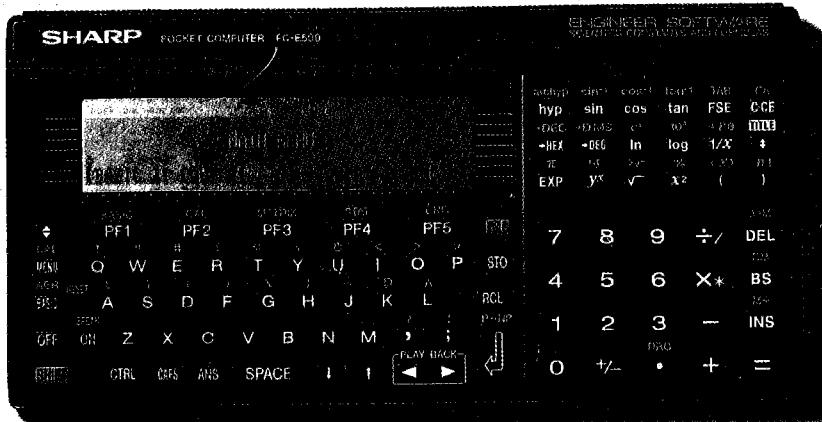


# Systemhandbuch zum Sharp PC-E500 Pocket Computer

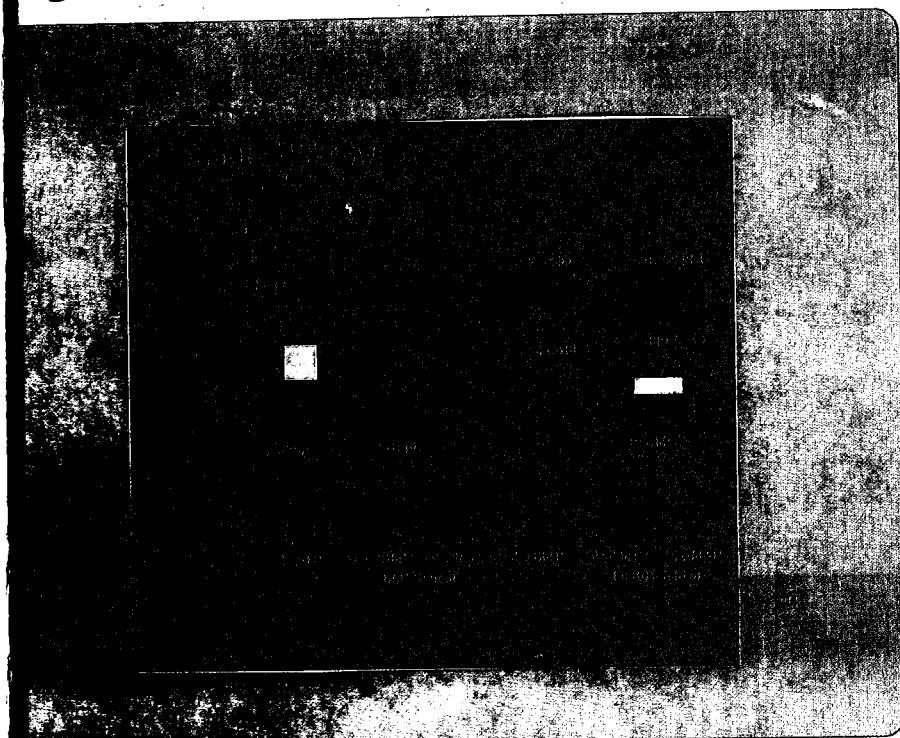


ISBN 3-89374-051-1  
Fischel GmbH

Dr. Joachim Stange

# Systemhandbuch zum Sharp PC-E500 Pocket Computer

*Gelöscht*



ISBN 3-89374-051-1  
Fischel GmbH

Dr. Joachim Stange

# **POCKET COMPUTER**

**FISCHEL GmbH**  
Zeitschrift für Taschencomputer

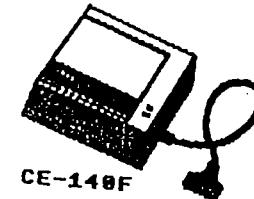
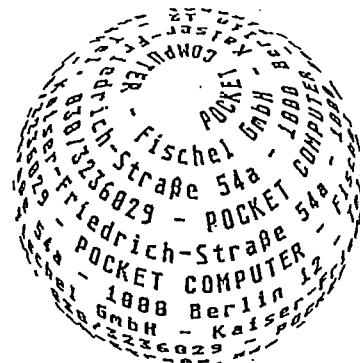
durch Information vorn

Kaiser-Friedrich-Straße 54a

1000 Berlin 12

Telefon (030)3236029

HRB 19396 Amtsgericht Charlottenburg



===== C FISCHEL GMBH =====

Alle Rechte vorbehalten. Ohne ausdrückliche Genehmigung des Herausgebers ist es nicht gestattet, das Buch oder Teile daraus auf fotomechanischem (Fotokopie, Mikrokopie) oder sonstigem Wege zu vervielfältigen. Es kann keine Haftung für die Richtigkeit der Programme übernommen werden, obwohl sie ausgetestet wurden.

===== 21 TXD 630+6 =====



\* 21 TXD 630/K6 \*

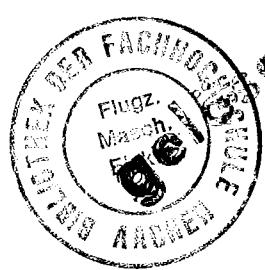
POCKET COMPUTER  
Fischel GmbH  
Kaiser-Friedrich-Straße 54a  
1000 Berlin 12 - Tel. 030/3236029

Bankverbindung: Postgiroamt Berlin-West, Bankleitzahl 10010010, Kontonummer 461533-103

Öffnungszeiten: Montag-Freitag 10.00-18.00 Uhr, Samstag 10.00-14.00

IS Inhaltsverzeichnis, Systemhandbuch

IS	Inhaltsverzeichnis, Systemhandbuch	1
IM	Inhaltsverzeichnis, Maschinensprachehandbuch	3
E	Einleitung, Systemhandbuch	5
