → N_Z	BE	BR / (A) = 0 → P + N
0 0 1 1 0	BU	BR / (A) ≠ 0 → N_Z	BU	BR / (A) ≠ 0 → P + N
0 1 0 0 0	BP	BR/(A1...A9) odd Parity → N_Z 2)	BP	BR/(A1...A9) odd Parity → P+N 2)
0 1 0 1 0	BL	BR / (A1) = 1 → N_Z	BL	BR / (A1) = 1 → P + N
0 1 1 0 0	BZ	BR / (C) = 0 → N_Z	BZ	BR / (C) = 0 → P + N
0 1 1 1 0	BC	BR / (C) = 1 → N_Z	BC	BR / (C) = 1 → P + N
1 0 0 0 0	LAD	N → A		
1 0 0 1 0	MAD	(A) ∧ N → A		
1 0 1 0 0	EAD	(A) ⊕ N → A		
1 0 1 1 0	OAD	(A) ∨ N → A		
1 1 0 0 0	SY	Sonderbefehl 5)		
1 1 0 1 0	AFD	(A) + N + H.B. → A, C 6)		
1 1 1 0 0	CAD	(A, C) + N → A, C		
1 1 1 1 0	AAD	(A) + N → A, C		
0 0 0 0 1	SML	(A) → S + N_L	SMB	(A) → $(S + N)_B$
0 0 0 1 1	MML	(A) ∧ $(S + N)_L$ → A, S + N_L	MMB	(A) ∧ $((S+N))_B$ → A $(S+N)_B$
0 0 1 0 1	EML	(A) ⊕ $(S+N)_L$ → A, S + N_L	EMB	(A) ⊕ $((S+N))_B$ → A $(S+N)_B$
0 0 1 1 1	OML	(A) ∨ $(S+N)_L$ → A, S + N_L	OMB	(A) ∨ $((S+N))_B$ → A $(S+N)_B$
0 1 0 0 1	IML	$(S+N)_L$ + 1 → A, S+N_L	IMB	$((S+N))_B$ + 1 → A $(S+N)_B$
0 1 0 1 1	DML	$(S+N)_L$ - 1 → A, S+N_L	DMB	$((S+N))_B$ - 1 → A $(S+N)_B$
0 1 1 0 1	CML	(A,C) + $(S+N)_L$ → A,C, S+N_L	CMB	(A,C) + $((S+N))_B$ → A,C, $(S+N)_B$
0 1 1 1 1	AML	(A) + $(S+N)_L$ → A,C, S+N_L	AMB	(A) + $((S+N))_B$ → A,C, $(S+N)_B$
1 0 0 0 1	LAL	$(S+N)_L$ → A	LAB	$((S+N))_B$ → A
1 0 0 1 1	MAL	(A) ∧ $(S+N)_L$ → A	MAB	(A) ∧ $((S+N))_B$ → A
1 0 1 0 1	EAL	(A) ⊕ $(S+N)_L$ → A	EAB	(A) ⊕ $((S+N))_B$ → A
1 0 1 1 1	OAL	(A) ∨ $(S+N)_L$ → A	OAB	(A) ∨ $((S+N))_B$ → A
1 1 0 0 1	IOL &)	IO $(N)_L$ ← A 7)	IOB	IO $((S+N))_B$ ← A 7)
1 1 0 1 1	DAL	$(S+N)_L$ - 1 → A	DAB	$((S+N))_B$ - 1 → A
1 1 1 0 1	CAL	(A,C) + (S+N), → A, C	CAB	(A,C) + $((S+N))_B$ → A,C
1 1 1 1 1	AAL	(A) + $(S+N)_L$ → A, C	AAB	(A) + $((S+N))_B$ → A,C

