Z U S E Z 43

Assembler

und

Grundbetriebssystem

Beschreibung

Vorläufige Ausgabe

September 1969

ZUSE KG

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Blatt - Nr. der Werksunterlage	
27	4
26	4
25	4
24	4
23	4
22	4
21	4
20	4
19	4
18	4
17	4
16	4
15	4
14	4
13	4
12	4
11	4
10	4
9	4
8	4
7	4
6	4
5	4
4	4
3	4
2	4
1	4
0/4	4
0/3	4
0/2	4
0/1	4
Ausgabe	4
Mitteilung	
Tag	1.9.69.
Name	EPB
TR	

Bemerkungen

Blatt 0/1 - 0/4 Leitblätter
Blatt 1 - 104 Programmierungsanleitung für die Z 43 (Grundbetriebssystem)

Ausgabe - Kennzeichnung oben eintragen

Tag | Ausgabe | Mitteilung | Name | TR | Freigabe:

Programmierungsanleitung (Grundbetriebssystem)

ZUSE KG

A26610-A9001-X-1-18

Blatt 0/1

4+104 Blätter

erste Vorausgabe | letzte Vorausgabe

ZUSE KG

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Blatt - Nr. der Werksunterlage	
58	4
57	4
56	4
55	4
54	4
53	4
52	4
51	4
50	4
49	4
48	4
47	4
46	4
45	4
44	4
43	4
42	4
41	4
40	4
39	4
38	4
37	4
36	4
35	4
34	4
33	4
32	4
31	4
30	4
29	4
28	4
Ausgabe	4
Mitteilung	
Tag	1.9.65.
Name	EPB
TR	

Bemerkungen

Ausgabe - Kennzeichnung oben eintragen

Tag	Ausgabe	Mitteilung	Name

TR | Freigabe:

Programmierungsanleitung (Grundbetriebssystem)

ZUSE KG

A26610-A9001-X-1-18

Blatt 0/2 Blätter

erste Vorausgabe | letzte Vorausgabe | Vordruck DV/4 (IV.15.62)

ZUSE KG

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Blatt - Nr. der Werksunterlage		
	89	4
	88	4
	87	4
	86	4
	85	4
	84	4
	83	4
	82	4
	81	4
	80	4
	79	4
	78	4
	77	4
	76	4
	75	4
	74	4
	73	4
	72	4
	71	4
	70	4
	69	4
	68	4
	67	4
	66	4
	65	4
	65	4
	63	4
	62	4
	61	4
	60	4
	59	4
Ausgabe		4
Mitteilung		
Tag		7.9.69.
Name		EPB
TR		

Bemerkungen

Ausgabe - Kennzeichnung oben eintragen

Tag | Mitteilung | Name

TR | Ausgabe | Freigabe:

Programmierungsanleitung (Grundbetriebssystem)

ZUSE KG

Blatt 0/3

Blätter

erste Vorausgabe | letzte Vorausgabe

Vordruck D V/4, (IX/5.67)

ZUSE KG

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Blatt - Nr. der Werksunterlage	
104	4
103	4
102	4
101	4
100	4
99	4
98	4
97	4
96	4
95	4
94	4
93	4
92	4
91	4
90	4
Ausgabe	4
Mitteilung	
Tag	1.9. 69.
Name	EPB
TR	

Bemerkungen

Ausgabe - Kennzeichnung oben eintragen

Tag | Mitteilung | Name

TR | Ausgabe | Freigabe:

Programmierungsanleitung (Grundbetriebssystem)

ZUSE KG

A26610-A9001-X-1-18

Blatt 0/4

Blätter

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

Inhalt

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten.

1. MASCHINENINTERNE BEFEHLE

1.1. Register

Die Anlage besitzt 16 Registerzellen von je 16 Bit Länge (Einfachwortformat). Diese Zellen können als Rechenregister oder als Indexzellen benutzt werden. Zur Adressierung der Registerzellen werden 4 Bit benötigt. Die Register tragen die Adressen 0 - 15. Die Register 0 - 14 stehen dem Benutzer zur Verfügung; dabei ist zu beachten, daß Register 0 nicht als Basisadressregister innerhalb von Langbefehlen (vgl. 1.4.4.) verwendet werden kann.

Register 15 ist als Befehlszählregister verdrahtet. Zwei benachbarte Registerzellen können als Doppelworte verarbeitet werden. (Dabei gelten Register 15 und Register 0 auch als benachbart.)

1.2. Kernspeicherorganisation

Der Kernspeicher ist hardwareseitig byteweise (ein Byte besitzt 8 Informationsbits) organisiert. Durch maschineninterne Befehle können Einfachworte (2 Bytes) und Doppelworte (4 Bytes) direkt angesprochen werden. Einfachworte besitzen gradzahlige, Doppelworte durch vier teilbare Kernspeicheradressen.

Bei ungradzahligen Kernspeicheradressangaben für maschineninterne Operationen wird das unterste Bit der Adresse nicht mitinterpretiert.

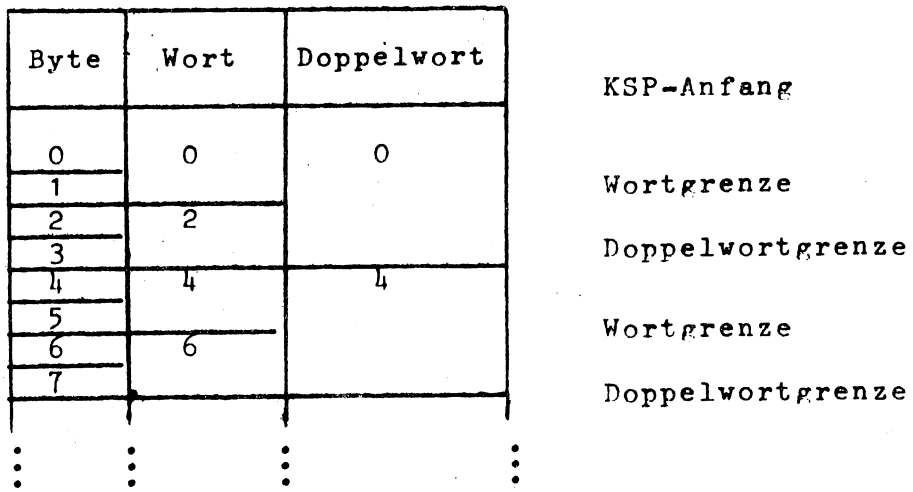
Kernspeicheradressen (es handelt sich immer um Byteadressen) besitzen Einfachwortformat. Mit den 16-Bit-Adressen ist ein Ausbau des Kernspeichers bis zu $65536 = 2^{16}$ Bytes möglich.

Der Mindestausbau des KSP beträgt 8 kBytes; er kann in Stufen von 8 kBytes erweitert werden. -

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Übersteigt die KSP-Adressangabe in einem Befehl die Ausbaustufe des KSP, so wird beim Lesen aus dem KSP 0 gelesen, während Schreibbefehle ignoriert werden.

Schema zur Byte-Adressierung des KSP



Am Anfang des KSP liegt der "geschützte Bereich". In den Zellen dieses Bereiches kann nur im Betriebszustand 2 (vgl. 1.7.6.1.) der Zentraleinheit geschrieben werden. -

Ein den geschützten Bereich umfassender Teil des Kernspeichers ist für das Betriebssystem reserviert. Dieser Bereich steht nicht für Benutzerprogramme zur Verfügung.

1.3. Formate der maschineninternen Befehle

Es gibt zwei Befehlsformate:

1.3.1. 16-Bit-Formate (Kurzbefehle)

1.3.2. 32-Bit-Formate (Langbefehle)

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung und Mitteilung ihres Inhalts nicht gestattet soweit nicht ausdrücklich zugestanden Zuwiderhandlungen verpflichten zu Schadenersatz Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

1.3.1. Kurzbefehle (Befehlsklassen R, C und S) werden in einem Wort von 16 Bit Länge abgelegt. Ein Kurzbefehl besteht im allgemeinen aus der Verschlüsselung des Operationsteils (8 Bit) und aus zwei Registeradressen (je 4 Bit).

Im Assemblercode wird ein Kurzbefehl in der Form θa,b programmiert (Genaueres vgl. 6.1.). Dabei bedeutet:

θ die Buchstabenverschlüsselung des Operationsteils

a die Adresse des ersten Registers (linkes Adressenfeld; Zielregister)

b die Adresse des zweiten Registers (rechtes Adressenfeld).

Kurzbefehle haben in der Maschine folgende Darstellung:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
eigentlicher Op-Teil					EW DW	Kl.		a				b			

Stellen 0 - 3	Registeradresse	b
Stellen 4 - 7	Registeradresse	a
Stellen 8 - 9	Befehlsklasse:	00 Klasse R
		OL Klasse C
		LO Klasse S
Stelle 10	Wortart der Operanden:	O Einfachwortbefehl
		L Doppelwortbefehl

Bei Doppelwortbefehlen ist immer die niedere Adresse des zu verarbeitenden Doppelwortes anzugeben.

Stellen 11 - 15 Verschlüsselung d. eigentl. Operationsteils

1.3.2. Adress-Langbefehle (Befehlsklasse A) werden in zwei aufeinanderfolgenden Worten des Kernspeichers abgelegt. Die Befehlsklasse ist durch LL gekennzeichnet. Das erste Wort enthält die Verschlüsselung des Operationsteils und zwei Registerangaben (wie bei Kurzbefehlen). Das zweite Wort enthält eine Kernspeicheradressangabe.

Im Assemblercode wird ein Langbefehl in der Form θa,b,c programmiert. Dabei bedeutet:

θ die Buchstabenverschlüsselung des Operationsteils
a die erste Registerangabe (Zielregister)
b die zweite Registerangabe (Basisregister)
c eine Kernspeicheradresse

(In der Regel wird c symbolisch angegeben;
Näheres vgl. 6.1.2.2.)

Langbefehle haben also in der Maschine folgende Darstellung:

Wort 1: Wie in 1.3.1.
Wort 2: Kernspeicheradressenangabe c

1.4. __Befehlsklassen__

Es werden vier Befehlsklassen unterschieden:

1.4.1. Klasse R (Register-Kurzbefehle)

Allgemeiner Code: θa,b

In der Befehlsklasse R wird mit den Inhalten von Register a (1. Operand, Ergebnis) und Register b (2. Operand) vermöge θ operiert.

In dieser Klasse werden 22 Befehle unterschieden.

θ ist eines der Symbole:

A	B	R	I	SP	PKA
AA	BB	RR		IP	PKE
S	BN	RZ		STP	
SS	BBN	RRZ		EG	
M		L		EU	
D		LL		EN	
				EP	

Delivery, duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

(Spalte 1: Arithmetische Befehle
2: Transportbefehle
3: Verschiebebefehle
4: Logische Befehle
5: Sprungbefehle
6: Sonderbefehle)

Die Befehle PKA und PKE (vgl. 1.7.6.3.) dürfen im Assemblercode für Benutzerprogramme nicht codiert werden.

1.4.2. Klasse C (Konstanten-Kurzbefehle)

Allgemeiner Code: Θa,b

In der Befehlsklasse C wird die im rechten Adressenfeld (also b) angegebene Zahl als Konstante aufgefaßt. Es können in C-Befehlen nur Konstanten zwischen 0 und 15 angegeben werden.

Die Buchstabencodes (Assembler) der C-Befehle beginnen mit C. Ausnahmen bilden die Befehle PZW und USE; diese Befehle dürfen im Assemblercode für Benutzerprogramme ohnehin nicht programmiert werden (vgl. 1.7.6.1. und 1.7.6.2.).

Es werden 19 Befehle unterschieden.

Θ ist eines der Symbole:

CA	CB	CR	CI	CWP	PZW
CS	CBN	CRR	CWS	CSP	USE
CM		CRZ	CWL	CIP	
		CRRZ			
		CL			
		CLL			

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

1.4.3. Klasse S (Substitutions-Kurzbefehle)

Allgemeiner Code: Θa,b

Bei Befehlen der Klasse S wird der zweite Operand durch eine Adressensubstitution gewonnen. Über das im rechten Adressenfeld angegebene Register wird eine Kernspeicherzelle aufgerufen. Die Buchstabencodes von Befehlen der Klasse S beginnen mit einem G. Es gibt 14 Substitutionsbefehle.

Θ ist eines der Symbole:

GA	GB	-	GI	GSP	-
GAA	GBB			GIP	
GS	GBN				
GSS	GBBN				
GM	GU				
	GUU				

1.4.4. Klasse A (Adress-Langbefehle)

Allgemeiner Code: Θa,b,c

Bei den Befehlen der Klasse A operiert der Inhalt des Registers a mit der Kernspeicherzelle, deren Adresse sich aus dem Inhalt des Registers b und der Adressangabe c vermöge <b>+c zusammensetzt. Wird für b das Befehlszählregister 15 angegeben, so ist zu beachten, daß zum Zeitpunkt der Berechnung der effektiven Adresse <b>+c das Befehlszählregister 15 auf der Kernspeicher-Adresse des nächsten Befehles steht.

Für b=0 wird nicht, wie oben beschrieben <b> (Inhalt von b), sondern 0 genommen; d.h. c ist bereits die effektive Adresse. Diese Regelung gilt aber nur für Langbefehle.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Copying or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Es gibt 22 Adress-Langbefehle.
Θ ist eines der Symbole:

A	B	-	I	SP	-
AA	BB			IP	
S	BN			EG	
SS	BBN			EU	
M	LCB			EN	
	U			EP	
	UU			F	
	BT				
	UT				

1.5. Doppelwortbefehle

Doppelwortbefehle verarbeiten zwei benachbarte Register bzw. Kernspeicherzellen als zusammenhängene Information. Als Adresse im Befehl wird immer die Zelle mit der niederen Adresse angegeben; diese Zelle enthält die Bits mit der niederen Wertigkeit. Kernspeicher-Adressen für diese Befehle müssen immer durch vier teilbar sein. - Im Assemblercode besitzen Doppelwortbefehle einen Doppelbuchstaben.

All rights are reserved

1.6. Befehlsliste

1.6.1. Symbolik

a	Registernummer
b	Registernummer
a+1,a	Registernummern von zwei benachbarten Registerzellen, die als Doppelwort betrachtet werden. (Es wird diese Schreibweise gewählt, weil in Register a die niederwertigen Bits stehen).
c	Kernspeicherangabe in Langbefehlen.
Θ	Allgemeiner Operationscode
<a>	Inhalt von Register a
<a+1,a>	Inhalt der Register a+1 und a als Doppelwort betrachtet
<<b>>	Inhalt der Kernspeicherzelle mit der Adresse <b>
<<b>+c>	Inhalt der Kernspeicherzelle mit der Adresse <b>+c
<b>→a	Transportsymbol : Der Inhalt von Zelle b geht nach Zelle a.

1.6.2. Befehlsübersicht und Interncodes

Grund-code	R	C	S	A	Interncode					
	Θa,b	Θa,b	Θa,b	Θa,b,c	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}
A	A	CA	GA	A	0	0	0	0	L	0
AA	AA	-	GAA	AA	0	0	0	0	L	L
S	S	CS	GS	S	0	0	L	0	L	0
SS	SS	-	GSS	SS	0	0	L	0	L	L
M	M	CM	GM	M	L	0	L	0	0	L
D	D	-	-	-	L	0	L	0	L	L
B	B	CB	GB	B	0	0	0	L	L	0
BB	BB	-	GBB	BB	0	0	0	L	L	L
BN	BN	CBN	GBN	BN	0	0	L	0	0	0
BBN	BBN	-	GBBN	BBN	0	0	L	0	0	L
LCB	-	-	-	LCB	0	L	0	0	L	0
U	-	-	GU	U	0	0	0	L	0	0
UU	-	-	GUU	UU	0	0	0	L	0	L
BT	-	-	-	BT	0	L	0	0	L	L
UT	-	-	-	UT	0	L	0	0	0	L
R	R	CR	-	-	L	0	0	0	L	0
RR	RR	CRR	-	-	L	0	0	0	L	L
RZ	RZ	CRZ	-	-	L	0	0	L	0	0
RRZ	RRZ	CRRZ	-	-	L	0	0	L	0	L
L	L	CL	-	-	L	0	0	L	L	0
LL	LL	CLL	-	-	L	0	0	L	L	L
I	I	CI	GI	I	0	0	L	L	0	L
CWS	-	CWS	-	-	0	L	L	0	L	0
CWL	-	CWL	-	-	0	L	L	L	0	0
CWP	-	CWP	-	-	0	L	L	L	L	0
SP	SP	CSP	GSP	SP	0	0	L	L	L	0
IP	IP	CIP	GIP	IP	0	0	L	L	0	0
EG	EG	-	-	EG	L	L	L	0	L	0
EU	EU	-	-	EU	L	L	L	L	0	0
EN	EN	-	-	EN	L	L	L	L	L	0
EP	EP	-	-	EP	L	L	L	0	0	0
F	-	-	-	F	0	L	0	0	L	0
STP	STP	-	-	-	0	0	0	0	0	0
PZW	-	PZW	-	-	0	L	0	L	L	0
USE	-	USE	-	-	0	L	0	0	0	0
PKA	PKA	-	-	-	L	L	0	0	L	0
PKE	PKE	-	-	-	L	L	0	L	0	0
PER	Nichtinterpretierber Interncode (Sonderbef.f. schnellen Einsprung in Peripherie-Programme)				L	L	0	0	0	0

Copyright duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

1.6.3. Operationszeiten in µsec

Grund-code	Befehlsklasse R (Reg.-Kurzbef.)	C (Konstanten-Kurzbef.)	S (G) (Substit. Kurzbef.)	A (Adress-Langbefehle)
A	1,96	1,96	3,9	5,9
AA	2,94	-	5,9	7,84
S	1,96	1,96	3,9	5,9
SS	2,94	-	5,9	7,84
M	12,4	12,4	14,4	16,4
D	13,4	-	-	-
B	1,96	1,96	3,9	5,9
BB	2,94	-	5,9	7,84
BN	1,96	1,96	3,9	5,9
BBN	2,94	-	5,9	7,84
U	-	-	3,9	5,9
UU	-	-	5,9	7,84
BT,UT	-	-	-	3,92+n.1,96
R	2,46+n.0,5	2,46+n.0,5	-	-
RR	4,46+n.0,5	4,46+n.0,5	-	-
RZ	2,46+n.0,5	2,46+n.0,5	-	-
RRZ	4,46+n.0,5	4,46+n.0,5	-	-
L	2,46+n.0,5	2,46+n.0,5	-	-
LL	4,46+n.0,5	4,46+n.0,5	-	-
I	1,96	1,96	3,9	5,9
CWS	-	9,8	-	-
CWL	-	9,8	-	-
CWP	-	2,46+n.0,5(1)	-	-
SP	1,96(+1)	1,96(+1)	3,9(+1)	5,9(+1)
IP	1,96(+1)	1,96(+1)	3,9(+1)	5,9(+1)
EG	1,96(+1)	-	-	5,9(+1)
EU	1,96(+1)	-	-	5,9(+1)
EN	1,96(+1)	-	-	5,9(+1)
EP	1,96(+1)	-	-	5,9(+1)
F	-	-	-	5,9
STP	1,96	-	-	-
PZW	-	3,92	-	-
USE	-	1,96	-	-
PKE	1,96	-	-	-
PKA	1,96	-	-	-

n = Anzahl der Schiebschritte bei Verschiebebefehlen
n = Anzahl der übertragenen Worte bei BT und UT
Bei Sprungbefehlen ist die Operationszeit davon abhängig, ob der Sprung ausgeführt wird oder nicht. Wird er ausgeführt, so verlängert sich die Operationszeit um 1 µsec

Delivery, duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

1.7. Erläuterung der Befehle

1.7.1. Arithmetische Befehle

1.7.1.1. Addieren

Die angesprochenen Zelleninhalte werden als binäre Festpunktzahlen verstanden; können jedoch auch als Bruchzahlen (vgl. 4.3.1.) aufgefaßt werden.