A	Abkürzungen bei der Beschreibung der Befehle	6
1.0	Zusatzgeräte und Erweiterungen	7
2.0	Zahlensysteme, BCD, ASCII, Bit, Byte, RAM, ROM	9
3.0	Dumpprogramm zur Betrachtung des Speicherinhalts	11
4.0	Die Speichereinteilung	13
5.0	Die Variablen	14
5.1	Codierung von einfach und doppelt genauen Zahlen	15
5.2	Die Standardvariablen, Organisation im RAM	16
5.3	Numerische Variable einfacher Genauigkeit	18
5.4	Numerische Variable doppelter Genauigkeit	20
5.5	Die Zeichenvariablen	21
5.6	Numerische Feldvariable einfacher Genauigkeit	23
5.7	Numerische Feldvariable doppelter Genauigkeit	25
5.8	Felder für Zeichenvariable	26
5.9	Die Organisation der verschiedenen Typen von Variablen im Speicher, Kopf des Variablenbereiches	28
6.0	Das Basicprogramm	32
6.1	Aufbau des Basicprogrammes	33
6.2	Die Codierung von Zahlen im Basicprogramm	35
6.3	Die Tokentabelle	36
6.4	Tokentabelle, low	37
6.5	Tokentabelle, high	38
6.6	Unbekannte Basicbefehle	39
6.7	Die Organisation des Basicprogrammes im Speicher	40
7.0	RAM-Disketten, RAM-Karten	42
7.1	RAM-Disk F	43
7.2	RAM-Disk E	52
7.3	RAM-Disk G	53
8.0	Weitere Datenbereiche im RAM	55
8.1	Funktionstasten, Organisation im Arbeitsspeicher	56
8.2	AER-Eingaben, Organisation im RAM	57
8.3	Organisation der Techniker Software (ENG-Modus)	59
8.4	Statistische- und Matrix-Berechnungen	60
9.0	Zusammenspiel der Datenbereiche in RAM	61
10.0	Systembereich und Systemadressen	63
11.0	Zeichensätze	66
12.0	Der Mikroprozessor des PC-E500 und seine Architektur	71