L = Linkes Byte

B = Einzelbyte

Mikrobefehle

Operationscode Bit 11 1 / 16 15 14 13 12	Assembler-schreibweise	1 0 — Mathematische Schreibweise	Assembler-schreibweise	1 1 — Mathematische Schreibweise
0 0 0 0 0	BSX	BS → (S+N) 1)	BSI	BS → (P+N) 1)
0 0 0 1 0	BRX	BR → (S+N)	BRI	BR → (P+N)
0 0 1 0 0	BEX	BR / (A) = 0 → (S+N)	BEI	BR / (A) = 0 → (P + N)
0 0 1 1 0	BUX	BR / (A) ≠ 0 → (S+N)	BUI	BR / (A) ≠ 0 → (P + N)
0 1 0 0 0	RTX	RT → (S+N) 3)	RTI	RT → (P + N) 4)
0 1 0 1 0	BLX	BR / (A1) = 1 → (S+N)	BLI	BR / (A1) = 1 → (P + N)
0 1 1 0 0	BZX	BR / (C) = 0 → (S+N)	BZI	BR / (C) = 0 → (P + N)
0 1 1 1 0	BCX	BR / (C) = 1 → (S+N)	BCI	BR / (C) = 1 → (P + N)
1 0 0 0 0	LA	$(N)_Z$ → A	LA	(P + N) → A
1 0 0 1 0	MA	(A) ∧ $(N)_Z$ → A	MA	(A) ∧ (P + N) → A
1 0 1 0 0	EA2	(A) ⊕ $(N)_Z$ → A	EA	(A) ⊕ (P+N) → A
1 0 1 1 0	OA	(A) ∨ $(N)_Z$ → A	OA	(A) ∨ (P+N) → A
1 1 0 0 0	IA	$(N)_Z$ + 1 → A	IA	(P + N) + 1 → A
1 1 0 1 0	DA	$(N)_Z$ - 1 → A	DA	(P + N) - 1 → A
1 1 1 0 0	CA	(A, C) + $(N)_Z$ → A, C	CA	(A, C) + (P + N) → A, C
1 1 1 1 0	AA	(A) + $(N)_Z$ → A, C	AA	(A) + (P + N) → A, C
0 0 0 0 1	SMX	(A) → S + N	SMI	(A) → (S + N)
0 0 0 1 1	MMX	(A) ∧ (S+N) → A, S+N	MMI	(A) ∧ ((S+N)) → A, (S+N)
0 0 1 0 1	EMX	(A) ⊕ (S+N) → A, S+N	EMI	(A) ⊕ ((S+N)) → A, (S+N)
0 0 1 1 1	OMX	(A) ∨ (S+N) → A, S+N	OMI	(A) ∨ ((S+N)) → A, (S+N)
0 1 0 0 1	IMX	(S+N) + 1 → A, S+N	IMI	((S+N)) + 1 → A, (S+N)
0 1 0 1 1	DMX	(S+N) - 1 → A, S+N	DMI	((S+N)) - 1 → A, (S+N)
0 1 1 0 1	CMX	(A,C) + (S+N) → A,C, S+N	CMI	(A,C) + ((S+N)) → A,C,(S+N)
0 1 1 1 1	AMX	(A) + (S+N) → A,C, S+N	AMI	(A) + ((S+N)) → A,C, (S+N)
1 0 0 0 1	LAX	(S+N) → A	LAI	((S+N)) → A
1 0 0 1 1	MAX	(A) ∧ (S+N) → A	MAI	(A) ∧ ((S+N)) → A
1 0 1 0 1	EAX	(A) ⊕ (S+N) → A	EAI	(A) ⊕ ((S+N)) → A
1 0 1 1 1	OAX	(A) ∨ (S+N) → A	OAI	(A) ∨ ((S+N)) → A
1 1 0 0 1	IOX	IO (N) ↔ A 7)	IOI	IO ((S+N)) ↔ A 7)
1 1 0 1 1	DAX	(S+N) - 1 → A	DAI	((S+N)) - 1 → A
1 1 1 0 1	CAX	(A,C) + (S+N) → A,C	CAI	(A,C) + ((S+N)) → A,C
1 1 1 1 1	AAX	(A) + (S+N) → A,C	AAI	(A) + ((S+N)) → A,C

Bemerkung:

1) Beim Unterprogrammsprung BS wird vor der Ausführung des Sprungs der um +2 erhöhte alte Befehlszähler P in das A-Register A2 bis A16 und das C-Register nach A1 geladen
(P) + 2 → A2 ... A16, (C) → A1

2) Dieser Sprungbefehl wird dann ausgeführt, wenn im A-Register in A1 bis A9 eine ungerade Anzahl von "Einsen" stehen.

3) Rücksprung vom Unterbrechungsprogramm ins Hauptprogramm:
Der Inhalt von B1 der Rücksprungadresse wird nach C geladen.
Die Generalunterbrechungssperre wird rückgesetzt und wenn die Generalunterbrechungssperre vorher nicht gesetzt war, wird die Uhrunterbrechungssperre gelöscht.

4) Rücksprung vom Unterbrechungsprogramm ins Hauptprogramm:
Der Inhalt von B1 der Rücksprungadresse wird nach C geladen.
Die Generalunterbrechungssperre wird rückgesetzt und wenn die Generalunterbrechungssperre vorher gesetzt war, wird die Uhrunterbrechungssperre gesetzt und der Uhrunterbrechungswunsch gelöscht.

5) Sonderbefehl

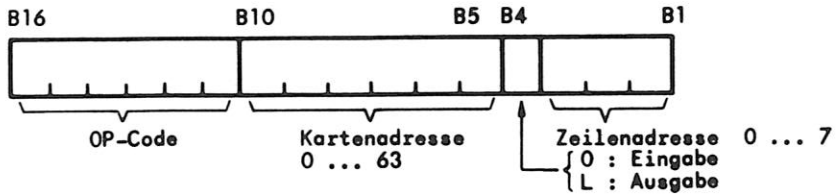
Der Sonderbefehl bezieht sich nur auf das A-Register. Im Adreßteil sind nebenstehende Befehle codiert. Kombinationen beider Befehlsgruppen sind möglich.

*	B10	B9	
RS	1	0	(A) Rechtsshift um 1 Bit; 0 → A16
RC	1	1	(A) Rechtsshift um 1 Bit; (C) → A16
GR	0	1 B1	Löschen Rechner
PR	1	x B3	Löschen Parityfehler (Option)

*	B3	B2	B1	
CC	0	0	0	: (C) → C
SC	0	0	1	: 1 → C
LC	0	1	0	: (A1) → C
HC	0	1	1	: (A16) → C
TC	1	0	0	: Uhrunterbrechungssperre → C
FC	1	0	1	: Netzausfall → C
PC	1	1	0	: Paritätsfehler → C
DC	1	1	1	: IO Unterbrechungswunsch → C

6) Dem Direktoperanden N werden die "Hohen Bits H.B." 11 ... 16 = 1 hinzugefügt.

7) IO-Befehlsformat:



Programmunterbrechung: 1 ms Uhr und Parityfehler (als Option) bewirkt (A) → XØ, BS → 12, IO-Unterbrechung bewirkt (A) → XØ, BS → 8.