R.) Aa,b

<a>+<b>→a

C.) CAa,b

<a>+b→a

S.) GAa,b

<a>+<<b>>→a

A.) Aa,b,c

<a>+<<b>+c>→a

Beispiel: Durch den Befehl CA15,b wird das Befehlszählregister verändert; auf diese Weise können also kurze Relativsprünge nach vorne programmiert werden. Dabei ist zu beachten, daß bei Ausführung der Addition das Befehlszählregister 15 bereits auf der nächsten Zelle steht; es kann also um maximal 8 Worte nach vorne gesprungen werden (für b=14).

1.7.1.2. Addieren DW

R.) AAa,b

<a+1,a>+<b+1,b>→a+1,a

S.) GAAa,b

<a+1,a>+<<b>+2,<b>>→a+1,a

A.) AAa,b,c

<a+1,a>+<<b>+c+2,<b>+c>→a+1,a

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

Copyright duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

1.7.1.3. Subtrahieren

R.) Sa,b

<a>-<b>→a

C.) CSa,b

<a>-b→a

S.) GSa,b

<a>-<<b>>→a

A.) Sa,b,c

<a>-<<b>+c>→a

Beispiel: Durch den Befehl CS15,b können kurze Relativsprünge nach hinten programmiert werden. Dabei ist zu beachten, daß bei Ausführung der Subtraktion das Befehlszählregister 15 bereits auf der nächsten Zelle steht; es kann also um maximal 6 Worte zurückgesprungen werden (für b=14).

1.7.1.4. Subtrahieren DW

R.) SSa,b

<a+1,a>-<b+1,b>→a+1,a

S.) GSSa,b

<a+1,a>-<<b>+2,<b>>→a+1,a

A.) SSa,b,c

<a+1,a>-<<b>+c+2,<b>+c>→a+1,a

Beispiel: Durch den Befehl SS3,3 wird die Doppelzelle 4,3 gelöscht.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten.

1.7.1.5. Multiplikation

Die in der Maschine verdrahtete Multiplikation kann man auf zweierlei Art interpretieren:

a) Bruchzahlmultiplikation

Die Multiplikationsbefehle verstehen den Inhalt eines Einfachwortes m als $m.2^{-15}$; also mit den Wertigkeiten einer Einfachwortbruchzahl (vgl. 4.3.1.). Das Ergebnis steht in einem Doppelwort mit denStellenwertigkeiten einer Doppelwortbruchzahl (vgl. 4.3.2.)

R.) Ma,b

<a> . <b>→a+1,a

C.) CMa,b

<a> . b→a+1,a

Das Bitmuster, das durch die Konstante b mit b<16 erzeugt wird, ist natürlich auch mit den Bruchzahlwertigkeiten zu verstehen.

S.) GMa,b

<a> . <<b>>→a+1,a

A.) Ma,b,c

<a> . <<b>+c>→a+1,a

b) Interpretation als Festpunktzahlmultiplikation.

Unmittelbar werden also nur Zahlen zwischen -1 und +1 mathematisch richtig multipliziert. Will man die Zelleninhalte <a> und <b> als echte Festpunktzahlen multiplizieren, so muß man das Ergebnis noch, wie man sich leicht aus den Stellenwertigkeiten herleitet, um eine Stelle nach rechts schieben; also

Ma,b
CRRa,0 (vgl. 1.7.3.2.) codieren.

Analoges gilt für die Multiplikationsbefehle der Klassen C, S und A. Vgl. dazu auch 2.4.

Copyright © by Zuse KG ... The contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

1.7.1.6. Dividieren

Die in der Maschine verdrahtete Division kann man auf zweierlei Art interpretieren:

R.) Da,b

<a+1,a> : <b>→a+1,a

a) Bruchzahldivision

Ein Doppelwort wird durch ein Einfachwort dividiert. Die verdrahtete Division versteht den Dividenden als Doppelwortbruchzahl (4.3.2.) und den Divisor als Einfachwortbruchzahl (4.3.1.). Der Quotient steht als Einfachwortbruchzahl in Register a. Die Größenordnungen von Dividend und Divisor müssen so beschaffen sein, daß der Quotient zwischen -1 und 1 liegt.

Um als Ergebnis die Bruchzahldarstellung zu erhalten, führt die verdrahtete Division einen Schritt zu wenig aus; d.h. das letzte Bit des Dividenden geht verloren. Will man mit dem undividierten Rest in Register a+1 weitere Bits (also 2^{-16}, 2^{-17},....) des Ergebnisses gewinnen, so muß man bei exakter Verarbeitung den Rest folgendermaßen aufbereiten:

<a+1> . 2+P wobei
P = 0, wenn der Dividend gerade war;
P = 1, wenn der Dividend ungerade war.

Vorzeichen: Der Quotient steht vorzeichenrichtig in Register a. Der Rest in Register a+1 besitzt das Vorzeichen des Dividenden.

b) Interpretation als Festpunktzahldivision.
Will man den Inhalt der Doppelzelle a+1,a und den Inhalt von b als echte Festpunktzahlen dividieren, so muß man den Dividenden vorher, wie man sich leicht aus den Stellenwertigkeiten herleitet, um eine Stelle nach links schieben; also

```
CLLa,0 |        | AAa,a      (schneller)
Da,b   | oder   | Da,b
```

Der Quotient steht dann vorzeichenrichtig in Register a. Der undividierte Rest steht mathematisch richtig mit dem Vorzeichen des Dividenden in Register a+1. Die Größenordnungen von Dividend und Divisor müssen so liegen, daß der Quotient Einfachwortformat besitzt; also kleiner als $2^{15} = 32768$ ist. Diese Einschränkung beinhaltet auch, daß die Verschiebung CLLa,0 möglich ist, ohne daß eine Stelle verloren geht. Der Absolutbetrag des Dividenden muß also stets kleiner sein als $2^{30} = 1073741824$; d.h. die beiden oberen Bits der Doppelzelle a+1,a müssen gleich sein.

1.7.2. Transportbefehle

1.7.2.1. Bringen

R.) Ba,b
<b>→a

C.) CBa,b
b→a

S.) GBa,b
<<b>>→a

A.) Ba,b,c
<<b>+c>→a

Beispiele: Ein unbedingter Sprung nach (SPRU) kann in der Form

GB15,15
O(SPRU) (vgl. 6.1.4.)

oder, wenn die Sprungadresse in Register b steht, in der Form

B 15,6

codiert werden.

1.7.2.2. Bringen DW

R.) BBa,b

<b+1,b>→a+1,a

C.) GBBa,b

<<b>+2,<b>>→a+1,a

A.) BBa,b,c

<<b>+c+2,<b>+c>→a+1,a

1.7.2.3. Bringen negativ

R.) BNa,b

-<b>→a

C.) CBNa,b

-b→a

S.) GBNa,b

-<<b>>→a

A.) BNa,b,c

-<<b>+c>→a

1.7.2.4. Bringen negativ DW

R.) BBNa,b

-<b+1,b>→a+1,a

S.) GBBNa,b

-<<b>+2,<b>>→a+1,a

A.) BBNa,b,c

-<<b>+c+2,<b>+c>→a+1,a

Copyright (C) ZUSE KG, Bad Hersfeld
All rights reserved. Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten insbesondere für den Fall der Patenterteilung oder GM-Eintragung.

1.7.2.5. Langbefehl Konstante bringen

A.) LCBa,b,c mit a≠15

<b>+c→a

Dieser Befehl hat intern die gleiche Verschlüsselung wie der F-Sprung (vgl. 1.7.5.8.); dort ist a=15 zu setzen. Mit dem Befehl LCBa,0,c können beliebige Festpunkt-Einfachwort-Konstanten c nach Register a (a≠15) gebracht werden (also auch ungerade Zahlen).

1.7.2.6. Umspeichern

Mit Hilfe der Umspeicherbefehle wird der Inhalt einer Registerzelle a in den Kernspeicher transportiert; dabei bleibt der alte Registerinhalt erhalten.

S.) GUa,b

<a>→<b>

A.) Ua,b,c

<a>→<b>+c

1.7.2.7. Umspeichern DW

S.) GUUa,b

<a+1,a>→<b>+2,<b>

A.) UUa,b,c

<a+1,a>→<b>+c+2,<b>+c

Copyright reserved. Duplication of this document and the use or disclosure of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

1.7.2.8. Bringe-Transfer

A.) BTa,b,c mit $a \leq 14$

<<b>+c> → a

<<b>+c+2> → a+1

. .

. .

. .

<<b>+c+2 . (14-a)> → 14

Durch diesen Befehl werden also die Register a bis 14 geladen.

1.7.2.9. Umspeicher-Transfer

A.) UTa,b,c mit $a \leq 14$

<a> → <b>+c

<a+1> → <b>+c+2

. .

. .

. .

. .

<14> → <b>+c+2 . (14-a)

Durch diesen Befehl werden also die Inhalte der Register a bis 14 in den Kernspeicher transferiert.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

Alle Rechte vorbehalten. Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. All rights reserved. Reproduction and communication of this document not permitted without express authorization.

1.7.3. Verschiebebefehle

1.7.3.1. Schieben rechts arithmetisch

R.) Ra,b

C.) CRa,b

Der Inhalt von Register a wird um <b>+1 bzw. b+1 Stellen nach rechts verschoben. Ist <b>>15, so wird modulo 16 verschoben. Die Stellen, die rechts herausgeschoben werden, gehen verloren. Links wird die Vorzeichenstelle nachgezogen.

1.7.3.2. Schieben rechts arithmetisch DW

R.) RRa,b

C.) CRRa,b

Der Inhalt der Register a+1 und a wird verkoppelt um <b>+1 bzw. b+1 Stellen nach rechts verschoben. Ist <b>>15, so wird modulo 16 verschoben. Die Stellen, die rechts aus dem Register a herausgeschoben werden, gehen verloren. In Wort a+1 wird die Vorzeichenstelle nachgezogen.

1.7.3.3. Schieben rechts zyklisch

R.) RZa,b

C.) CRZa,b

Der Inhalt von Register a wird um <b>+1 bzw. b+1 Stellen zyklisch nach rechts verschoben. Bei jedem Verschiebeschritt wird also die Stelle, die rechts herausgeschoben wird, in die Vorzeichenstelle des Registers a übernommen.

Alle Rechte vorbehalten. Jede Vervielfältigung, auch auszugsweise, bedarf unserer Zustimmung. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

1.7.3.4. Schieben rechts zyklisch DW

R.) RRZa,b

C.) CRRZa,b

Der Inhalt der Register a+1 und a wird verkoppelt zyklisch um <b>+1 bzw. b+1 Stellen nach rechts verschoben. Bei jedem Verschiebeschritt wird also die Stelle, die rechts aus dem Register a herausgeschoben wird, in die Vorzeichenstelle des Registers a+1 übernommen. Ist <b>>15, so wird modulo 16 verschoben.

1.7.3.5. Schieben links

R.) Ln,b

C.) CLa,b

Der Inhalt von Register a wird um <b>+1 bzw. b+1 Stellen nach links verschoben. Ist <b>>15, so wird modulo 16 verschoben. Die Stellen, die links herausgeschoben werden, gehen verloren. Rechts werden Nullen nachgezogen.

1.7.3.6. Schieben links DW

R.) LLa,b

C.) CLLa,b

Der Inhalt der Register a+1 und a wird verkoppelt und <b>+1 bzw. b+1 Stellen nach links verschoben. Ist <b>>15, so wird modulo 16 verschoben. Die Stellen, die links aus dem Register a+1 herausgeschoben werden, gehen verloren. Im Wort a werden Nullen nachgezogen.

1.7.4. Logische Befehle

1.7.4.1. Intersektion (Konjunktion, UND)

R.) Ia,b
<a>^<b>→a

C.) CIa,b
<a>^b→a

S.) GIa,b
<a>^<<b>>→a

A.) Ia,b,c
<a>^<<b>+c>→a

Das Zeichen ^ soll die stellenweise logische Konjunktion symbolisieren.

1.7.4.2. Weiche setzen

C.) CWSa,b

Das Bit mit der Wertigkeit 2^b von Register a wird auf L gesetzt. Ansonsten bleibt der Inhalt von Register a erhalten.

1.7.4.3. Weiche löschen

C.) CWLa,b

Das Bit mit der Wertigkeit 2^b von Register a wird auf 0 gesetzt. Ansonsten bleibt der Inhalt von Register a erhalten.

Copyright ... All rights reserved ... Offenders are liable to the payment ... the event of the grant of a Patent or the registration of a Utility Model

1.7.5. Sprungbefehle

1.7.5.1. Weiche prüfen

C.) CWPa,b

Das Bit mit der Wertigkeit 2^b von Register a wird geprüft. Wenn dieses Bit = 0 ist, dann wird der nächste Befehl ausgeführt. Wenn dieses Bit ≠ 0 ist, dann werden 4 Bytes übersprungen. Der Inhalt von a wird nicht verändert.

1.7.5.2. Subtraktion mit Prüfung

R.) SPa,b
C.) CSPa,b
S.) GSPa,b
A.) SPa,b,c

Die Differenz R.) D = <a>-<b>
C.) D = <a>- b
S.) D = <a>-<<b>>
A.) D = <a>-<<b>+c> wird geprüft.

Ist D≥0, so wird der nächste Befehl ausführt.
Ist D<0, so werden 4 Bytes übersprungen.
Die Inhalte von a und b bleiben unverändert.

1.7.5.3. Intersektion mit Prüfung

R.) IPa,b
C.) CIPa,b
S.) GIPa,b
A.) IPa,b,c

Copyright © 1969 by Konrad Zuse KG, Hünfeld. All rights reserved. Copying or duplication of this document or any part thereof is not permitted without the express written authority of the owner. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

Das Bitmuster R.) I =<a>^<b>

C.) I =<a>^b

S.) I =<a>^<<b>>

A.) I =<a>^<<b>+c> wird geprüft

Ist I≠0, so wird der nächste Befehl ausgeführt.
Ist I=0, so werden 4 Bytes übersprungen.
Die Inhalte von a und b bleiben unverändert.

1.7.5.4. Springen wenn = 0

R.) EGa,b
Wenn <a>=0, dann Sprung nach <b>; sonst nächster Befehl.

A.) EGa,b,c
Wenn <a>=0, dann Sprung nach <b>+c; sonst nächster Befehl.

1.7.5.5. Springen wenn ≠ 0

R.) EUa,b
Wenn <a>≠0, dann Sprung nach <b>; sonst nächster Befehl.

A.) EUa,b,c
Wenn<a>≠0, dann Sprung nach <b>+c; sonst nächster Befehl.

Beispiele: a) Ein unbedingter Sprung kann vom Benutzer in der Form EU15,b,c programmiert werden; denn der Inhalt des Befehlszählregisters 15 wird für den Benutzer nie Null.

b) Relative Sprünge sind auch in der Form EU15,15,c programmierbar. Dabei ist zu beachten, daß bei der Berechnung von <b>+c das Befehlszählregister 15 auf dem nächsten Befehl steht.

1.7.5.6. Springen wenn ≥ 0

R) EPa,b

Wenn <a>$\geq$0, dann Sprung nach <b>; sonst nächst.Bef.

A) EPa,b,c

Wenn <a>$\geq$0, dann Sprung nach <b>+c; " " "

1.7.5.7. Springen wenn < 0

R) ENa,b

Wenn <a><0, dann Sprung nach <b>; sonst nächst.Bef.

A) ENa,b,c

Wenn <a><0, dann Sprung nach <b>+c; " " "

1.7.5.8. Springe nach UP

A) F15,b,c

Für b=0 (Normalfall) ergibt sich folgender Ablauf:
Der F-Sprung stehe in der Kernspeicherzelle K.
Die Zelle c wird mit K+4 (Rücksprungadresse) geladen; d.h. am Kopf eines UP muß eine Speicherzelle für die Rückkehradresse freigehalten werden.
Außerdem wird auf Kernspeicherzelle c+2 gesprungen (Start des UP).
Symbolisch: K+4 → c
c+2 → R15
Der Rücksprung ins Hauptprogramm erfolgt durch den Befehl B15,0,c

1.7.5.9. Stop

R) STPa,b

Trat der Stop im Programmzustand 2 auf, so fällt die Maschine in einen statischen Stop. Durch Drükken der Starttaste wird der Programmablauf mit dem nächsten Befehl fortgesetzt. Ein Stop im Programmzustand 1 hat die gleiche Wirkung, außerdem wird der Stop mit dem nächsten Interrupt (vgl. 1.7.6.1.) aufgehoben.

Dieses Dokument ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Jede Verwendung, Vervielfältigung oder Weitergabe ohne vorherige schriftliche Genehmigung der ZUSE KG ist untersagt. Dieses Dokument ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Jede Verwendung, Vervielfältigung oder Weitergabe ohne vorherige schriftliche Genehmigung der ZUSE KG ist untersagt.

1.7.6. Sonderbefehle

Die folgenden Befehle sind für den Benutzer, der im Assembler-Code programmiert, nicht codierbar; sondern nur über Makro-Aufrufe (Sprung ins Betriebssystem) erreichbar.

1.7.6.1. Programmzustand wechseln

C.) PZWa,b

Allgemeine Betrachtung über Programmzustände:
Der Rechner verfügt über zwei Programmzustände.
In Programmzustand 1 (P1) laufen Benutzerprogramme ab. In diesem Zustand sind Programme unterbrechbar (durch eine der drei unten beschriebenen Ursachen) und können den geschützten Bereich über F-Sprung- und Umspeicherbefehle nicht erreichen (Schreibsperre eingeschaltet).
In Programmzustand 2 (P2) läuft das Betriebssystem ab.

Der Übergang von P1 nach P2 erfolgt durch eine der drei folgenden Unterbrechungsursachen:

(1) Nicht interpretierbarer Befehl, privilegierter Befehl (PKA oder PKE) in P1 oder Schreibsperren-Alarm.
(2) Programmierte Unterbrechung (PZW-Befehl) in P1.
(3) Interrupt (Meldung von der Peripherie-Schnittstelle).

Programmzustandswechsel von P1 nach P2:

Jede der drei Unterbrechungsursachen (1), (2) oder (3) führt zu einem Übergang von P1 nach P2 (d.h. die Unterbrechbarkeit und die Schreibsperre werden abgeschaltet); weiterhin wird die im Befehlszähler stehende Rückkehradresse in die Kernspeicherzelle 0,1 gebracht und es wird ein Sprung ausgeführt. Bei Unter-

brechungsursache (1) erfolgt der Sprung nach Zelle 2,3, bei Ursache (2) nach Zelle 6,7 und bei Ursache (3) nach Zelle 10,11.