TXD 630 +6

9/13658

12.1	Der Befehlssatz	74
12.2	Der Adressraum	75
12.3	Die Adressierung	76
12.4	Kurzbeschreibung der Befehle	79
13.0	Aufbau und Aufruf von Maschinenprogrammen	96
13.1	Basicbefehle zur Maschinenprogrammierung	97
13.2	Bereitstellung von Speicherraum für Maschinenprogramme	99
14.0	Befehlstabellen	100
S	Stichwortverzeichnis	104

<b>IM Inhaltsverzeichnis, Maschinensprachehandbuch</b>		
IM	Inhaltsverzeichnis, Maschinensprachehandbuch	1
IS	Inhaltsverzeichnis, Systemhandbuch	3
E	Einleitung, Maschinensprachehandbuch	5
A	Abkürzungen bei der Beschreibung der Befehle	6
1.0	Zusatzgeräte und Erweiterungen	6
2.0	Zahlensysteme, BCD, ASCII, Bit, Byte, RAM, ROM	8
3.0	Der Mikroprozessor des PC-E500 und seine Architektur	10
3.1	Das Interruptsystem	13
4.0	Der Befehlssatz	14
4.1	Der Adressraum	14
4.2	Die Adressierung	15
4.3	Transportbefehle	18
4.4	Immediate Befehle, laden der CPU-Register oder des inneren RAM mit Zahlenwerten	19
4.5	Transport zwischen den CPU-Registern	21
4.6	Transport zwischen den Registern des inneren RAM	22
4.7	Transport zwischen CPU-Registern und dem innerem RAM	26
4.8	Transport zwischen CPU-Registern und dem äußeren RAM	28
4.9	Transport zwischen dem inneren RAM und dem äußeren RAM	34
4.10	Blocktransport	46
4.11	Transport zwischen CPU-Registern und dem Stack	47
4.12	Sprungbefehle	49
4.13	Unterprogrammbefehle	51
4.14	Additions/Subtraktions Befehle	52
4.15	Pointer Addition Befehle	60
4.16	Inkrement/Dekrement Befehle	60
4.17	Logische Befehle	61
4.18	Bit-Test Befehle	63
4.19	Vergleich Befehle, Compare	64
4.20	Carry Flag Control Befehle	66
4.21	Swap- Rotate- und Shift-Befehle	67
4.22	Reset und Interrupt Befehle	69
4.23	CPU Control Befehle	70
5.0	Aufbau und Aufruf von Maschinenprogrammen	71
5.1	Basicbefehle zur Maschinenprogrammierung	71
5.2	Bereitstellung von Speicherraum für Maschinenprogramme	73
6.0	ROM-Routinen	74
6.1	Anzeigeroutinen	75
6.2	Tastenroutinen	81
6.3	Drucker-Routinen	87
7.0	Hilfsprogramme	89
7.1	Dumpprogramm	89
7.2	Disassembler	91
7.3	Breakpointmonitor	95

8.0	Die Speichereinteilung	99
9.0	Das Basicprogramm	100
9.1	Der Aufbau des Basicprogrammes	100
9.2	Die Tokentabelle	102
9.3	Tokentabelle, low	103
9.4	Tokentabelle, high	104
10.0	Kurzbeschreibung der Befehle	105
10.1	Befehlstabellen	122
S	Stichwortverzeichnis	126

## E Einleitung, Systemhandbuch

Da zum System des PC-E500 bisher kaum Informationen zu erhalten waren, ist das dargestellte Material im wesentlichen das Ergebnis von Tests, deren hier gegebene Interpretation möglicherweise nicht immer den tatsächlichen Gegebenheiten entspricht. Auch bleiben viele Fragen offen.