Bei jeder der drei Unterbrechungsursachen wird eine bestimmte Routine des Betriebssystems durchlaufen. Für die hardware sind die Angaben a und b in PZWa,b ohne Bedeutung. Das Betriebssystem interpretiert a,b jedoch als eine Nummer, der ein bestimmter X- oder Y-Befehl (vgl. 2.1. und 3.) zugeordnet ist.

Programmzustandswechsel von P2 nach P1:

Ein Programmzustandswechsel von P2 nach P1 kann nur durch die Befehle PZW und USE (vgl. 1.7.6.2.) erreicht werden.
Bei PZW (im Betriebssystem wird PZW255 verwendet) wird die Unterbrechbarkeit und Schreibsperre wieder eingeschaltet und es erfolgt ein Sprung nach der Adresse, die in Zelle 0,1 notiert ist.

1.7.6.2. Unterbrechbarkeit und Schreibsperre ein

C.) USEa,b

Programmzustandswechsel von P2 nach P1:

Die Unterbrechbarkeit und die Schreibsperre werden wieder eingeschaltet. Anschließend wird kein Sprung (wie bei PZWa,b) ausgeführt; sondern der nächste Befehl wird ausgeführt (vgl. auch 1.7.6.1.).

1.7.6.3. Peripheriebefehle

Da, wie oben schon erwähnt, Peripheriebefehle (privilegiert für den P2-Zustand) für den Benutzer nicht direkt programmierbar (sondern nur über E/A-Makrobefehle aufrufbar) sind, wird hier nur ein Überblick

über die E/A-Schnittstelle und die Peripheriebefehle gegeben.

Allgemeines zur E/A-Schnittstelle der Z 46:

Die Schnittstelle besitzt zwei Kanaltypen; und zwar einen Multiplexkanal (MPX-Kanal) und einen Schnellkanal (wahlweise). Der Datenverkehr über Multiplexkanal wird Byte für Byte durch ein Ein-Ausgabeprogramm gesteuert. Über den Schnellkanal wird der Datenverkehr nicht durch Programm, sondern durch hardware gesteuert. Entsprechend den Angaben eines Bytezählers und eines Adresszählers (die beide zur Schnellkanal-Steuerung gehören) wird ein ganzer Datenblock zwischen der angewählten PST und dem Kernspeicher übertragen.

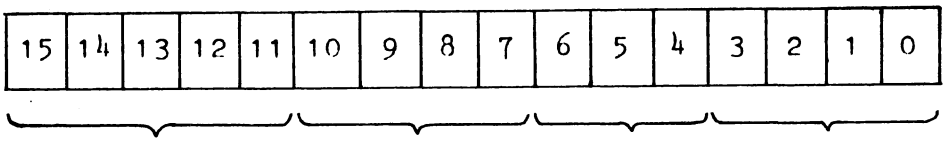
Von der E/A-Schnittstelle führen 32 Leitungen zu den Steuerungen der peripheren Geräte:

8 Leitungen (≙ 1 Byte) dienen zur Anwahl der verschiedenen peripheren Steuerungen (PST).
8 Leitungen dienen zur Eingabe; 8 weitere Leitungen zur Ausgabe von Datenbytes. Über eine Interrupt-Leitung kann eine PST eine Programmunterbrechung (vgl. 1.7.6.1.) Ursache (3)) anmelden.
Eine Leitung ("Anlage rücksetzen") dient zur Übertragung eines Signals (z.B. bei der Spannungseinschaltung), welches die peripheren Steuerungen in eine definierte Anfangslage bringt.
Zwei weitere Leitungen (STROBE 1 und STROBE 2) übertragen Ein/Ausgabe-Steuersignale.
Vier zusätzliche Leitungen dienen zur automatischen Steuerung des Datenverkehrs über den Schnellkanal.

Die Peripheriebefehle der Z 43 sind Kurzbefehle. Entsprechend der Operationsrichtung werden zwei Gruppen unterschieden.

1. Die PKE-Befehle (Eingabe) operieren von der Schnittstelle zum Standard-Registersatz.
2. Die PKA-Befehle (Ausgabe) operieren vom Standard-Registersatz zur Schnittstelle.

Der Aufbau des Befehlswortes unterscheidet sich vom Aufbau der anderen Internbefehle:



In den folgenden symbolischen Bezeichnungen PKEz,r bzw. PKAz,r soll z der Operations-Zusatz-Code sein (Darstellung analog zum Assemblercode).

1. PKE-Befehle:

PKE4,r: Von den Eingabe-Leitungen der E/A-Schnittstelle wird ein Byte in das niederwertige Byte von Register r übernommen. Das höherwertige Byte von Register r bleibt unverändert. Es wird keines der Signale STROBE 1 oder STROBE 2 an die angewählte PST gesendet.

PKE0,r Wie bei PKE4,r; jedoch wird das Steuersignal STROBE 1 an die angewählte PST gesendet.

PKE1,r Wie bei PKE4,r; jedoch wird das Steuersignal STROBE 2 an die angewählte PST gesendet.

PKE2,r: Durch diesen Befehl wird nur das Bit 2^0 von Register r angesprochen; die anderen Bits bleiben unberührt. Bit 2^0 von Register r wird in L gesetzt, wenn der Schnellkanal tätig ist; sonst wird es in 0 gesetzt.

2. PKA-Befehle:

PKA3,r: Im niederwertigen Byte von Register r wird die Adresse einer PST vorausgesetzt. Durch diesen Befehl wird dann die zugehörige PST angewählt.

PKA4,r: Der Inhalt des niederwertigen Bytes von Register r wird auf die Ausgabe-Leitungen der E/A-Schnittstelle gelegt. An die angewählte PST wird das Signal STROBE 1 gesendet.

PKA5,r: Wie PKA4,r; jedoch wird das Signal STROBE 2 gesendet.

PKA6,r: Der Bytezähler für Datenverkehr über den Schnellkanal wird mit dem Inhalt von Register r geladen.

PKA7,r: Der Adresszähler für Datenverkehr über den Schnellkanal wird mit dem Inhalt von Register r geladen.

2. MAKROBEFEHLE

2.1. X-Befehle

Es handelt sich im allgemeinen um Aufrufe für Unterprogramme (UP), die Bestandteil des Betriebssystems (GBS) sind. Durch den GBS-Generator ist die Möglichkeit gegeben, nur einen Teil der zur Verfügung stehenden Makro-UP in das GBS aufzunehmen. Ist ein durch einen X-Aufruf angefordertes UP nicht Bestandteil des GBS, unter dessen Steuerung das Programm ablaufen soll, so können diese UP auch noch beim Laden in das Programm eingebunden werden.

2.4. Z-Befehle

Z-Aufrufe werden vom Assembler in eine kurze Folge von Grundbefehlen übergetzt; diese Folge wird beim Assemblieren bei jedem Aufruf in das Programm eingebunden.

2.5. W-Befehle

Es handelt sich um Aufrufe für "externe Unterprogramme"; das sind UP, die erst beim Laden in das Programm eingebunden werden.

2.1. X-Befehle

2.1.1. Darstellung im Assemblercode:

$$\underbrace{XX}_{1.}\ \underbrace{.\ .\ .\ .\ .\ ,}_{2.}\ \underbrace{P_1,\ P_2\ .\ .\ .\ .\ .\ ,\ P_n;}_{3.}$$

2. Der Makroname besteht aus maximal 5 alphanumerischen Zeichen (0....9 und A....Z).
Das erste Zeichen des Namens muß ein Buchstabe ≠ X sein.
Werden mehr als fünf Zeichen angegeben, so werden die restlichen vom Assembler überlesen. Solche weiteren Zeichen können z.B. verwendet werden, um die Ausbaustufe eines X-Makros zu kennzeichnen.

3. Im Anschluß an den Makronamen folgen n Parameterangaben ($1 \leq n \leq 8$), die für den Ablauf des Makros erforderlich sind.
Nach dem Namen und nach den Parameterangaben (bis auf d. letzten) muß jeweils ein Komma stehen. Nach dem letzten Parameter folgt ein Semikolon. Trennzeichen vor und nach einem Komma und dem Semikolon werden überlesen.

2.1.2. Erklärung der Parameterangaben:

Parameterangaben sind stets Einfachworte. -
Werden für den Ablauf eines Makros Mehrfachworte benötigt, so wird als Parameterangabe z.B. dessen Anfangsadresse (evtl. auch substituiert angegeben) verlangt.

Im folgenden Beispiel wird erläutert, in welcher Form ein Makrobefehl die verschiedenen möglichen Arten der Codierung von Parameterangaben verwertet:

XMA25B,±370,5,G5,±4(ANF),G±8(END),G76

Name P_1 P_2 P_3 P_4 P_5 P_6

P_1: Parameter = ±370

P_2: Parameter = 5
(Der Makrobefehl muß selbst entscheiden, ob dieser Parameter als Zahl 5 oder als Adressenangabe, im letzten Fall als Registernummer 5 zu verwerten ist).

P_3: Parameter = <Register 5>
(Substituierte Parameterangaben sind immer durch ein vorangestelltes G gekennzeichnet. Der Makro weiß auch hier, wie er den Inhalt vonRegister 5 zu verwerten hat).

P_4: Parameter = 5000±4
(Dabei sei 5000 die zu (ANF) gehörige absolute KSP-Adresse; vgl. 6.1.2. Symbolische Adressierung).

P_5: Parameter = <8000±8>
(8000 sei die zu (END) gehörige absolute KSP-Adresse).

P_6: Parameter = <76>
(Dadurch hat man Zugriff zu festen Zellen des Betriebssystems).

Bei Makrobefehlen, die periphere Geräte benötigen, wird der erste Parameter in Form einer symbolischen Gerätenummer angegeben (eventuell auch substituiert). Symbolische Gerätenummern sind Einfachwort-Zahlen, mit Hilfe der das Betriebssystem Zugriff zu den charakterlichen Daten des peripheren Anschlusses hat. Diese Daten sind im Betriebssystem in der Peripherie-Geräte-Liste (PGL) zusammengefaßt.

2.1.3. Ablage im KSP

Nach dem Assemblieren eines Quellenprogramms und nach dem Laden des vom Assembler erzeugten Quellenstreifens steht das Programmablauffähig im KSP. Für die Ablage von X-Makro-Aufrufen gibt es dabei zwei Möglichkeiten:

1. Der angeforderte Makro sei im GBS der Maschine enthalten.

   Als erstes wird PZWn abgelegt, n mit ($0 \leq n \leq 128$) ist dabei eine Nummer, die dem speziellen Makro-Befehl zugeordnet ist. Die Zuordnung nimmt der Assembler

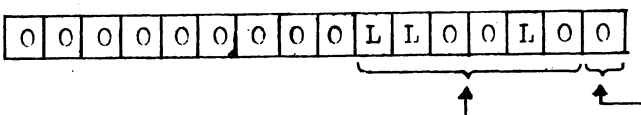
Hilfe der Makronamen-Liste (MNL) vor. Die MNL ist GBS-unabhängig; d.h. die PZW-Nummern n sind für gleiche Makros in jedem GBS gleich.

Im Anschluß an PZWn wird ein Bitmuster (Einfachwort) abgelegt, in welchem notiert wird, in welchem Programmzustand der Makro ablaufen soll, und welche Parameterangaben substituiert sind.

Anschließend folgen die Parameterangaben. Bei der Ablage im Kernspeicher sind symbolische Adressen durch die zugehörigen Adressen ersetzt und gegebenenfalls der (pos. oder neg.) Zuschlag addiert.

Beispiel aus 2.1.2.:

PZW75 — 75 sei n von MA25B



2^0 = 0, da X-Makro

2^i = L, wenn P_i substit.

±370
5
5
5000±4
8000±8
76

2. <u>Der angeforderte Makro sei nicht im GBS der Maschine enthalten</u>

Viele Makro-UP existieren in zwei Versionen; nämlich als

a) UP, das durch den GBS-Generator in das Betriebssystem eingebunden werden kann (Aufruf durch PZWn).

b) UP, das durch den Lader in ein Benutzerprogramm eingebunden werden kann (Aufruf durch F15,0,c).

Mit Hilfe des GBS-Generators kann der Benutzer für sein spezielles GBS unter den vorhandenen Makro-UP (Version a) eine Auswahl treffen. Wird in einem Benutzerprogramm ein X-Makro verwendet, dessen UP nicht im GBS vorhanden ist, so hat der Benutzer noch die Möglichkeit dies beim Laden einzubinden, sofern es auch als unabhängiges UP vorhanden ist.
Der X-Aufruf wird in diesem Fall wie folgt abgelegt:

Auf den ersten vier Bytes wird der Unterprogramm-Sprung F15,0,c abgelegt. c ist dabei die Adresse des vom Lader zusätzlich eingebundenen UP, welches dem Makro-Aufruf entspricht. Anschließend folgen die Parameterangaben wie bei Version 1. Diese Version ist nur möglich, wenn alle Aufrufe dieses UPs keine substituierten Parameterangaben haben und wenn nicht XX codiert wurde (Bitmuster = 0).

2.2. Arithmetische X-Befehle

2.2.1. Sonderaussprung (AFEHL)

Für arithmetische Makrobefehle, die wegen unzulässigen Parametern nicht ablauffähig sind, kann der Benutzer unter der Adresse (AFEHL) eine eigene Fehlerroutine codieren.

Die zu (AFEHL) gehörige absolute Adresse wird vom Lader in die Zelle 96/97 des Benutzerprogramms gebracht. Wird die Marke (AFEHL) nicht deklariert, so legt der Assembler an diese Stelle eine 1 ab.

Tritt in einem Makrobefehl ein solcher Fehler auf, so wird in diesem Fall ein Ersatzwert als Ergebnis eingesetzt bzw. berechnet, auf Platz 90/91 des Benutzerprogramms wird die "normale Rücksprungadresse" und auf Platz 92/93 eine Fehlernummer abgelegt.

Wurde (AFEHL) codiert, so wird ein Sprung an diese Stelle ausgeführt. Wurde (AFEHL) nicht codiert, so wird auch im Fehlerfall der normale Rücksprung ausgeführt.

Fehlerart und Ersatzwert:

<92/93> = 1: Überschreitung des Zahlenbereichs.
Als Ergebnis wird der größtmögliche pos. Wert gesetzt.

<92/93> = 2: Bei $\sqrt{x}$ ist $x < o$.
Es wird $\sqrt{|x|}$ berechnet.

<92/93> = 3: Bei lnx oder lgx ist $x < o$.

<92/93> = 4: Bei lnx ist $x = o$. $\ln|x|$ bzw. $\lg|x|$
Als Ergebnis wird der größtmögliche neg. Wert gesetzt.

Wurde beim Ablauf des Makros kein Fehler festgestellt, dann ist <92/93> = o.

2.2.2. Erklärungen zu den Befehlstabellen

n = 2 Gleitpunktzahl (en) in Doppelwortformat
n = 3 " " Dreifachwortformat

n = 3 Festpunktzahl (en) in Dreifachwortformat
n = 4 " " Vierfachwortformat

P_1	Anfangsadresse des 1. Operanden	entweder Registernummer oder symb. Adresse mit und ohne Zuschlag oder substituierte Angabe
P_2	" " 2. "	
P_E	" " Ergebnisses	

Anmerkung: Gleitpunktzahlen werden - wie in 4.2.1. beschrieben - stets normalisiert dargestellt. Hierzu ist bei den arithmetischen Operationen in Abhängigkeit von den Operanden eine gewisse Anzahl von Normalisierungsschritten nötig. Mit zunehmender Anzahl kann die Genauigkeit des Ergebnisses herabgesetzt werden (z.B. bei der Subtraktion nahezu gleichgroßer Operanden). Damit ein Benutzer auf solche Grenzfälle Bezug nehmen kann, wird nach jeder Normalisierung die Anzahl der Normalisierungsschritte auf Platz 94/95 des Benutzerprogramms abgelegt.

Bei Festpunktzahl-Divisionen wird der Rest in Zelle 94/95 des Benutzerprogramms abgelegt (vgl. 2.2.4.).

2.2.3. Gleitpunktoperationen (n=2 oder n=3)

Operation	Befehl	AFEHL	Bemerkungen
Addition	XGLAn,P_1,P_2,P_E	ja	
Subtraktion	XGLSn,P_1,P_2,P_E	ja	
Multiplik.	XGLMn,P_1,P_2,P_E	ja	
Division	XGLDn,P_1,P_2,P_E	ja	
Qu.Wurzel	XGLWn,P_1,P_E	ja	
Sinus Cosinus Tangens Cotangens	XTRIn,K,P_1,P_E	ja	K sin cos tan cot Altgrad 0 1 2 3 Bogenmaß 4 5 6 7 Neugrad 8 9 10 11
e^x	XEXPn,P_1,P_E	ja	
$\ln x$	XLNn,P_1,P_E	ja	
$\lg x$	XLGn,P_1,P_E	ja	
$\arctan \frac{\Delta y}{\Delta x}$	XARCn,K,P_1,P_2,P_E	ja	1.Op.=Δy=Gegenkath. 2.Op.=Δx=Ankathete Ergebnis in ~ ~ Altgrad für 0 ~ Bogenmaß " 1 ~ Neugrad " 2
Umwandlung	XFGLn,P_1,P_E	nein	FPZ-Doppelwort in GPZ-n-fach-Wort
Umwandlung	XGLFn,P_1,P_E	ja	GPZ-n-fach-Wort in FPZ-Doppelwort

2.2.4. Festpunktoperationen

Operation	Befehl	AFEHL	Bemerkungen
Kompl.	XKOMm,P_1,P_E	nein	m=3 Dreifachwort m=4 Vierfachwort
Addition	XADDm,P_1,P_2,P_E	nein	m=3 Dreifachwort m=4 Vierfachwort
Subtraktion	XSUBm,P_1,P_2,P_E	nein	m=3 Dreifachwort m=4 Vierfachwort
Multiplik.	XMUL22,P_1,P_2,P_E	nein	2W.2W = 4W
Division	XDIV42,P_1,P_2,P_E	ja	4W:2W = 2W und Rest R
Multiplik.	XMUL31,P_1,P_2,P_E	nein	3W.1W = 4W
Division	XDIV41,P_1,P_2,P_E	ja	4W:1W = 3W und Rest R

2.3. Ein/Ausgabe-X-Befehle

2.3.1. Allgemeines

2.3.1.1. Grundperipherie

Das hier beschriebene System ist in der ersten Ausbaustufe für folgende Geräte vorgesehen:

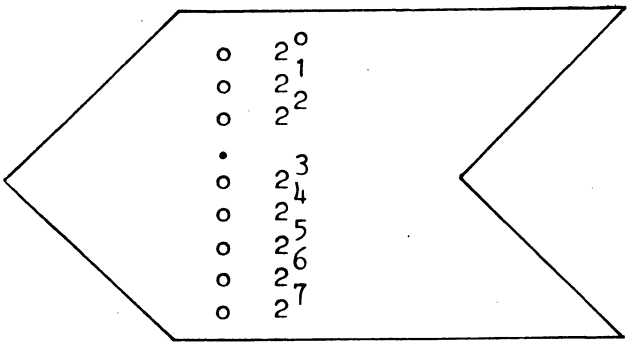
Bedienungsfernschreiber: 5-Kanal Ein- und Ausgabe im CCIT-Code über den Multiplex-Kanal

Lochstreifen-Leser: 5- und 8-Kanal Eingabe binär oder im CCIT-Code über Multiplex- und Schnellkanal
Typen:
Siemens-Leser 1200 u. 38
(1200 u. 120 Z/sec)

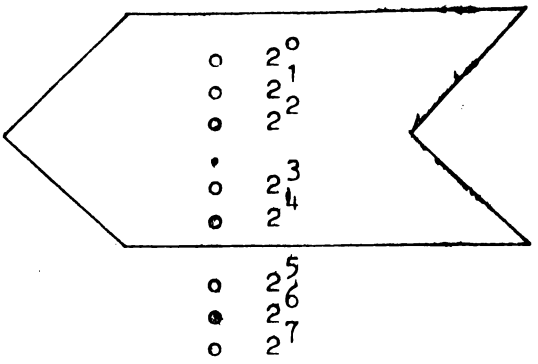
Lochstreifen-Stanzer: 5-und 8-Kanal Ausgabe binär oder im CCIT-Code über Multiplex-Kanal.
Typen:
Siemens-Stanzer 158
(max. Datenrate 150 Z/sek)
Siemens-Stanzer 38
(max. Datenrate 30 Z/sek.)