### Die Informationen zum System:

- Die Speichereinteilung
- Die Darstellung des Basicprogrammes und der dort verwandten Variablen.
- Die Organisation der RAM-Disketten.
- Die Verwaltung der Funktionstasten.
- Die Verwaltung von AER-Modus und ENG-Modus (Technikersoftware).
- Die Zeichensätze
- Die Systembereiche und Systemadressen.

### Die Informationen zum Prozessor:

- Die CPU
- Die Maschinensprache

Als Hilfsmittel zur Betrachtung des Speicherinhaltes wurde ein Dump-Programm entwickelt.

Die Basic-Programmierung des Sharp Pocket Computer PC-E500 ist in der mitgelieferten Bedienungsanleitung ausführlich beschrieben. Dort sind die Basic-Befehle PEEK, POKE, CALL, CSAVEM und CLOADM nicht erwähnt. Der PC-E500 verfügt jedoch über diese Befehle und damit ist neben der Basicprogrammierung auch die Programmierung in Maschinensprache möglich. Dies erweitert die Möglichkeiten des PC-E500 beträchtlich. Auch laufen Maschinenprogramme, die mit CALL Speicheradresse von Basicprogrammen oder von der Tastatur aufgerufen werden, vielmals schneller ab als Basicprogramme.

Die Maschinensprache der CPU des PC-E500 ist jedoch so vielfältig, daß hier nur eine Einführung gegeben werden kann. Eine ausführliche Darstellung ist einem Handbuch zur Maschinensprache vorbehalten, in dem auch ein Disassembler und ein Breakpointmonitor abgedruckt sind. Auch werden dort die Maschinenroutinen zur Bedienung der Anzeige, Tastatur und des Druckers behandelt.

Eine Einführung in die verschiedenen Zahlensysteme, insbesondere die Hexzahlen, und die wichtigsten Abkürzungen aus dem Gebiet der Mikroprozessoren soll dem Anänger den Einstieg in die Materie erleichtern. Die Darstellung von Adressen und Daten in Hexzahlen ist so praktisch und üblich, daß ihre Benutzung nicht zu umgehen ist.

Bezüglich der Schaltbilder sei auf das Sharp Service Manual zum PC-E500 verwiesen.

#### A Abkürzungen bei der Beschreibung der Befehle

A	Akkumulator, 8 Bit-Register
B	Hilfsakkumulator, 8 Bit-Register
BA	Doppelregister aus B und A, 16 Bit
C/Cy	Carry-Flag, 1 Bit.
	In der Befehlsliste bedeutet: C = Carry wird geändert . = wird nicht geändert 1 = wird 1 gesetzt 0 = wird 0 gesetzt
F	Flagregister, 8 Bit
I	I-Register, 16 Bit
IL	IL-Register, low Byte von I, 8 Bit
K/k	1 Byte, für Adresse oder Immediate
L/l	1 Byte, für Adresse oder Immediate
L	als letzter Buchstabe eines Befehlswortes bedeutet L Blockverarbeitung
LD	als letzte Buchstaben eines Befehlswortes bedeutet LD dekrementierender Blocktransport
M/m	1 Byte, für Adresse oder Immediate
N/n	1 Byte, für Adresse oder Immediate
P	als letzter Buchstabe eines Befehlswortes bedeutet P Transfer von drei Byte
PC	Programmcounter, 16 Bit
PS	Segmentregister, 4 Bit
R/r	Ein Register: A, IL, BA, I, X, Y, U, S
R <sub>1</sub> /r <sub>1</sub>	Ein 1-Byte Register: A, IL
R <sub>2</sub> /r <sub>2</sub>	Ein 2-Byte Register: BA, I
R <sub>3</sub> /r <sub>3</sub>	Ein 3-Byte Register: X, Y, U, S
S	S-Register, 20 Bit, Stackpointer für CALL/RET
U	U-Register, 20 Bit, Stackpointer für Datenstack
W	als letzter Buchstabe eines Befehlswortes bedeutet W Worttransfer
X	X-Register, 20 Bit
Y	Y-Register, 20 Bit
Z	Zero-Flag, 1 Bit
Zy	Zyklenzahl
(*)	Inhalt der durch * gegebenen Adresse des inneren RAM
[*]	Inhalt der durch * gegebenen Adresse des äußeren RAM
&	das Zeichen & vor einer Hexzahl wird im allgemeinen nicht benutzt, da im Zusammenhang mit den Befehlen nur Hexzahlen benutzt werden

#### 1.0 Zusatzgeräte und Erweiterungen

**RAM-Karten** sind Speichererweiterungen. Sharp bietet vier Karten mit verschiedenen Kapazitäten an:  
 CE-212M hat eine Kapazität von 8 KByte  
 CE-2H16M hat eine Kapazität von 16 KByte  
 CE-2H32M hat eine Kapazität von 32 KByte  
 CE-2H64M hat eine Kapazität von 64 KByte  
 Sie können in drei verschiedenen Speicherkonfigurationen verwandt werden:

**MEM\$="S1"**  
 Die RAM-Karte wird für die RAM-Diskette F verwandt.

**MEM\$="S2"**  
 Die RAM-Karte wird für die Variablen und das Basic-Programm verwandt. Für die RAM-Diskette E und die sonstigen Funktionen des PC-E500 wird das eingebaute RAM benutzt.

**MEM\$="B"**  
 Der Speicher der RAM-Karte wird dem Arbeitsspeicher des eingebauten RAM zugeschlagen, d.h. der Umfang des Arbeitsspeichers wird um die KByte der RAM-Karte erhöht. Die RAM-Diskette F steht in diesem Falle nicht zur Verfügung.

**CE-124 Cassetteninterface** ist eine Kabelverbindung zwischen dem Pocketcomputer und einem Cassettenrecorder. Es lassen sich damit Basicprogramme, Daten oder Maschinenprogramme auf Audiocassette speichern und wieder laden. Während jedoch mit dem CE-126P über die Remotesteuerung der Cassettenrecorder zum Laden/Speichern ein- und ausgeschaltet werden kann, ist dies beim CE-124 nicht möglich, da hier nur die beiden Audioleitungen vorhanden sind.

**CE-126P Thermodrucker/Cassetteninterface** ist ein Thermodrucker (24 Zeichen pro Zeile), der mit seinem Kabel an den Pocketcomputer angeschlossen wird. Der Drucker besitzt Anschlüsse für einen Cassettenrecorder. Es lassen sich damit Basicprogramme, Daten oder Maschinenprogramme auf Audiocassette speichern und wieder in den Pocketcomputer laden.

**CE-140F Diskettenlaufwerk** ist ein 2,5-Zoll Dikettenlaufwerk, das mit einem Kabel an den Pocketcomputer angeschlossen wird. Das Diskettenlaufwerk besitzt selbst wieder einen Steckkontakt, an den der CE-126P Thermodrucker/Cassetteninterface angeschlossen werden kann. Folgende Befehle bedienen die Disk: CHAIN, CLOSE, COPY, DSKF, EOF, FILES, INPUT#, INIT, KILL, LFILES, LOAD, LOC, LOF, MERGE, NAME, OPEN, PRINT#, SAVE, und SET.

**CE-515P Farbplotter/Drucker** ist ein Vierfarben-Drucker (schwarz, blau, grün, rot), mit dem Programme, Rechenergebnisse oder Grafiken mehrfarbig bis zu DIN A4 Format ausgedruckt werden können. Zum Anschluß an das serielle Interface ist das Kabel CE-516L erforderlich.