Stellen-Zuordnung zwischen den Lochungen eines Lochstreifens und der Maschinen-Darstellung:

8-Kanal-LS:



5-Kanal-LS:



2.3.1.2. Codeumsetzung

Zur internen Verarbeitung von Zeichen wird der 7-Bit-Iso-Code verwendet (vgl. 6.1.1.), während die Grund-Peripherie in der Regel CCIT-Code-Zeichen anbietet, so daß im allgemeinen eine Code-Wandlung notwendig ist.

2.3.1.3. Puffer

Innerhalb eines Benutzerprogramms sind für die Ein- und Ausgabe ein oder mehrere Bereiche als Ein/Ausgabe-Puffer freizuhalten. Die Anfangsadressen und Längen solcher Puffer kann der Benutzer innerhalb seines Programms frei wählen. Solche Puffermerkmale sind in einem sogenannten Pufferetikett einzutragen. Die Anfangsadresse eines solchen Pufferetiketts ist wiederum frei wählbar, sie wird beim Makro-Aufruf als Parameter angegeben.

2.3.1.4. Sonderaussprünge der Makros

Falls ein Gerät "nicht betriebsklar" meldet, erfolgt der Rücksprung aus dem Makro nach ‹98/99› des Benutzerprogramms.

Zur Bearbeitung eines solchen Fehlers muß der Benutzer eine entsprechende Fehlerroutine codieren. Die Startadresse dieses Fehlerprogramms ist mit der Marke (PFEHL) zu versehen. Beim Assemblieren wird die zugehörige Adresse an die Stelle 98/99 des Benutzerprogramms abgelegt.

Wird kein Fehlerprogramm codiert, so legt der Assembler auf 98/99 eine 1 ab. In diesem Fall wird das Programm bei "nicht betriebsklar" ohne Meldung auf dem BFS abgebrochen.

2.3.2. Ausgabe

Durch Makroaufrufe werden Informationen unter Beachtung entsprechender Druckmasken ausgabegerecht in den Puffer abgelegt. Nach Füllen des Puffers (entspricht Druckzeile) wird dieser durch einen weiteren Makroaufruf auf das gewünschte Gerät ausgegeben.

2.3.2.1. Pufferetikett

a	l	p	n

Adresse: e e+2 e+4 e+6

a = Anfangsadresse des Puffers
l = gesamte Bytelänge des Puffers (l=1,2,3,...)
p = Pufferpunkt (p=1,2,3,...)
n = Anzahl der vor Ausgabe der Zeile gewünschten ZL (ein WR erfolgt durch den Makro).

Die Angaben a,l,n sind vor dem Makro-Aufruf zu definieren.

Der Pufferpunkt p wird vom Makro gesetzt, dabei ist p die Nummer des am weitesten rechts gelegenen Pufferbytes, welches belegt ist. Vor dem Aufbereiten eines Puffers muß p=0 sein (nach den Makro XBLANK und XAUSCO steht p automatisch auf 0).

Das Pufferetikett kann beim Einlesen oder per Programm aufgebaut bzw. verändert werden.

Beispiel für das Ablochen eines Pufferetiketts und des zugehörigen Puffers:

```
(ETIK)      0(PUFF)     = a
            80          = l
            0           = p
            3           = n
            .
            .
            .
            .
(PUFF)      /80
```

(vgl. dazu 6.1.)

2.3.2.2. Puffer-BLANKEN

XBLANK,e,p

Funktion:
1. Ab <e> werden p Bytes mit der Zahl 32 (= ISO-Bandwert von ZW) belegt, jedoch maximal 1 Bytes
2. Anschließend wird <e+4> = p gleich Null gesetzt.

2.3.2.3. Puffer-Vorbereiten

XPUVORk,e,i,m

k = Aussage über die Ausbaustufe des Makros (vgl. 2.3.2.4.)
e = Anfangsadresse des Pufferetiketts
i = Anfangsadresse der zur Ausgabe vorzubereitenden Information
m = Anfangsadresse der Druckmaske (vgl. 6.1.6.)

Funktion: 1. Durch diesen Makro wird die ab Adresse i stehende Information (Zahl) als ausgabegerechtes Druckbild entsprechend der Druckmaske ab Position d (vgl. 6.1.6.) im ISO-Code abgelegt.

2. Der Pufferpunkt <e+4>=p wird eventuell erhöht.

3. Der Puffer wird bis maximal p=1 geladen, ein eventueller Überlauf geht verloren.

2.3.2.4. Ausbaustufen von XPUVORk,e,i,m

k ist eine der Ziffern 0,1,2,3,4 oder 5 und ist Bestandteil des Makronamens! k gibt die erforderliche Ausbaustufe des Makros im verwendeten Betriebssystem an. Beim Assemblieren wird k zwar überlesen, aber trotzdem ist die Angabe von k beim Codieren zweckmäßig, um an der Codierung von Programmen die notwendige Ausbaustufe zu erkennen.

k	FPZ 1W	FPZ 2W	Text	binär	GPZ 2W	GPZ 3W	FPZ 3W	FPZ 4W
0	+	+	+	+	+	+	+	+
1	+	+	+	+	+	+	-	-
2	+	+	+	+	+	-	-	-
3	+	+	+	+	-	+	-	-
4	+	+	+	+	-	-	+	+
5	+	+	+	+	-	-	-	-

\+ in der Ausbaustufe k enthalten
\- in der Ausbaustufe k nicht enth.

2.3.2.5. Puffer-Ausgabe (mit Codewandlung)

XAUSCO,g,e

g = symbolische Gerätenum. (vgl. 2.1.2.)

e = Anfangsadresse des Pufferetiketts

Funktion:

1. Es erfolgt die Ausgabe von einem WR und <e+6> = n Zeilenvorschüben. (Ist n = 0, so erfolgt nur ein WR)
2. Byteweise Ausgabe des Puffers bis zum Pufferpunkt p, wobei die gepufferten Iso-Zeichen in dem CCIT-Code gewandelt werden.
3. Nach der Ausgabe wird <e+4> = p gleich Null gesetzt.
4. Im Falle eines Gerätefehlers erfolgt der Rücksprung nach<98/99>; vgl. 2.3.1.4.

2.3.2.6. Puffer-Ausgabe (binär)

XAUSBI,g,a,l

g = symbolische Gerätenum. (vgl. 2.1.2.)

a = Anfangsadresse des Puffers

l = Bytelänge des Puffers

Funktion:

1. Durch diesen Makro wird der durch a und l definierte Puffer byteweise ausgegeben. (Dieser Makro setzt also kein Pufferetikett voraus).
2. Im Falle eines Gerätefehlers erfolgt der Rücksprung nach <98/99>; vgl. 2.3.1.4.

2.3.3. Eingabe

Durch einen entsprechenden Makroaufruf wird ein Puffer vom gewünschten Gerät her byteweise geladen. Durch weitere Makroaufrufe werden die eingelesenen Informationen entschlüsselt und an die gewünschten Arbeitsspeicherstellen abgelegt.

2.3.3.1. Pufferetikett

a	l	p	w

Adresse: e e+2 e+4 e+6

a = Anfangsadresse des Puffers
l = gesamte Bytelänge des Puffers
p = Pufferpunkt
w = 0 oder Iso-Bandwert des Endekennzeichens (vgl. 2.3.3.2.)

2.3.3.2. Puffer-Eingabe (mit Codewandlung)

XEINCO,g,e

g = symbolische Gerätenum.
e = Anfangsadresse des Pufferetiketts

Funktion: 1. Der Puffer wird vom Gerät g her über den MPX-Kanal ab p+1 geladen. Bei erstmaliger Puffereingabe muß <e+4> = p gleich 0 sein.

2. Jeder gelesene CCIT-Bandwert wird in den entsprechenden Iso-Bandwert umgewandelt. Dabei dient <e> = a gleichzeitig als Zi/Bu-Weiche. Und zwar steht bei Zi das Bit 2^0 in 0, bei Bu in L. Nur wenn der zu lesende Datenstreifen zu Beginn kein Zi-oder Bu-Zeichen trägt, muß der Benutzer a gerade bzw. ungerade machen.

3. Entspricht ein gelesenes Zeichen dem Wert <e+6> = w, so wird dieses Zeichen als letztes abgelegt.

4. Ist <e+6> = w gleich 0, so wird keine Abfrage auf Endekennzeichen durchgeführt.

5. Im Falle eines Gerätefehlers erfolgt der Rücksprung nach <98/99>; vgl. 2.3.1.4.

6. Nach der Eingabe wird <e+4> = p in 0 gesetzt.

7. Die Ausführung des "normalen Rücksprunges" wird nach Behandlung von XPUENT in dem gesonderten Abschnitt 2.3.3.4. erläutert.

2.3.3.3. Puffer entschlüsseln

XPUENTk,e,i,f

k = Aussage über die Ausbaustufe des Makros (vgl. 2.3.3.5.)

e = Anfangsadresse des Pufferetiketts

i = Anfangsadresse, ab der die entschlüsselte Information gespeichert werden soll

f = Rücksprungadresse bei Erreichen des Sonderzeichens <e+6> = w und bei unerlaubten Zeichen.

Delivery or duplication of this document and the use or communication of the contents thereon are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Funktion:

1. Ab p+1 wird die nächste Information entschlüsselt, wobei Trennzeichen vor der Information überlesen werden. Informationsendekennzeichen sind: Trennzeichen, das Sonderzeichen <e+6> = w, unerlaubte Zeichen bei Zahlen und Textendezeichen.

2. Die entschlüsselte Information wird ab <e> = i abgelegt und der Pufferpunkt wird neu gesetzt.

3. Nach dem Abspeichern erfolgt der "normale Rücksprung", wenn Zahlen durch Trennzeichen und Text durch Textendekennzeichen abgeschlossen waren.
   Dabei wird im Parameterfeld notiert:
   In 64/65 der Informationstyp,
   in 66/67 die Anzahl der abgespeicherten Worte.

4. Beim Lesen unerlaubter Zeichen und des Zeichens w erfolgt der Rücksprung nach Adresse f. Dabei wird im Parameterfeld notiert:

   $$<68/69> = \begin{cases} 0 \text{ beim Lesen von w.} \\ \bar{w} \text{ Bandwert des gelesenen unerlaubten Zeichens.} \end{cases}$$

   <70/71> = "Normale Rücksprungadresse"

   Ferner wird unterschieden, ob w bzw. $\bar{w}$ auf ein Trennzeichen folgte oder nicht. Stand vor w bzw. $\bar{w}$ ein Trennzeichen (also keine Information), so wird <64/65> = <66/67> = 0 gesetzt.

   Stand vor w bzw. $\bar{w}$ eine Information (also ohne Trennzeichen), so wird diese ent-

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

schlüsselt und abgespeichert und in 64/65 der Informationstyp und in 66/67 die Anzahl der abgespeicherten Worte notiert.

Übersichtstabelle

<64/65> = 0 wenn w bzw. $\bar{w}$ allein gelesen
1 wenn FPZ gelesen
2 wenn GPZ gelesen
3 wenn Text gelesen

<66/67> = Anzahl der gespeicherten Worte

<68/69> = 0 wenn w gelesen
$\bar{w}$ = ISO-Bandwert des unerlaubten Zeichens

<70/71> = "Normale Rücksprungadresse

} wird nur beim Sonderaussprung f gesetzt

Bemerkungen:

1. Innerhalb von Text sind alle Zeichen (auch w) außer Textendekennzeichen zulässig.

2. Mit dem geschilderten Eingabesystem hat man z.B. die Möglichkeit, ein Datenfeld blockweise abzuarbeiten, indem man z.B. nach jedem Block ein unerlaubtes Zeichen als ein $\bar{w}$ (z.B. "/") und als Dateiende-Kennzeichen (z.B. "×") als w wählt. Da bei w und $\bar{w}$ der Sonderaussprung nach f erfolgt, kann an dieser Stelle f eine entsprechende Programmverzweigung vorgenommen und vermöge <70/71> das Programm fortgesetzt werden.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

2.3.3.4. Zusammenhang zwischen XPUENT und XEINCO

"Normaler Rücksprung"

Normalerweise werden die beiden Makroaufrufe in der Reihenfolge

```
⋮
XPUENTk,e,i,f
XEINCO,g,e
⋮
```

direkt nacheinander codiert.

Nach dem Assemblieren und Laden des Programms (vgl. 6. und 7.) steht dafür folgendes im KSP:

Adresse	Ablage
λ	$PZWn_1$
$\lambda+2$	BIM
$\lambda+4$	e
$\lambda+6$	i
$\lambda+8$	f
$\lambda+10$	$PZWn_2$
$\lambda+12$	BIM
$\lambda+14$	g
$\lambda+16$	e

BIM = von Assembler gesetzte Anweisung an das Betriebssystem

$PZWn_1$ ≙ XPUENT

$PZWn_2$ ≙ XEINCO

... contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Ablauf:

1. Beim <u>erstmaligen</u> Aufruf dieser Befehlsfolge kann der Einsprung entweder bei λ oder bei λ+10 erfolgen.
   - a) Bei Einsprung λ muß <e+4>=p=1 sein.
   - b) Bei Einsprung λ+10 muß <e+4>=p=0 sein. (Dieser Einsprung λ+10 erspart Zeit).

2. Ablauf von XPUENT:
   - a) Nächste Information vollständig im Puffer : Entschlüsseln und speichern dieser Information. "Normaler Rücksprung" nach λ+18 oder Sonderaussprung nach f.
   - b) Nächste Information nicht vollständig im Puffer:
     Der vorhandene Teil der Information wird an den Pufferanfang gebracht.
     Der Rücksprung erfolgt nach λ+10, dadurch wird der Puffer nachgeladen.

3. Ablauf von XEINCO:
   Laden des Puffers. Der "normale Rücksprung" erfolgt stets nach λ.

2.3.3.5. <u>Ausbaustufen von XPUENTk,e,i,f</u>

Hier gilt das gleiche wie in 2.3.2.4.; lediglich die Spalte "binär" entfällt.

2.3.3.6. <u>Puffer-Eingabe (binär)</u>

XEINBI,g,a,l

g = symbolische Gerätenum.
a = Anfangsadresse des Puffers
l = Bytelänge des Puffers

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Funktion: 1. Durch diesen Makro wird der durch a und l definierte Puffer über das Gerät g byteweise gefüllt. (Dieser Makro setzt also kein Pufferetikett voraus).

2. Im Falle eines Gerätefehlers erfolgt der Rücksprung nach <98/99>; vgl. 2.3.1.4.

2.3.4. Verwendung von Y-Befehlen.

Mit Hilfe der Steuerbefehle YSTART,n,a,b (vgl. 3.4.1.) und YWARTE,n kann ein Programmsystem auf mehrere Prioritäten (Plätze in der Programmwarteschlange) aufgeteilt werden. Ein Teilprogramm wählt man rechenintensiv und andere Ein/Ausgabe-intensiv. Auf diese Weise wird eine bessere Ausnutzung der Maschinenzeit für das vorliegende System erreicht, da die Ein/Ausgabe zeitmultiplex zum eigentlichen Rechenprogramm abläuft. -

2.3.5. Die Makrobefehle

XAUSCO,g,e
XAUSBI,g,a,l
XEINCO,g,e
XEINBI,g,a,l

sind in jedem Grundbetriebssystem (GBS) enthalten.

All rights reserved

2.4. Z-Befehle

Z-Befehle dienen zur bequemeren Codierung von Programmen im Assembler-Code. Sie werden vom Assembler in eine kurze Folge von Internbefehlen, die fest in das Programm eingebunden wird, übersetzt; bei jedem Auftreten eines Z-Befehls wird diese Befehlsfolge vom Assembler an Stelle des Z-Befehls abgespeichert.

2.4.1. Darstellung im Assemblercode

Die Codierung beginnt mit dem Buchstaben Z (als Steueranweisung für den Assembler).
Nach dem Z dürfen maximal 3 Buchstaben als Name des Z-Befehls folgen. Danach folgen die Register- und Parameterangaben, die durch Komma voneinander zu trennen sind. (Vor dem ersten Parameter steht also kein Komma). Ein Trennungszeichen schließt den Aufruf ab.
Bei symbolischen Adressenangaben kann ein positiver oder negativer Zuschlag vorliegen. Die angesprochene Marke muß aber bereits definiert sein, wenn sie innerhalb eines Z-Befehls vorkommt.
Ist eine Registerangabe ≥ 16, so wird sie modulo 16 verwertet. Eine ganze Zahl n als Parameterangabe darf positiv oder negativ sein; sie muß Einfachwort-Format haben.

2.4.2. Erläuterungen zur folgenden Tabelle

1. Spalte: Abzulochender Aufruf

a = Registerangabe (≤ 15)
b = Registerangabe (≤ 15)
c = symbolische oder numerische Kernspeicher-Adressangabe
n = Einfachwort-Festpunktzahl

2. Spalte: Befehlsfolge, die der Assembler für den Aufruf in Spalte 1 ablegt.

3. Spalte: Anzahl der abzulegenden Worte.

4. Spalte: Wirkung bei der Ausführung.
(Bei Divisionsbefehlen hat der Rest immer das Vorzeichen des Dividenden).

2.4.3. Übersicht über die vorhandenen Z-Befehle

Z-Befehl	Übersetzung	Worte	Ausführung	
ZMa,b	Ma,b CRRa,o	2	<a> . <b> → a+1,a	
ZCMa,n	LCBa+1,0,n Ma,a+1 CRRa,0	4	<a> . n → a+1,a	
ZLMa,b,c	Ma,b,c CRRa,o	3	<a> . <<b> + c> → a+1,a	
ZDa,b	AAa,a Da,b	2	<a+1,a> : <b> → a, Rest → a+1	
ZCDa,n	LCBa-1,o,n AAa,a Da,a-1	4	<a+1,a> : n → a, Rest → a+1 n → a-1	
ZLDa,b,c	Ba-1,b,c AAa,a Da,a-1	4	<a+1,a> : <<b>+c> → a Rest → a+1 <<b>+c> → a-1	
ZGAa,b,n	EUa,15,10 Ab,15,2 CA15,2 n	6	wenn <a> = 0	dann <b> +n → b (bedingte Addition)
ZNAa,b,n	CSPa,0 CA15,6 n Ab,15,-6	5	wenn <a> < 0	
ZPAa,b,n	CSPa,0 Ab,15,2 CA15,2 n	5	wenn <a> ≥ 0	

2.5. W-Befehle

2.5.1. Darstellung im Assemblercode

WABCDE,P_1,P_2,...,P_n;

Es gelten weitgehend die gleichen Ablochvorschriften wie für X-Befehle (vgl. 2.1.1.).

Der Name besteht aus maximal 5 alphanumerischen Zeichen. Das erste Zeichen des Namens muß ein Buchstabe sein. (Dabei ist auch X zulässig).

Als Parameterangaben sind Einfachwortzahlen und symbol. Adressen (mit oder ohne Zuschlag) zulässig. Substituierte Parameterangaben (durch G gekennzeichnet) sind nicht zulässig. Sind keine Parameterangaben verlangt, so folgt nach dem Namen (eventuell durch Trennzeichen getrennt) das Semikolon.

2.5.2. Verwendung von W-Befehlen

Durch W-Befehle werden "externe Unterprogramme" aufgerufen; das sind UP, die erst beim Laden in das Programm eingebunden werden. Ein W-Aufruf WABCDE,P_1, P_2,...,P_n; wird nach dem Assemblieren und Laden abgelegt als:

F15,0,c
P_1
P_2
.
.
.
P_n

c ist dabei die absolute KSP-Adresse des vom Lader zusätzlich eingebundenen UPs. Dieses UP trägt im

Programmkopf den Namen (ABCDE). Die Anzahl der Parameterangaben $P_1,\ldots,P_n$ ist nicht begrenzt; sie muß übereinstimmen mit der Anzahl der Parameter, die das Programm mit dem Namen (ABCDE) verlangt.

Die Parameter ($P_1,\ldots,P_n$), die das UP benötigt, holt es sich aus dem übergeordneten Programm ab. Den Rücksprung muß das UP auf die P_n folgende Zelle des übergeordneten Programms ausführen.

3. STEUERBEFEHLE (Y-BEFEHLE)

3.1. Darstellung im Assemblercode

YABCDE,$P_1,\ldots,P_n$;

Vor dem eigentlichen Namen des Steuerbefehls steht das Steuerzeichen (für den Assembler) Y.

Der Name des Steuerbefehles besteht aus maximal fünf alphanumerischen Zeichen (aus 0....9, A....Z). Das 1. Zeichen des Namens muß ein Buchstabe sein. (Auch X ist dabei zulässig).

Im Anschluß an den Namen folgen n Parameterangaben ($0 \leq n \leq 3$), die für den Ablauf des Steuerbefehls erforderlich sind. Als Endekennzeichen des Y-Aufrufes muß ein Semikolon abgelocht werden. Der Name und die einzelnen Parameterangaben werden durch Kommata voneinander getrennt. Trennzeichen vor und nach einem Komma und dem Semikolon werden überlesen. Wenn kein Parameter anzugeben ist, folgt nach dem Namen (eventuell durch Trennzeichen getrennt) das Semikolon.

Parameterangaben sind auch hier grundsätzlich Einfachworte. Parameterangaben können Einfachwort-Festpunktzahlen oder symbolische Adressen (mit oder ohne Zuschlag) sein. Substituierte Parameterangaben sind nicht zugelassen.

3.2. Ablage im Kernspeicher

Die Y-Befehle werden vom Assembler PZWn übersetzt, jedoch mit $128 \leq n < 255$. Nach dem PZWn werden die Parameter abgelegt.
(Ein Bitmuster wie bei den X-Befehlen wird nicht aufgebaut, da Parameter nicht substituiert angegeben werden können).

3.3. Ablauf.

Y-Befehle laufen analog zu den X-Makros ab.
Y-Befehle können jedoch nur im Programmzustand 2 ablaufen (bei Y-Befehlen handelt es sich meist nur um kurze Befehlsfolgen). Y-Befehle können in ihrem Ablauf also nicht unterbrochen werden.

3.4. Liste der Y-Befehle

Mit Y-Befehlen kann der Benutzer Anweisungen an das ~~Betriebssystem~~ geben. Unter anderen stehen folgende Y-Befehle zur Verfügung:

3.4.1. YSTART,n,a,b

Das Programm mit der rel. Priorität n wird gestartet

bzw. fortgesetzt.

a=0 Das laufende Programm soll fortsetzbar sein

a≠0 Das laufende Programm wird als nicht fortsetzbar gekennzeichnet. (Fortsetzung nur durch einen anderen YSTART-Aufruf)

b=0 Es werden keine Parameter bzw. gemeinsame Datenfelder benötigt

b≠0 Adresse der Parameter bzw. gemeinsamen Datenfelder

Wenn das Programm mit der Priorität n noch nicht gestartet oder bereits beendet wurde, wird es gestartet. War es bereits gestartet, so wird untersucht, ob es durch einen YSTART-Aufruf angehalten wurde. War dies der Fall, so wird es fortgesetzt. Andernfalls wird eine Fehlermeldung ausgegeben, das aufrufende Programm wird abgebrochen und ein fortsetzbares Programm wird gesucht.

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

3.4.2. YSTOP

Programm-Ende.
Das Programm wird in der Programmwarteschlange (PWS) als beendet eingetragen und ein fortsetzbares Programm wird gesucht. Sämtliche durch dieses Programm eingetragene Sperren werden gelöscht (siehe 3.4.7.)

3.4.3. YHALT

Das Programm wird angehalten und ein fortsetzbares Programm wird gedruckt.

3.4.4. YFEHL,m,a

Fehlerprotokoll ausgeben.
Es wird ein Fehlerprotokoll in der Form F m n über den Bedienungsfernschreiber ausgegeben (n=Programm-Prio.).

a=0 Das Programm wird fortgesetzt.

a≠0 Das Programm wird abgebrochen und ein fortsetzbares Programm gesucht.

3.4.5. YANW,m

Anweisung ausgeben.
Es wird eine Anweisung in der Form A m n über den Bedienungsfernschreiber ausgegeben (n=Programm-Prio.).
Das Programm wird erst fortgesetzt, wenn der Operator ein Quittungssignal ✠0,n,m (vgl. 8.1.) gegeben hat.

Delivery, duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

3.4.6. YSPERR,g,n

g = Symbolische Gerätenum.

n = RP des Programms, für das die Sperre eingetragen wird. RP = relative Priorität.

Das periphere Gerät mit dem Namen g wird vom laufenden Programm aus für alle Programme außer Programm n und Bedienungsprogramm gesperrt. Nach Notierung der Sperre wird das Programm fortgesetzt. Kann die Eintragung nicht gemacht werden, weil bereits eine Sperre von einem anderen Programm her vorliegt, so wird die Rückkehradresse soweit zurückgesetzt, daß der Sperr-Aufruf bei Programmfortsetzung nochmal erfolgt, und ein fortsetzbares Programm wird gesucht.

Wenn sich mehrere Programme gegenseitig aufrufen, so dürfen YSPERR-Aufrufe nur von dem Programm aus gegeben werden, welches als übergeordnetes Programm durch Bedienungsaufruf gestartet wird.

3.4.7. YFREI,g

g = Symbolische Gerätenum .

Wurde das Gerät g vom laufenden Programm gesperrt, so wird diese Sperre aufgehoben und dieses Programm wird fortgesetzt. Ist gar keine Sperre für das Gerät g eingetragen, so wird der Befehl ignoriert. Liegt für g ein Sperreintrag von einem anderen Programm her vor, so wird das laufende Programm abgebrochen, eine Fehlermeldung ausgegeben und ein fortsetzbares Programm gesucht. -

Sperre aufheben.
Außer durch YFREI werden in folgenden Fällen sämtliche von einem Programm (mit der RP n) ausgelösten Sperr-

Vervielfältigung dieser Unterlage sowie Mitteilung ihres Inhalts an Dritte ohne ausdrückliche Zugeständnisse ist verboten. Zuwiderhandlungen verpflichten zu Schadenersatz. Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

eintragungen gelöscht:

1. Programm n wurde wegen eines Fehlers abgebrochen
2. Programm n wurde durch YSTOP beendet
3. Programm n wurde durch Bedienungsaufruf (siehe 8.3.) beendet.

Sperr-Eintragungen können also nur von dem Programm aus aufgehoben werden, von dem aus die Sperre durch YSPERR eingetragen wurde.

3.4.8. YWARTE,n

Warten auf Ende von Programm n. n= rel.Priorität
Wenn das Programm n beendet ist, wird das aufrufende Programm fortgesetzt. Andernfalls wird die Rückkehradresse zurückgesetzt, um den Wartenaufruf bei Programmfortsetzung zu wiederholen und das Programm n wird, wenn möglich, fortgesetzt.
Wenn Programm n abgebrochen worden ist, oder noch nicht gestartet worden ist, wird eine Fehlermeldung ausgegeben und das laufende Programm wird ebenfalls abgebrochen.

3.4.9. YEINTR,n,m

Eintragung in die PWS vornehmen
n = Position in der PWS (Priorität)
n = Absolute Adresse, an der die Werte (1., 2. und 4. Wort der PWS), die in die PWS eingetragen werden sollen, stehen.

Dieser Aufruf wird innerhalb des Laders (vgl. 7.) benötigt.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten.

Copyright and duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

4. ZAHLENFORMATE

4.1. Festpunktzahlen (FPZ)

4.1.1. Einfachwort-Festpunktzahlen

Einfachworte werden in einem Wort (16 Bit) binär verschlüsselt abgelegt. Negative Zahlen werden im Zweierkomplement notiert. Die oberste Stelle gibt das Vorzeichen an.

0 kennzeichnet positive Zahlen

L kennzeichnet negative Zahlen

Es können also alle Festpunktzahlen z mit $|z| \leq 2^{15} - 1 = 32767$ in einfacher Wortlänge dargestellt werden.

4.1.2. Doppelwort-Festpunktzahlen

Doppelworte werden in zwei benachbarten Speicherzellen (bzw. Registern) binär verschlüsselt dargestellt. In der Speicher- bzw. Registerzelle mit der niederen Adresse werden die Bits mit dem niederen Gewicht abgelegt. Diese Zelle wird bei Doppelwortbefehlen (vgl. 1.5.) auch als Adresse des zu bearbeitenden Doppelwortes angegeben und muß durch vier teilbar sein. Negative Zahlen werden im Zweierkomplement notiert. In doppelter Wortlänge können alle Festpunktzahlen z mit $|z| \leq 2^{31} - 1 = 2\ 147\ 483\ 647$ dargestellt werden.

Copyright reserved. Duplication of this document and the use or communication of its contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

4.1.3. Ablochvorschriften für FPZ

Eine Festpunktzahl z mit $|z| \leq 32\,767$ wird durch den Assembler in einem Einfachwort abgelegt, wenn sie ohne führende Null abgelocht ist; wird sie mit einer oder mehreren führenden Nullen abgelocht, so wird sie in einem Doppelwort abgelegt. Eine Festpunktzahl z mit $32\,767 < |z| \leq 2\,147\,483\,647$ wird grundsätzlich in einem Doppelwort abgelegt.

Eine Einfachwort-Null muß als 0 abgelocht werden
Eine Doppelwort-Null muß als 0 0 abgelocht werden.

4.2. Gleitpunktzahlen (GPZ)

4.2.1. Doppelwort-Gleitpunktzahlen

Darstellung in der Maschine:

Bit 2^{31} : Vorzeichenstelle. Dieses Bit markiert das Vorzeichen; die Mantisse negativer Zahlen wird nicht komplementiert dargestellt.

Bit $2^{30} - 2^{24}$: Exponent zur Basis 2. Der effektive Zweierexponent e errechnet sich aus der in der Maschine stehenden Zahl durch Subtraktion von 64.
Es gilt also $-64 \leq e \leq 63$.

Bit $2^{23} - 2^{0}$: Mantisse (24 Bit). -

Die Mantisse wird stets so errechnet, daß als oberstes Bit 2^{-1} in L steht (Normalisierte Darstellung). Dieses Bit wird aber nicht mitgespeichert, so daß also effektiv

25 Mantissenbits zur Verfügung stehen. Die Mantissenstellen haben also von links nach rechts die Wertigkeiten 2^{-2}, 2^{-3}, ... 2^{-25}.

Zahlenbereich:
Es können Doppelwort-Gleitpunktzahlen z mit
$2^{-64} \leq |z| \leq 2^{63}$ dargestellt werden.

Dies entspricht dem dezimalen Bereich:

$0.5421010 \cdot 10^{-19} \leq |z| \leq 0.9223372 \cdot 10^{19}$

Ablochvorschriften:

Eine Doppelwort-Gleitpunktzahl (GPZ-2W) darf in halblogarithmischer Darstellung mit Trennung der Mantisse und eines (maximal zweistelligen) Dezimalexponenten durch E abgelocht werden. Um als GPZ-2W gekennzeichnet zu sein, muß sie entweder den Dezimalpunkt, das E oder beides enthalten. Positives Vorzeichen der Mantisse und des Exponenten können weggelassen werden; negative müssen jeweils davor abgelocht werden.

Nach führenden Nullen werden bis zu 14-stellige Zahlen eingelesen; die Genauigkeit wird (durch Abschneiden) auf 7 Mantissenstellen reduziert. Werden mehr als 14 echte Mantissenstellen abgelocht, so wird die Zahl völlig verfälscht, da die oberen Stellen (durch Herausschieben) verloren gehen.

Eine Zahl, die kleiner ist als im Zahlenbereich angegeben, wird als Gleitkomma-Null (alle Bits des Doppelwortes gleich Null) eingelesen.

Beispiel:
Die Zahl 471000 kann z.B. auf folgende Arten als GPZ-2W eingelesen werden:

```
471000.
471000E
0.4710E6
4710000.E-1
+4.7100000E5
```

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

4.2.2. Dreifachwort-Gleitpunktzahlen.

Die Maschinendarstellung ist ähnlich wie oben:

Bit 2^{47} : Vorzeichenstelle

Bit $2^{46} - 2^{39}$: Exponent zur Basis 2 (8 Bit)

Bit $2^{30} - 2^{0}$: Mantisse (39 Bit)

Zahlenbereich:
Hier können Gleitpunktzahlen z mit
$2^{-128} \leq |z| \leq 2^{127}$ dargestellt werden.
Das entspricht dem dezimalen Bereich:
$0.293\ 873\ 587\ 70 \cdot 10^{-38} \leq |z| \leq 0.170\ 141\ 183\ 46 \cdot 10^{39}$

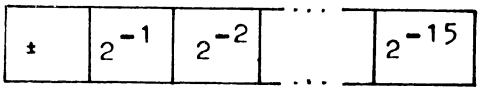
Ablochvorschriften:

Eine Dreifachwort-Gleitpunktzahl (GPZ-3W) darf ebenfalls in halblogarithmischer Darstellung mit Trennung der Mantisse und eines Dezimalexponenten durch D abgelocht werden. Um als GPZ-3W gekennzeichnet zu sein, muß sie unbedingt das D enthalten. Bei GPZ-3W werden, wie bei GPZ-2W, 14 Mantissenstellen eingelesen; jedoch beträgt die eingelesene Genauigkeit in diesem Fall 11 Dezimalstellen.

4.3. Bruchzahlen (BRZ)

4.3.1. Einfachwortbruchzahlen.

Ein Einfachwort kann in der Maschine mit folgenden Stellenwertigkeiten als Bruchzahl verstanden werden:



Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Negative BRZ werden im Zweierkomplement notiert.

Zahlenbereich:

Als Einfachwort können Bruchzahlen z mit

$|z| \geq 2^{-15}$ = ,000030517578125 und

$|z| \leq 1-2^{-15}$ = ,999969482421875 abgelegt sein.

Ablochvorschriften:

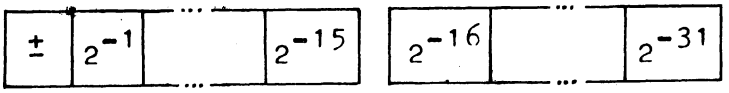
Einfachwort-BRZ müssen mit Dezimalkomma ohne Null davor als Zahlen zwischen -1 und +1 abgelocht werden. Nach dem Komma dürfen maximal 14 Ziffern folgen. In der Maschinendarstellung wird jedoch nur eine Genauigkeit von vier Dezimalen erreicht.

Beispiele: ,412589 oder -,61

Ist der Betrag der abgelochten Zahl größer als $1-2^{-15}$, wird die größtmögliche positive bzw. negative BRZ abgelegt. Ist der Betrag der abgelochten Zahl kleiner als 2^{-15}, so wird die BRZ Null (alle Bits in 0) abgelegt.

4.3.2. Doppelwortbruchzahlen

Ein Doppelwort kann in der Maschine mit folgenden Stellenwertigkeiten als Bruchzahl verstanden werden:



Negative Doppelwort-BRZ werden komplementiert notiert.

Zahlenbereich:

Als Doppelwort können Bruchzahlen z mit

$|z| \geq 2^{-31}$ = 0,000 000 000 465 .... und

$|z| \leq 1-2^{-31}$ = 0,999 999 999 534 .... abgelegt sein.

Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Ablochvorschriften:

Doppelwort-BRZ müssen mit Dezimalkomma mit Null davor als Zahlen zwischen -1 und +1 abgelocht werden. Nach dem Komma dürfen maximal 14 Ziffern folgen.

Hier gilt: $0{,}999\,999\,999 < 1-2^{-31}$

Es wird also etwa eine Genauigkeit von 9 Dezimalstellen in der Maschinendarstellung erreicht.

4.4. Weitere Zahlenformate

Zum Einlesen weiterer Zahlenformate dient das Kennzeichen ":".

Vorgesehen ist das Einlesen von Dreifachwort-FPZ(:3), von Vierfachwort-FPZ(:4) und von vierstelligen Hexadezimalzahlen (:2). **und Binräzahlen (:1).**

Abzulochen ist jeweils zuerst der Doppelpunkt, dann die Kenn**ziffer** , dann ein Zwischenraum und zuletzt die eigentliche Information.

Durch Verwendung weiterer **Kennziffern (5,6,7,8,9)** ist das System nach Bedarf erweiterungsfähig.

4.4.1. Dreifachwort-FPZ

Ablochvorschrift:	:3␣z
Bereich:	$\lvert z \rvert \leq 2^{47}-1$
Vorzeichen:	+,- oder keines.

4.4.2. Vierfachwort-FPZ

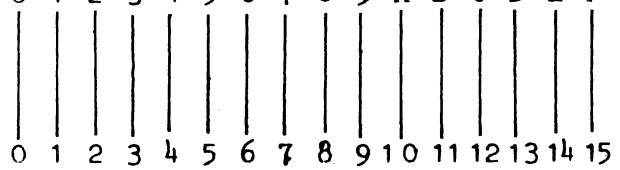
Ablochvorschrift:	:4␣z
Bereich:	$\lvert z \rvert \leq 2^{63}-1$
Vorzeichen:	+,- oder keines

4.4.3. Hexadezimalzahlen

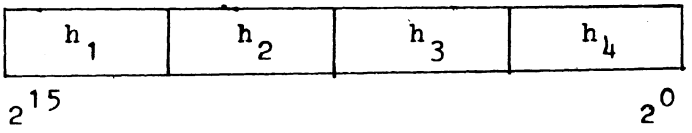
Ablochvorschrift: :2␣$h_1h_2h_3h_4$

Jedes h_i ist eines der Zeichen: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Im KSP werden für jedes h_i 4 Bit belegt mit:



Reihenfolge der KSP-Ablage:



Anmerkung: Will man z.B. die Doppelwort-Hexadezimalzahl F10B84A einlesen, so ist

:2 B84A
:2 F10 abzulochen.

4.4.4. Binärzahlen

Ablochvorschrift: :1␣100101101011

Maximal 16 Bits (1 oder 0) dürfen angegeben werden. Die Bits sind ohne Trennzeichen als Ziffern zu codieren. Weniger als 16 Bits werden im 16-Bit-Wort rechtsbündig abgelegt.

5. TEXTFORMATE

5.1. Darstellung in der Maschine

Text wird byteweise im 7-Bit-Iso-Code (vgl. DIN 66003, Juli 1968) abgelegt. Als Endekennzeichen wird das Iso-Zeichen Apostroph abgelegt. Ein Textanfangs-Kennzeichen wird nicht gespeichert.

5.2. Ablochvorschriften

Als Textanfangs-Kennzeichen (sowohl im Iso- als auch im CCIT-Code) dient der Apostroph.

Als Textende-Kennzeichen (sowohl im Iso- als auch im CCIT-Code) dient ebenfalls der Apostroph.

Innerhalb des Textes sind alle Zeichen bis auf den Apostroph zulässig und werden abgespeichert.

Copyright duplication of this document and the use of ... of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

6. DER ASSEMBLER

6.1. Ablochvorschriften für Quellenprogramme im Assemblercode

6.1.1. Benutzbare Codes

Intern arbeitet der Assembler (wie auch die Makros zur Ein/Ausgabe-Steuerung; vgl. 2.3.1.) im 7-Bit-Iso-Code (vgl. DIN 66003; Juli 1968 und ISO/R646-1967) Es ist aber möglich, Quellenprogramme im CCIT-Fernschreibcode abzulochen. In diesem Fall übersetzt der Assembler mit Hilfe des Makros XEINCO die CCIT-Bandwerte in ISO-Bandwerte.
Einige Fernschreibcode-Sonderzeichen gibt es im Iso-Code nicht.
Sie werden in folgende Iso-Zeichen übersetzt:

× in EOT (Ende der Übertragung)

1o in ! (Ausrufungszeichen)

✱ in ETB (Ende des Datenübertragungsblocks)

Die Fernschreibzeichen Zi und Bu werden nicht in Iso-Bandwerte übersetzt; sondern sie steuern nur den Code-Übersetzungsvorgang.

6.1.2. Symbolische Adressierung

6.1.2.1. Marken

Bei der Programmierung im Assembler-Code werden Kernspeicherzellen in der Regel symbolisch adressiert. Symbolische Adressen beziehen sich dabei auf Marken, denen der Assembler die zugehörigen Kernspeicheradressen zuordnet (Aufbau eines Adress-Buches)

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Marken werden durch einen Namen gekennzeichnet, der in runde Klammern einzuschließen ist. Als Name ist eine beliebige Folge von alphanumerischen Zeichen zugelassen; die Folge muß jedoch aus mindestens einem Zeichen bestehen, außerdem haben nur die ersten fünf alphanumerischen Zeichen des Markennamens für den Assembler unterscheidende Bedeutung. Zwischenräume (SP) innerhalb der runden Klammern werden überlesen. Zum Beispiel haben die Marken (FREIBURG), (␣F␣REIBURG) und (FREIBIER) für den Assembler gleiche Bedeutung. Marken werden also in der Form (ABCD) vor dem zu adressierenden Befehls- bzw. Datenwort mit einem oder mehreren Trennzeichen dazwischen abgelocht. Als Trennzeichen dienen die Iso-Code-Zeichen (bzw. CCIT-Code-Zeichen) LF (bzw. ZL), SP (bzw. ZW) und CR (bzw. WR).

In 1.2. und 1.5. wurde erwähnt, daß Doppelworte im Kernspeicher eine durch vier teilbare Adresse tragen müssen, damit sie durch Doppelwort-Befehle angesprochen werden können. Aus diesem Grunde können Marken mit einer zusätzlichen Klammer (als Steuersymbol für den Assembler) auch in der Form ((ABCD) definiert werden. Der Assembler ordnet einer so deklarierten Marke eine durch 4 teilbare Adresse zu. Das geschieht in der Form, daß, falls erforderlich, der Befehl CA0,0 zusätzlich weggespeichert wird und dann erst die durch ((ABCD) markierte Zelle, deren Adresse nun durch vier teilbar und somit zur Ablage eines Doppelwortes geeignet ist, folgt. Um Speicherplatz zu sparen, ist es zweckmäßig, Doppelwörter en bloc abzulegen, denn dann wird höchstens ein zusätzliches Füllwort benötigt. -

In jedem Benutzerprogramm muß genau einmal die Marke (START) definiert werden, (START) muß dabei die Startadresse des Programms sein.

(AFEHL) markiert die Rückkehradresse aus arithmetischen Makros für den Fall, daß diese (z.B. wegen unzulässigen Parametern)

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

nicht ablauffähig sind. Die zu (APEHL) gehörige Adresse wird in die Zelle 96/97 des Benutzerprogramms gebracht. Wird die Marke (APEHL) nicht deklariert, d.h. wenn der Benutzer auf eine eigene Fehlerroutine für arithmetische Makros verzichtet, so wird an diese Stelle eine 1 abgelegt. (Näheres dazu vgl. 2.2.1.).

Sinngemäß das gleiche gilt für die Marke (PFEHL) bei Fehlern beim Ablauf von Makrobefehlen, die periphere Geräte benötigen (vgl. 2.3.1.4.). Die zu (PFEHL) gehörige Adresse bzw. die Zahl 1 wird in die Zelle 98/99 des Benutzerprogramms gebracht.

6.1.2.2. Symbolische Adressen

Symbolische Adressen rufen Marken auf und werden wie diese codiert. Sie können innerhalb von Adress-Langbefehlen, als Parameterangaben bei W-, X-, Y- oder Z-Befehlen oder als eigenständige Informationen auftreten.

Der Assembler legt für symbolische Adressen die zugehörigen numerischen Adressen ab. (Diese gewinnt er aus dem Adressbuch).

Wenn die zugehörige Marke in der Codierung vor dem Aufruf einer symbolischen Adresse bereits deklariert wurde, darf diese symbolische Adresse mit einem numerischen Zuschlag versehen werden; d.h. Worte, die dem durch (ABCD) oder ((ABCD) markierten Wort folgen (bzw. vorangehen), können mit 2 (ABCD), 4 (ABCD), ... (bzw. -2 (ABCD), -4 (ABCD), ...) adressiert werden. Durch Adressierung mit Zuschlag dürfen keine Zellen, die in der Form ((ABCD) markiert wurden, überzählt werden.

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung Mitteilung ihres Inhalts nicht gestattet soweit nicht ausdrücklich zugestanden Zuwiderhandlungen verpflichten zu Schadenersatz Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

6.1.3. Internbefehle

Maschineninterne Befehle werden in folgender Weise abgelocht:
Nach der Buchstabenverschlüsselung des Operationsteils (vgl. 1.3.) folgen die Adressenangaben; diese werden durch Komma voneinander getrennt. Innerhalb der Codierung eines maschineninternen Befehls dürfen keine Trennzeichen abgelocht werden, da der Assembler mit einem Trennzeichen die Eingabe eines Befehls als beendet ansieht und eine neue Information erwartet. Registeradressen werden numerisch (0-15) abgelocht.

Kernspeicher-Adressangaben c, die von der Lage des Programms im KSP abhängig sind, müssen symbolisch angegeben werden.

Beispiele: B7,0,-8(ABCD)
F15,0,(EFG)

Ist die Angabe c nicht von der Lage des Programms im KSP abhängig, so wird c numerisch angegeben:

Beispiele:

EU15,15,62	(Relativer Sprung)
LCB1,0,65520	(Bringen der
LCB1,0,-16	Konstante -16)
B4,0,78	(Bringen des Inhaltes einer Zelle des Betriebssystems)

6.1.4. Symbolische Adressen als Zahlen

Symbolische Adressen können vom Assembler auch als Einfachwortzahlen eingelesen werden. Dazu muß die symbolische Adresse mit einem Zuschlag versehen abgelocht werden (nötigenfalls muß also der Zuschlag 0 angegeben werden). Der Zuschlag ist notwendig, da der Assembler die symbolische Adresse sonst als Marke auf-

fassen würde. Durch den Assembler wird die zugehörige Adresse (unter Berücksichtigung des Zuschlags) numerisch abgelegt.

Beispiel: Ab KSP-Zelle (T) stehe eine Sprungtabelle

Codierung der Sprungtabelle:	Rel.Adr. bzgl. (T):
(T) 0(A)	0
0(B)	2
4(C)	4
-2(D)	6
0(D)	8
.	.
.	.
.	.

Mit Hilfe einer bezüglich (T) relativen Adresse, die in Register 1 steht, soll ein Sprung nach Zelle <<1>+(T)> ausgeführt werden. (So soll zum Beispiel für <1>=6 nach -2(D) gesprungen werden). Codierung des Sprungbefehls: B15,1,(T)

6.1.5. Datenfelder

Durch das Steuersymbol /m können <u>am Anfang</u> und <u>am Ende</u> eines Benutzerprogramms Datenfelder reserviert werden. In 6.1.8. wird beschrieben, wie die Symbole /m von der eigentlichen Codierung abzusetzen sind. - Diese Datenfelder können durch Marken (Feldname) definiert werden. Sollen für ein Feld m Bytes reserviert werden, so codiert man nach der Marke durch Zwischenraum getrennt /m. Dabei muß m gradzahlig sein.

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Beispiel: Durch

(NAME) /500

werden 250 Worte freigehalten. Der erste dieser freigehaltenen Plätze kann durch (NAME) symbolisch adressiert werden. /m ist auch ohne vorhergehende Marke zulässig.

Es sind beliebig viele Steuersymbole /m erlaubt; jedoch müssen diese in der Codierung an den in 6.1.8. beschriebenen Stellen stehen. Tritt innerhalb der eigentlichen Codierung /m auf, so gibt der Assembler eine Fehlermeldung aus.

Beim Assemblieren benötigen freizulassende Datenfelder keinen zusätzlichen Speicherplatz, da nur die Gesamtlänge der freizuhaltenden Datenfelder notiert wird. (Die bei der Definition, der Datenfelder codierten Feldnamen benötigen natürlich Platz im Adressbuch).

Auch durch das im Assembler enthaltene Stanzprogramm (vgl. 6.2.) werden die hier beschriebenen Datenfelder nicht voll ausgestanst; sondern innerhalb der Kopfdaten des Streifens wird wieder nur die Gesamtlänge der freizuhaltenden Datenfelder notiert. Durch dieses Verfahren wird die Länge von binären Programmstreifen reduziert.

Da der Lader immer an das KSP-Ende gelegt wird, kann beim Laden durch Datenfelder, die ans Ende eines Programms gelegt werden, KSP-Platz gespart werden; da diese Datenfelder den Speicherplatz des Laders einnehmen können.

6.1.6. Druckmasken

Druckmasken werden für den Makro XPUVOR,e,i,m (vgl. 2.3.2.3.) benötigt, um im Puffer das gewünschte Druckbild der Zeile aufzubauen.

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Ablochvorschrift: d;v;n;t;b;z

d = Druckposition (Nr. des Pufferbytes) $1 \leq d \leq 255$

v = Maximale Stellenanzahl vor dem Trennzeichen $0 \leq v \leq 15$

n = Stellenanzahl nach dem Trennzeichen $0 \leq n \leq 15$

t = Iso-Bandwert des gewünschten Trennzeichens $1 \leq t \leq 127$

bzw. t = Anzahl von Textzeichen (s. Z=9) $1 \leq t \leq 255$

$b = b_1 + b_2 + b_3 + b_4$

wobei:

$b_1 = \begin{cases} 0 & \text{Ausgabe ohne führende Nullen} \\ 1 & \text{Ausgabe mit führenden Nullen} \end{cases}$

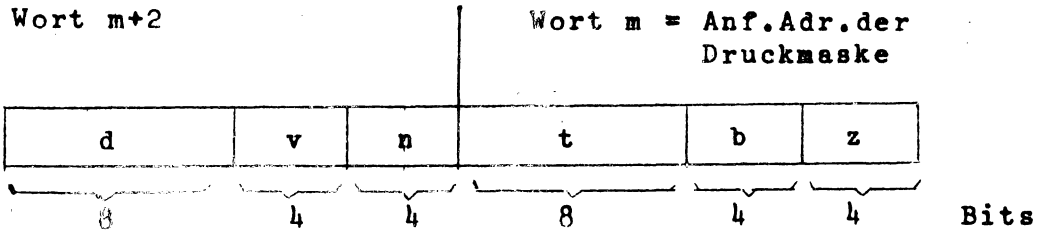
$b_2 = \begin{cases} 0 & \text{Ist die Zahl gleich 0, so wird diese entsprechend der Druckmaske abgelegt.} \\ 2 & \text{Ist die Zahl gleich 0, so wird das Druckbild durch Zwischenräume ersetzt (Spaltensprung).} \end{cases}$

$b_3 = \begin{cases} 0 & \text{Als Vorzeichen wird "-" bzw. "␣" abgelegt} \\ 4 & \text{Als Vorzeichen wird "-" bzw. "+" abgelegt} \end{cases}$

$b_4 = \begin{cases} 0 & \text{Das Vorzeichen wird vor der Zahl abgelegt} \\ 8 & \text{Das Vorzeichen wird hinter der Zahl abgelegt} \end{cases}$

z = Typ der aufzubauenden Zahl (vgl. Tab.) $1 \leq z \leq 15$

Die abgelochte Zahlenfolge d;v;n;t;b;z wird vom Assembler folgendermaßen abgelegt:



Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung ur Mitteilung ihres Inhalts nicht gestattet soweit nicht ausdrücklich zugestanden Zuwiderhandlungen verpflichten zu Schadenersatz Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

z	Typ der zur Ausgabe vorzubereitenden Zahl 1≤z≤15	
0	binär 1W	Ein Wort (1W) wird byteweise abgelegt. Beispiel: LOLOOLOO OOOLLOOL (Buchstaben)
1	FPZ 1W	Festpunktzahlen können entsprechend der Vorgaben von v und n und t nach Belieben in zwei Teile getrennt werden, wobei n vorrangig ist. Beispiel: Die gespeicherte Zahl -32006 wird durch d;5;2;47;1;1 in der Form -00320/06 abgelegt
2	FPZ 2W	
3	FPZ 3W	
4	FPZ 4W	
5	GPZ 2W ohne Exp.	Zahl: 4.78 Druckmaske: d;3;4;44;13;5 Ablage: 004,7800+
6	GPZ 2W mit Exp.	Zahl: 4.78 Druckmaske: d;v;4;t;b;6 Für v, t und b kann jeweils eine 0 stehen, da die Ablage stets normalisiert ist mit Dezimalpunkt mit v=0 und mit b=0. Ablage: .4780E+01 Exponent stets zweistellig
7	GPZ 3W ohne Exp.	Siehe Bsp. z=5
8	GPZ 3W mit Exp.	Siehe z=6; bei der Ablage steht D statt E
9	Text	Für v,n und b kann jeweils eine 0 stehen, da diese Werte für die Ablage von Text ohne Bedeutung sind. Wenn der gespeicherte Text mit dem Endekennzeichen ' (Datenwert=39) abschließt, so wird dieser (ohne ') in den Puffer abgelegt, jedoch maximal t Textzeichen. Ist t>1-d+1, so werden maximal 1-d+1 Zeichen abgelegt.

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung Mitteilung ihres Inhalts nicht gestattet soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

6.1.7. Kommentar

Durch Trennzeichen von der Codierung getrennt, darf beliebiger Kommentar mit abgelocht werden. Dieser ist durch [zu eröffnen und durch] zu beenden. Innerhalb des Kommentars darf] nicht vorkommen. Bei der Speicherung des Programms durch den Assembler werden alle Zeichen zwischen [und] überlesen. Ein Kommentar darf an beliebiger Stelle des Quellenprogramms abgelocht werden; also z.B. auch im oder vor dem Kopf (vgl. 6.1.8.) des Quellenprogramms.

6.1.8. Aufbau eines Quellenprogramms

Ein Quellenprogrammstreifen hat folgenden Aufbau:

(Name) — Programmname.
Es gelten die gleichen Ablochvorschriften wie bei Marken (vgl.6.1.2.1.); jedoch muß das erste Zeichen des Namens ein Buchstabe sein.

n_A — n_A ist eine Einfachwortzahl, die die Programmart kennzeichnet:

- $n_A = 1$ Hauptprogramm
- $n_A = -1$ Hauptprogramm; zusätzlich soll der Rest des KSP als Datenbereich reserviert werden
- $n_A = 2$ Unterprogramm

n_S — Anzahl der Startparameter eines Hauptprogramms bzw. Anzahl der Parameter eines Unterprogramms.

Diese drei Kopfinformationen müssen auf jedem Quellenprogrammstreifen in der angegebenen Reihenfolge vorhanden sein. Für Kommentar gilt das in 6.1.7. gesagte.

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

(MARKE)	$/n_1$	Anfangs-Datenfelder (vgl. 6.1.5.)
•		
•		
•		
(MARKE)	$/n_i$	
=		Endekennzeichen der Kopfdaten
•		
•		
•		
•		
Programm		Eigentliche Codierung
•		Folgende Informations-Arten sind erlaubt:
•		Internbefehle (vgl. 6.1.3.)
•		W-Befehle (vgl. 2.5.)
•		X-Befehle (vgl. 2.1.)
•		Y-Befehle (vgl. 3.)
•		Z-Befehle (vgl. 2.4.)
•		Zahlen (vgl. 4.)
•		Text (vgl. 5.)
•		Marken (vgl. 6.1.2.)
•		Druckmasken (vgl. 6.1.6.)
•		Kommentar (vgl. 6.1.7.)
•		Nicht erlaubt ist /n (vgl. 6.1.5.).
=		Ende der eigentlichen Codierung
(MARKE)	$/n_1$	Schluß-Datenfelder (vgl. 6.1.5.)
•		
•		
•		
(MARKE)	$/n_i$	
x		Programmende-Kennezeichen
		x bei Codierung im CCIT-Code
		EOT bei Codierung im ISO-Code

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung u. Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten.

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Die einzelnen Informationen werden durch Trennzeichen voneinander getrennt abgelocht. Als Trennzeichen dienen die ISO-Code-Zeichen (bzw. CCIT-Code-Zeichen) LF (bzw. ZL), SP (bzw. ZW) und CR (bzw. WR). Leerstreifen (NUL) und voll gelochte Streifen (DEL) werden überlesen.

6.2. Funktionsweise des Assemblers

6.2.1. Generieren des Assemblers

Der Assembler-Generator ist ein Dienstprogramm, welches durch den Lader (vgl. 7) eingelesen werden kann. Er ist unter jedem Grund-Betriebssystem (GBS) ablauffähig; denn er benötigt nur Makros, die in jedem GBS enthalten sind (vgl. 2.3.5.).

Der Ass.-Generator hat die Aufgabe, einen Assembler zu generieren, der ebenfalls unter Steuerung jedes Grundbetriebssystems ablauffähig ist. Die Ausgabe des Ass.-Generators besteht aus einem 8-Kanal-Lochstreifen, der durch den Lader eingelesen werden kann.

Beim Start des Ass.-Generators kann man vorgeben, ob der zu generierende Assembler folgende Funktionen besitzen soll oder nicht:

Adress-Buch (ADB) ausgeben,
Bruchzahlen lesen,
Gleitpunktzahlen lesen,
3-fach-Wort-FPZ lesen,
4-fach-Wort-FPZ lesen,
Hexadezimalzahlen lesen,
Binärzahlen lesen,
Druckmasken lesen,
Z-Befehle lesen,
Weite (Benutzer-)Informationsarten lesen.

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung u. Mitteilung ihres Inhalts nicht gestattet soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

Jeder generierte Assembler besitzt grundsätzlich die folgenden Funktionen:

Grundbefehle lesen,
W-, X-, Y-Makro-Aufrufe entschlüsseln,
FPZ (Einfach- und Doppelwort lesen),
Text lesen,
Marken lesen und ADB aufbauen,
Datenfeld freihalten,
Kommentar überlesen.

Weiter können folgende Funktionen von jedem Assembler durchgeführt werden:

Fehlermeldungen bearbeiten,
Programm nur auf formale Fehler prüfen (vgl. 6.2.2.),
Programm stanzen (in ladefähiger Form).

Außer den obengenannten Blöcken besteht die Eingabe des Assembler-Generator noch aus

der X-MNL (X-Makro-Namenliste) und
der Y-MNL (Y-Makro-Namenliste).

6.2.2. Ablauf des Assemblers

Der Assembler besitzt zwei Startparameter S und G, durch die die gewünschte Ablaufmöglichkeit angegeben werden kann.

S=0 Nur PRÜFEN

Das über den Assembler einzulesene Quellenprogramm wird nicht abgespeichert (und kann demzufolge auch nicht gestanzt werden). Es wird nur auf formale Programmierungsfehler geprüft; diese werden auf dem Bedienungs-Fernschreiber protokolliert. Fehlermeldungen werden in Form von Anweisungen ausgegeben (vgl. 3.4.5.). Der Übersetzungsvorgang wird

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

erst fortgesetzt, wenn der Operator das entsprechende Quittungssignal (vgl. 8.3.1.) ausgegeben hat. (Der Operator hat z.B. die Möglichkeit, auf dem Lochstreifen eine Markierung, die die Korrektur des Quellenprogramms erleichtert, anzubringen).
Das Adress-Buch wird zwar auch bei dieser Ablauf-Version aufgebaut, aber es wird nicht ausgegeben.

S=1 Ablegen und Stanzen

Das Programm wird übersetzt und am Anfang des noch freien Kernspeicher-Bereiches abgelegt. Dabei wird das Adress-Buch (ADB) vom Kernspeicher-Ende an dem abgelegten Programm entgegenlaufend aufgebaut. Erkennt der Assembler einen formalen Fehler im Quellenprogramm oder ist der noch freie KSP-Bereich zu klein, so wird wie bei S=0 eine Meldung ausgegeben. Vom ersten Fehler an erfolgt der restliche Assemblerlauf wie bei S=0, es wird also nicht mehr gespeichert, das ADB wird nicht ausgegeben und das Programm wird nicht gestanzt.

Durch den Assembler im KSP abgelegte Programme sind nicht ablauffähig; denn sie sind so abgelegt, daß sie in einer Form gestanzt werden können, die ein vom Betriebssystem unabhängiges Laden und Binden in beliebige KSP-Bereiche erlaubt.

Wenn das Programm abgelegt ist, kann auf Wunsch das ADB ausgegeben werden (vgl. Startparameter G). Anschließend wird der zum Stanzen notwendige Kopf aufgebaut und das Programm wird auf 8-Kanal-Lochstreifen in binärer Form ausgestanzt.

G=0 Es erfolgt keine ADB-Ausgabe.

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

G=g Das ADB wird auf dem Gerät (Fernschreiber oder Drucker) mit der Geräte-Nummer g ausgegeben (Voraussetzung ist natürlich, daß der betreffende Assembler den Programmteil "ADB ausgeben" enthält).

6.2.3. Aufbau eines binären Programm-Lochstreifens

Der Datenträger eines vom Assembler ausgestanzten binären Programms ist grundsätzlich ein 8-Kanal-Lochstreifen.

Der Programm-Lochstreifen besteht aus vier Teilen:

Anfangs-Leerstreifen
6.2.3.1. Programm-Kopf
6.2.3.2. Programm-Rumpf
Ende-Leerstreifen

6.2.3.1. Programm-Kopf

Der Programm-Kopf enthält folgende Daten in binärer Form:

a) Programm-Name
b) Programm-Art
   Hier wird notiert, ob es sich um ein Hauptprogramm (mit oder ohne Reservierung des KSP-Restes) oder um ein Unterprogramm handelt (vgl. 6.1.8.).
c) Anzahl der Startparameter bei Hauptprogrammen bzw. Anzahl der Parameter bei Unterprogrammen
d) Effektive Länge des Programms
e) Relative Startadresse (bei Hauptprogrammen)
f) Zu AFEHL gehörige Relativadresse (bzw. 1) bei Hauptprogrammen (vgl. 2.2.1. und 6.1.2.1.)
g) Zu PFEHL gehörige Relativadresse (bzw. 1) bei Hauptprogrammen (vgl. 2.3.1.4. und 6.1.2.1.)
h) Anzahl der am Kopf eines Programms freizuhaltenden Datenfelder. (Bei Hauptprogrammen einschließlich

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung ihres Inhalts nicht gestattet soweit nicht ausdrücklich zugestanden Zuwiderhandlungen verpflichten zu Schadenersatz Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten

der 100 Bytes, die als Hilfsbereiche für das Betriebssystem reserviert werden).

i) Angaben über am Ende des Programms freizuhaltende Datenfelder
(Bemerkung: Freizulassende Datenfelder (vgl. 6.1.5.) benötigen beim Assemblieren keinen Platz im KSP. Auch die binären Programm-Lochstreifen sind kürzer.).

j) Transfer-Vektor
Im Transfer-Vektor stehen Angaben über die im Programm verwendeten W-Befehle (vgl. 2.5.).
Jedes im Quellenprogramm durch einen W-Aufruf angeforderte Unterprogramm wird folgendermaßen im Transfer-Vektor notiert:
1. Name des durch den W-Aufruf angesprochenen externen Unterprogramms.
2. Relative Adresse des letzten Aufrufes für diesen W-Befehl. (Die W-Aufrufe gleichen Namens sind durch eine Adress-Verkettung im Objektprogramm wieder auffindbar.)
3. Parameter-Anzahl. (Diese wird beim Assemblieren festgestellt).

Vor den Eintragungen zu den verschiedenen W-Befehlen ist die Länge dieses Transfer-Vektors notiert.

k) Makro-Liste
In dieser Liste stehen Angaben über im Programm verwendete X-Befehle (vgl. 2.1.).
Jeder im Quellenprogramm vorkommende X-Makro-Aufruf wird folgendermaßen notiert:
1. PZW-Nummer des Makro-Aufrufes
2. Relative Adresse des letzten Aufrufes für diesen X-Befehl. (Die X-Makro-Aufrufe mit gleichen PZW-Nummern sind ebenfalls durch eine Adress-Verkettung im Objektprogramm wieder auffindbar).

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Vor den Eintragungen zu den verschiedenen X-Makro-Aufrufen ist die Länge der Makro-Liste notiert.

1) Am Ende des Kopfes befindet sich zu Kontrollzwecken beim Laden eine Summe über alle Kopfdaten.

6.2.3.2. Programm-Rumpf

Unmittelbar nach dem Kopf folgt das eigentliche Programm; d.h. die Übersetzung der im Quellenprogramm zwischen den beiden Gleichheitszeichen stehenden Informationen (vgl. 6.1.8.). Das übersetzte Programm wurde vom Assembler blockweise abgelegt und wird auch blockweise ausgestanzt. Ein Block besteht aus 86 Worten (= 172 Stanzungen auf einem 8-Kanal-Lochstreifen).
Der letzte Block eines Programms kann jedoch auch kürzer sein.

Aufbau eines Blockes:

a) Programm-Informationen (80 Worte).

b) Relativierungsvektor (5 Worte = 80 Bit).
Der Relativierungsvektor ist eine Bitleiste, wobei jedem der 80 Programm-Informationsworte ein Bit zugeordnet ist. Alle symbolischen Adressen werden vom Assembler in relative Adressen (relativ zum effektiven Programmanfang) übersetzt und auch so abgelegt. Im Relativierungsvektor sind alle relativen Adressen markiert. - Mit Hilfe des Relativierungsvektors muß das Programm beim Laden (damit es ablauffähig ist) absolut adressiert werden.

c) Längssumme (1 Wort) über die 80 Informations- und 5 Relativierungsvektor-Worte des Blockes (Kontrollsumme beim Laden).

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung, Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

Wenn man davon absieht, daß aus organisatorischen Gründen (Einsparen von Speicherplatz beim Assemblieren) der Assembler im KSP erst den Programm-Rumpf und dann den Programm-Kopf ablegt, entspricht das Bild des im KSP abgelegten Programms vollbinär dem Bild auf dem 8-Kanal-Lochstreifen.
(1 Byte im KSP ≙ 1 Lochung auf dem Streifen).
Die Stellenzuordnung wurde in 2.3.1.1. erläutert.

Dieses Verfahren bringt u.a. folgende Vorteile mit sich:

1. Das in jedem Assembler vorhandene Stanzprogramm hat einen sehr einfachen Aufbau (vollbinäres Stanzen).
2. Es besteht eine Kompatibilität zu anderen Datenträgern. Die vollbinäre Darstellung erlaubt zum Beispiel ohne weiteres eine Übertragung auf einen Platten-Speicher.

7. DER LADER (BINDER)

Der Lader hat die Aufgabe, nach den in 6.2.3. angegebenen Normen gestanzte Programme in ablauffähiger Form in den KSP einzulesen. Dabei können mehrere Programmstreifen (ein Hauptprogramm und mehrere Unterprogramme) ein ablauffähiges Programm im KSP ergeben. -
Der Lader selbst ist nicht ständiger Bestandteil des Betriebssystems. Er wird nur bei Bedarf durch ein spezielles, ständig im Betriebssystem enthaltenes, Leseprogramm an das Ende des KSP eingelesen. -

Der Lader wird mit einem Startparameter, der die gewünschte Programm-Priorität angibt, gestartet.

Weitergabe sowie Vervielfältigung dieser Unterlage Verwertung Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM Eintragung vorbehalten

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

U.a. hat der Lader (mit Binder) folgende Aufgaben:

Einziehen der zum Programm gehörigen Streifen (Hauptprogramm und eventuell benötigte Unterprogramme und einzubindende Makro-Unterprogramme).

Ersetzen der relativen Adressen durch absolute Adressen mit Hilfe der Relativierungsvektoren.

Interpretation der Kopfdaten.
Dazu gehört insbesondere:
Datenfelder freihalten
Einsetzen der effektiven Adressen bei W-Aufrufen (Adress-Verkettung).
Einsetzen von PZWn (Adress-Verkettung) bei Makros, die im GBS der Maschine enthalten sind.
Ersetzen der Makro-Aufrufe durch F-Unterprogrammsprünge und Einbinden der entsprechenden Unterprogramme, falls die Makro-Unterprogramme nicht im GBS der Maschine enthalten sind.

8. BEDIENUNGSPROGRAMM

8.1. Programmsteuerung

Beliebig viele Programme können in der Maschine unter Steuerung des Grundbetriebssystems im Zeitmultiplexbetrieb laufen. Die Steuerung der verschiedenen Programmabläufe wird entsprechend vorgegebener Prioritäten vom Betriebssystem vorgenommen.
Wie schon erwähnt, werden die im Kernspeicher stehenden Programme in der Programmwarteschlange (PWS) registriert. Die Reihenfolge der Notierungen in der PWS entspricht

Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

dabei den Prioritätsstufen. Neben der Anfangsadresse, der Startadresse und der Programmlänge enthält die PWS noch eine Bitliste, in der Notierungen gemacht werden, ob und wie das Programm gestartet wurde, ob und weswegen das Programm unterbrochen wurde und ob der Programmablauf beendet wurde; außerdem steht in dieser Zelle die Anzahl der Startparameter des Programms. Mit Hilfe des Bedienungsprogramms hat der Operator Zugriff zur Programmsteuerung.

8.2. Bedienungsprogramm

Das Bedienungsprogramm besteht aus mehreren, voneinander unabhängigen Teilprogrammen, die über den Bedienungsfernschreiber (BFS) gestartet werden können. Um das Bedienungsprogramm zu starten, ist zunächst die Anruf-Taste am BFS zu drücken; das Betriebssystem aktiviert dann die BFS-Eingabe; d.h. am BFS leuchtet die rote Eingabe-Signallampe. Dann ist das Zeichen ✠ anzuschlagen. Das jeweils gewünschte Teilprogramm wird danach durch eine Ziffer gekennzeichnet. Wenn ein Parameter (positive Einfachwortzahlen) anzugeben ist, folgt dieser durch Komma getrennt; als Endekennzeichen folgt ein Semikolon.

Jedes angeschlagene Zeichen wird sofort automatisch abgedruckt, so daß das Protokoll also nicht als Quittung für die Annahme des Zeichens durch die Zentraleinheit aufgefaßt werden kann. Zur Kennzeichnung der Beendigung eines Teilprogramms wird ein weiteres Semikolon gedruckt. Konnte das Teilprogramm nicht beendet werden (Fehler), so wird -; ausgedruckt.

Die Bedienungsaufrufe sind kurz formuliert; sie enthalten außer wenigen Sonderzeichen auch nur Ziffern, so daß für die Aufrufe kein umfangreiches Leseprogramm ständtiger Bestandteil des GBS sein muß.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.

Copyright reproduction of this document and the use or communication of its contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model

8.3. Aufrufe des Bedienungsprogramms

Die folgenden vier Teilprogramme stellen den Mindestumfang des Bedienungsprogramms dar:

8.3.1. Quittungssignal

Aufruf: ✠0,n,m;

Das durch YANW,n (vgl. 3.4.5.) unterbrochene Programm mit der Priorität n wird fortgesetzt. m wird in Zelle 64/65 des Programms abgelegt!

8.3.2. Programm starten

Aufruf: ✠1,n;

In der PWS wird ein Startvermerk bei dem Programm mit der Priorität n eingetragen, die Startparameter werden eingelesen und im Parameterfeld (Plätze 64/65,... des Benutzerprogramms) von Programm n abgelegt; außerdem wird Platz 15 im Registerfeld 2 belegt (vgl. 10.1.).

8.3.3. Benutzerprogramm beenden

Aufruf: ✠2,n;

Das Programm mit der Priorität n wird beendet. In der Programmwarteschlange wird ein Stopvermerk eingetragen und die vom Programm der Priorität n eingetragenen Gerätesperren (vgl. 3.4.6.) werden aufgehoben.

8.3.4. Programm löschen

Aufruf: ✠3,n;

Alle Eintragungen, die das Programm mit der Priorität n betreffen, werden gelöscht. (Dazu gehören auch die Sperr-Eintragungen, die ein anderes Programm für dieses Programm vorgenommen hat). Das Programm kann dann nicht mehr gestartet werden. Handelte es sich um das hinterste Programm im KSP, so wird der von diesem Programm belegte und beanspruchte Platz freigegeben.

Zusätzlich können über den Grundbetriebssystem-Generator noch die folgenden drei Teilprogramme in das Bedienungsprogramm aufgenommen werden:

8.3.5. PWS ausdrucken Aufruf: ✠4;
Der Inhalt der PWS (Programm-Warteschlange) wird über den Bedienungsfernschreiber ausgegeben.

8.3.6. Programm zyklisch starten Aufruf: ✠5,n;
Wie bei Aufruf ✠2,n; jedoch erfolgt zusätzlich noch eine Eintragung über zyklischen Start.

8.3.7. Programm zyklisch beenden Aufruf: ✠6,n;
Die Eintragung über zyklischen Start wird in der PWS gelöscht.

9. TESTPROGRAMME

9.1. Überwacherprogramm

9.1.1. Aufgaben des Überwacherprogramms
Benutzerprogramme, die im Kernspeicher stehen und in der Programmwarteschlange (PWS) notiert sind, können über das Überwacherprogramm gestartet werden. Das geschieht in der Form, daß der Überwacher mit einem Startparameter, der die Priorität des zu überwachenden Programms angibt, gestartet wird. Innerhalb des Überwacherprogramms wird dann das zu

testende Benutzerprogramm Befehl für Befehl simuliert. Dabei wird im wesentlichen geprüft, ob die Kernspeicher-Adressen von Sprung- und Umspeicherbefehlen innerhalb des Bereiches d.zu überwachenden Programm liegen. Ist das nicht der Fall, so wird vor Ausführung dieses unzulässigen Befehls eine Fehlermeldung ausgegeben und der Überwacherlauf beendet.

Über den Überwacher kann auf zwei verschiedene Arten getestet werden:

1. Automatische Testläufe
   Man läßt das zu testende Programm mit einem geeigneten Protokollprogramm-Parametersatz über den Überwacher von Anfang bis Ende durchlaufen und prüft hinterher das Ausgabeprotokoll.

2. Manuelle Testläufe
   Man schiebt Zwischenstops ein oder beendet den Überwacher über das Bedienungsprogramm und ändert nach Bedarf das Programm oder läßt gewisse Programmteile mit anderen Parametern wiederholt ablaufen. Der Vorzug sollte allerdings automatischen Testläufen gegeben werden (Einsparung von Maschinenzeit). Zu manuellen Tests solllte nur in solchen Fällen zurückgegriffen werden, wo automatische Läufe nicht zum Erfolg führen.

9.1.2. Funktionsweise des Überwachers

Der Überwacher wird wie ein normales Benutzerprogramm bei Bedarf in den Kernspeicher geladen und gestartet. Der Überwacher enthält eine Programmschleife; ein Durchlauf dieser Schleife entspricht der Ausführung eines Befehls (maschinenintern oder Makroaufruf) des zu überwachenden Programms. Der Überwacher wird mit drei Startparametern gestartet:

STAPA1 = Priorität des über den Überwacher zu startenden Programms

STAPA2 = Kernspeicheradresse, an der das zu überwachende Programm gestartet werden soll.

Wird 0 angegeben, so wird das Programm an der Stelle gestartet, die in der Assembler-Codierung mit der Marke (START) versehen war. Diese Adresse holt sich der Überwacher aus der PWS. Wird 1 angegeben, so wird der Testlauf dort fortgesetzt, wo er zuletzt unterbrochen wurde.

STAPA3 = Angaben für das Protokollprogramm (vgl. 9.2.)

9.2. Protokollprogramm zum Überwacher

9.2.1. Aufgaben des Protokollprogramms

Durch das Protokollprogramm können nach der Ausführung eines Befehls für den Benutzer für das Testen wichtige Routinen ausgeführt werden. Die für das Protokollprogramm notwendigen Parameter werden auf einem Datenstreifen zusammengefaßt.

Es sind folgende Routinen vorgesehen:

1. Kernspeicherausgabe
   Ein KSP-Bereich kann auf verschiedene Arten ausgegeben werden:

Einfachwort:	Festpunktzahl
	binär
Doppelwort:	Festpunktzahl
	Gleitpunktzahl
Dreifachwort:	Festpunktzahl
	Gleitpunktzahl

2. Registerausgabe
   Hier ist die gleiche Aufgliederung wie in 1. vorgesehen.

3. Zähler
   Es wird gezählt wie oft ein bestimmter Befehl durchlaufen wurde. Außerdem können an beliebiger Stelle des Programms die Zähler ausgegeben werden.

4. Zwischenstop
   Der Überwacherlauf wird beendet.

9.2.2. Funktionsweise des Protokollprogramms

Zum Protokollprogramm gehören:

1. Leseprogramm für den Datenstreifen des Protokollprogramms

2. Eigentliches Protokollprogramm.

Durch den Startparameter STAPA3 des Überwachers werden dem Programm folgende Informationen gegeben:

STAPA3 = 0 Es ist kein (bzw. kein neuer) Datenstreifen einzulesen.

STAPA3 = 1 Es ist ein Datenstreifen für das Protokollprogramm einzulesen; die Angaben des vorher eingegebenen Datenstreifens stehen dann nicht mehr zur Verfügung.

Jede Information des Datenstreifens für das Protokollprogramm besteht aus zwei Teilen

1. Teil: Wann soll das Protokollprogramm angesprochen werden? Es wird die Adresse des Befehls angegeben, nach dessen Ausführung eine Routine des Protokollprogramms zu durchlaufen ist.

2. Teil: Welche Routinen hat das Protokollprogramm auszuführen?

10.1. ANHANG I

Das Grundbetriebssystem

Kurzfassung einer Beschreibung von Abläufen bei Programmunterbrechung

Das Grundbetriebssystem (GBS) kann an vier verschiedenen Stellen gestartet werden. Diesen Eingängen sind hardwareseitig bestimmte Ereignisse zugeordnet:

1. Wenn ein Benutzerprogramm auf einen nicht interpretierbaren oder privilegierten Befehl läuft oder in eine geschützte Speicherzelle schreiben will, wird hardwareseitig nach Adresse 2/3 gesprungen.
2. Wenn ein Benutzerprogramm auf einen programmierten PZWn läuft (dieser Befehl wurde für einen X- oder Y-Makro eingesetzt), wird hardwareseitig nach 6/7 gesprungen.
3. Wenn von der Nahtstelle die Meldung Interrupt kommt, wird hardwareseitig nach 10/11 gesprungen.
4. Wenn die Maschine eingeschaltet wird, erfolgt hardwareseitig ein Sprung nach 14/15.

Ein Sprung an eine der angegebenen Stellen impliziert immer, daß der Programmablauf im Programmzustand 2 fortgesetzt wird (Abschalten von Unterbrechbarkeit und Schreibsperre) und in den ersten drei Fällen auch eine Notierung der Rückkehradresse in den Bytes 0/1 des Kernspeichers.

In den Bytes 2-5, 6-9, 10-13 und 14-17 stehen Sprungbefehle, die auf die Programmroutinen für die vorliegenden Ursachen führen.

Alle Programme, die ablauffähig im KSP stehen, werden in der Programmwarteschlange (PWS) geführt. Die Reihenfolge der Notierungen entspricht dabei den Prioritätsstufen. In der PWS

sind alle Größen und laufenden Notierungen, die zur Programmsteuerung durch das Betriebssystem benötigt werden, zu den einzelnen Programmen eingetragen (vgl. 8.1.).

Wurde ein Programm durch eine der unter 1. bis 3. genannten Ursachen unterbrochen, so muß das GBS nach Ausführung der zugehörigen Routinen (vgl. unten) unter Berücksichtigung der vorgegebenen Prioritäten ein fortsetzbares Programm suchen. Im Programm-Ablaufregister (PAR) notiert das GBS, welches Programm jeweils läuft.

Am Kopf jedes Benutzerprogramms sind 100 Bytes als Hilfsbereiche für das Betriebssystem reserviert.

Registerfeld 1:	Relativzellen 0/1 bis 30/31. In diese Zellen wird der Registersatz bei programmiertem PZW gerettet.
Registerfeld 2:	Relativzellen 32/33 bis 62/63. In diese Zellen wird der Registersatz bei Interrupt gerettet.
Parameterfeld:	Relativzellen 64/65 bis 78/79. In diese Zellen legt das Betriebssystem die Parameter nach einem X- oder Y-Makroaufruf ab. Auch Startparameter eines Programms werden hier abgelegt.
Zwischenspeicherfeld:	Relativzellen 80/81 bis 94/95. Dieses Feld dient als Zwischenspeicher für X- oder Y-Makrobefehle. Teilweise werden auch Ergebnisse hier abgeliefert.
AFEHL:	Relativzelle 96/97
PFEHL:	Relativzelle 98/99

1. Alarmfall

Nach Start des Betriebssystems auf Byte 2/3 wird zunächst untersucht, ob der Einsprung durch den Befehl F15,0,0 oder durch den Peripherie-Sonderaufruf PERa,b ausgelöst wurde. PERa,b ist ein Spezialaufruf für einen schnellen Zugriff zu bestimmten Peripherie-Kanalprogrammen.
Ist dies nicht der Fall, so handelt es sich um einen echten der vorstehend aufgeführten Alarmfälle. Es werden dann eine Fehlermeldung und Kennzeichnungen ausgegeben, die eine Lokalisierung der Alarmursache erleichtern sollen. Das betroffene Benutzerprogramm wird als nicht fortsetzbar abgebrochen.
Wurde der Einsprung durch den Befehl F15,0,0 ausgelöst, so handelte es sich um den Rücksprung aus einem Makro-Unterprogramm. Benötigte dieser Makro kein peripheres Gerät, so wird nach Regenerierung des Registersatzes das Benutzerprogramm, das den Makro aufgerufen hatte, fortgesetzt. Im anderen Fall müssen zunächst spezifische Eintragungen in der Programmwarteschlange (PWS) gelöscht werden; danach ist noch zu prüfen, ob ein anderes Programm auf das gleiche periphere Gerät wartet. Ist dies der Fall, so wird für das wartende Programm der verlangte Peripherie-Makro gestartet und in der PWS ein entsprechender Vermerk eingetragen.

2. Programmierter PZW

Tritt im Ablauf eines Benutzerprogramms ein PZWn auf, so wird im Grundbetriebssystem das Teilprogramm gestartet, das letztlich den Ablauf des Makro-Unterprogramms mit der Nummer n zur Folge hat. Zunächst wird der Registersatz in das erste Registerfeld (R_1) des

laufenden Benutzerprogramms gerettet, danach mit Hilfe der Rückkehradresse aus Kernspeicherzelle 0/1 die Makrobefehlsnummer n festgestellt. Ist n größer als 127, so handelt es sich nicht um einen X-, sondern um einen Y-Makrobefehl. In beiden Fällen werden jeweils aus einer Makrobefehlsliste (MBL) die für diesen Makrobefehl benötigten Angaben abgeholt. Falls die MBL den Makrobefehl nicht enthält, erfolgt ein entsprechendes Fehlerprotokoll. Die für den Ablauf des Makrobefehles benötigten Parameter werden abgeholt und in dem Parameterfeld am Anfang des Benutzerprogramms abgelegt.

Wenn der Makrobefehl kein peripheres Gerät benötigt, kann er jetzt gestartet werden. Andernfalls wird untersucht, ob das benötigte periphere Gerät frei ist. Dann werden die Eintragungen für diesen Makro in der PWS vorgenommen. War das periphere Gerät bereits vorgemerkt, so wird ein fortsetzbares Programm gesucht. Andernfalls wird das Kennzeichen "Warten auf Interrupt" in der PWS gesetzt und der Makro gestartet.

Soll ein fortsetzbares Programm gesucht werden, so muß zunächst geprüft werden, ob ein Interrupt ansteht. In diesem Fall wird das Retten des Registersatzes verhindert. Es wird danach nach 10/11 gesprungen, wo der Start für die Bearbeitung von Interrupt-Meldungen liegt.

Liegt kein Interrupt vor, so wird die PWS nach einem fortsetzbaren Programm abgesucht und dieses fortgesetzt. Der Aufbau der PWS gewährleistet, daß das fortsetzbare Programm mit der höchstmöglichen Priorität fortgesetzt wird. Ist kein fortsetzbares Programm vorhanden, so wird in eine Programmwarteschleife gesprungen.

3. **Interrupt**

Erfolgt ein Einsprung bei 10/11 so wird, falls der Inhalt des Programmablaufregisters ungleich Null ist, der Registersatz in das Registerfeld R_2 des Benutzerprogramms gerettet.

Liegt ein Interrupt für den Bedienungsfernschreiber vor, so wird untersucht, ob es sich um einen Bedienungsaufruf handelt. In diesem Fall wird das Bedienungsprogramm gestartet bzw. fortgesetzt. In allen anderen Fällen wird in der PWS dasjenige Benutzerprogramm gesucht und fortgesetzt, das dieses periphere Gerät aufgerufen hatte.

4. **Maschine EIN**

Nach Start auf 14/15 wird ein fortsetzbares Programm gesucht, nachdem alle Programme, deren fehlerfreie Fortsetzung nicht gewährleistet ist, als nicht fortsetzbar notiert wurden. Dies betrifft insbesondere das Programm, das beim Ausschalten der Anlage (bzw. beim Zusammenbruch der Stromversorgung) im Programmablaufregister eingetragen war.

10.2. ANHANG

Einschränkungen bei der Programmierung

Für Quellenprogramme im Assembler-Code, die unter Steuerung des Grundbetriebssystems (GBS) assembliert werden und ablaufen sollen, muß der Programmierer stets gewisse Einschränkungen, die das vorliegende System vorschreibt, beachten. In der folgenden Zusammenstellung dieser Einschränkungen werden drei Klassen unterschieden:

A. Einschränkungen, die mit dem Aufbau des GBS zusammenhängen

B. Einschränkungen für den Assembler des GBS

C. Hardware-Einschränkungen.

A. Grundbetriebssysteme (GBS)

1. F15,0,0 darf in einem Benutzerprogramm nicht codiert werden. Dieser Befehl würde, in einem Benutzerprogramm codiert, zu einem Schreibsperrenalarm führen. Da aber der Befehl F15,0,0 im Betriebssystem eine Sonderfunktion (Rücksprung aus Makro-Unterprogrammen) besitzt, darf er auch nicht codiert werden, wenn man diesen Effekt (Schreibsperre) erreichen will.

2. Am Kopf jedes Benutzerprogramms sind 100 Bytes als Hilfsbereiche für das Betriebssystem reserviert (vgl. 10.1.)
Das Registerfeld 1, das Parameterfeld und das Zwischenspeicherfeld stehen dem Benutzer als Zwischenspeicher zur Verfügung; diese Felder werden nur bei Aufruf eines X- oder Y-Befehls (PZWn) überschrieben (das Parameterfeld und Zwischenspeicherfeld nur teilweise). - Das Registerfeld 2 ist nicht als Zwischenspeicher geeignet, da es bei jedem Interrupt (Meldung von der E/A-Schnittstelle) überschrieben wird. -

3. In der Programmwarteschlange (PWS) sind die Anzahl der Startparameter und die relative Startadresse eines Benutzerprogramms in einem Wort notiert. In den Bits 2^{15} bis 2^{13} dieses Wortes steht die Anzahl der Startparameter, in den Bits 2^{12} bis 2^{0} die relative Startadresse. Das hat folgende Einschränkungen

zur Folge: Ein Benutzerprogramm darf maximal 7 Startparameter besitzen und die relative Startadresse (im Assembler-Code durch die Marke (START) gekennzeichnet) darf nicht größer als 8190 sein.

B. Assembler

Im Assembler-Code werden KSP-Zellen in der Regel symbolisch adressiert (vgl. 6.1.2.). Dabei sind folgende Einschränkungen zu beachten:

1. Marken werden durch Namen gekennzeichnet. Als Name ist eine beliebige Folge von alphanumerischen Zeichen zulässig; jedoch haben nur die ersten fünf Zeichen unterscheidende Bedeutung.
2. Die Marke (START) ist für die Startadresse des Programms reserviert; auch die Marken (AFEHL) und (PFEHL) haben Sonderbedeutungen.
3. Symbolische Adressen dürfen mit einem numerischen Zuschlag in der Form z (MARKE) versehen werden. Ein solcher Zuschlag ist nicht zulässig, wenn die zugehörige Marke erst später deklariert wird.
4. Ein Zuschlag ist auch bei symbolischen Adressenangaben innerhalb von Z-Befehlen unzulässig.

C. Hardware

1. Doppelworte müssen im Kernspeicher eine durch vier teilbare Adresse haben.
2. Register 0 kann nicht als Basisadressregister verwendet werden.

3. Die Befehle LCBa,b,c und Fa,b,c besitzen den gleichen internen Operationscode. Ist die erste Registerangabe a gleich 15, so handelt es sich um den Unterprogrammsprung. LCB15,b,c ist also ein Unterprogrammsprung.

4. Codiert man relativ zum Befehlszählregister 15 (Θa,15,c), so muß man beachten, daß zum Zeitpunkt der Adressrechnung <15>+c der Befehlszähler immer auf der Adresse des nächsten Befehls steht.

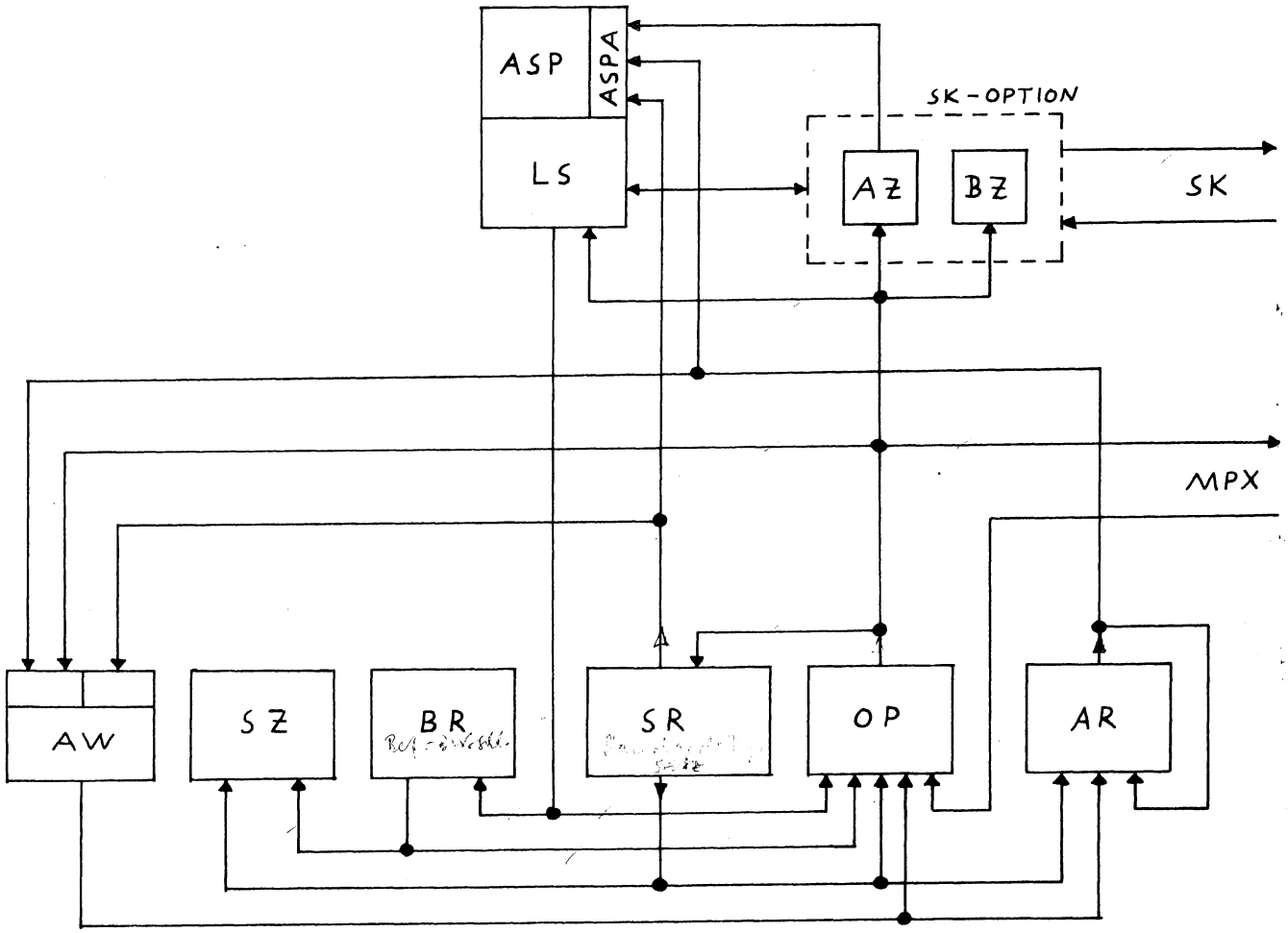
Delivery or duplication of this document and the use or communication of the contents thereof are forbidden without express authority. Offenders are liable to the payment of damages. All rights are reserved in the event of the grant of a Patent or the registration of a Utility Model.

10.3. ANHANG III

Blockschaltbild der Z 46

Zeichenerklärung:

ASP	=	Arbeitsspeicher
ASPA	=	Arbeitsspeicheranwahl
LS	=	Lese-Schreib-Register
SK	=	Schnellkanal (wahlweise)
AZ	=	Adresszähler für Schnellkanal
BZ	=	Bytezähler für Schnellkanal
MPX	=	Multiplex-Kanal
AW	=	Addierwerk
SZ	=	Schrittzähler (für Verschiebe- u. Transferbefehle)
BR	=	Befehlsregister (zur Befehlsentschlüsselung)
SR	=	Standard-Registersatz
OP	=	Operandenregister
AR	=	Adressregister



Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patenterteilung oder GM-Eintragung vorbehalten.