**Transfile PC-E500** gibt die Möglichkeit, den PC-E500 über die serielle Schnittstelle an einen PC anzuschließen. Es wird von verschiedenen Herstellern über die einschlägigen Händler oder Zeit-

schriften angeboten.

CE-130T ist ein Pegelkonverter für den Anschluß der RS-232C Schnittstelle an einen Personal Computer.

In den PC-E500 eingebaute RAM-Erweiterungen auf 128 oder 256 KByte werden von verschiedenen Herstellern über die einschlägigen Zeitschriften angeboten.

RAM-Karten mit 128 oder 256 KByte werden ebenfalls von verschiedenen Herstellern über die einschlägigen Zeitschriften angeboten.

## 2.0 Zahlensysteme, BCD, ASCII, Bit, Byte, RAM, ROM

Das Dezimalsystem sind wir gewohnt aus dem täglichen Leben. Dies ist jedoch nicht geeignet für die Durchführung von Rechenoperationen im Computer. Der Computer ist in seinem Kern von recht beschränkter Intelligenz und kann nur zwischen zwei Zuständen, niedrige/hohe Spannung, unterscheiden. Dies führt zum Dualsystem, was später genauer erläutert wird. Dieses, auf den beiden Zahlen Null und Eins aufbauende Zahlensystem, wird jedoch nur im Kern des Computers, der CPU, bei den Grundrechenoperationen angewandt. Die Rechenergebnisse werden per Programm in andere Zahlensysteme umgewandelt und so ausgegeben, so daß der Benutzer des Computers die Ergebnisse nicht in Dualzahlen sondern im allgemeinen in Dezimalzahlen erhält. Oft werden die Ergebnisse jedoch auch in Oktalzahlen oder in Hexadezimalzahlen ausgegeben. Diese beiden Zahlensysteme benutzt man, weil sich mit ihnen die Computerworte besser darstellen lassen. Z.B. läßt sich das bei Mikrocomputern häufig benutzte 8-Bit Wort am einfachsten und übersichtlichsten durch zwei Hexadezimalziffern darstellen.

### Dezimalzahlen

$$1234 = 4 \cdot 1 + 3 \cdot 10 + 2 \cdot 100 + 1 \cdot 1000$$

oder

$$1234 = 4 \cdot 10^0 + 3 \cdot 10^1 + 2 \cdot 10^2 + 1 \cdot 10^3$$

Die Dezimalzahlen werden also in Vielfachen von Zehnerpotenzen dargestellt. Das gleiche Prinzip wird auch für die anderen Zahlensysteme angewandt.

### Dualzahlen

Dualzahlen werden in Vielfachen von Zweierpotenzen dargestellt.

$$\text{Dual } 10111 = 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4 + 0 \cdot 8 + 1 \cdot 16$$

oder

$$\text{Dual } 10111 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 \\ = 23 \text{ (dezimal)}$$

### Oktalzahlen

Oktalzahlen werden in Vielfachen von Achterpotenzen dargestellt.

$$\text{Oktal } 70243 = 3 \cdot 1 + 4 \cdot 8 + 2 \cdot 64 + 0 \cdot 512 + 7 \cdot 4096$$

oder

$$\text{Oktal } 70243 = 3 \cdot 8^0 + 4 \cdot 8^1 + 2 \cdot 8^2 + 0 \cdot 8^3 + 7 \cdot 8^4 \\ = 28835 \text{ (dezimal)}$$

### Hexadezimalzahlen

Hexadezimalzahlen werden in Vielfachen von Sechzehnerpotenzen dargestellt. Für eine Stelle werden hierbei sechzehn Möglichkeiten benötigt. Dazu reichen die Zahlen von Null bis Neun nicht aus. Darum wird gezählt: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Es ist A = 10(dezimal), B = 11(dezimal), C = 12(dezimal), D = 13(dezimal), E = 14(dezimal) und F = 15(dezimal).  
Hexadezimal 9E4A1 = 1 \* 1 + 10 \* 16 + 4 \* 256 + 14 \* 4096 + 9 \* 65536  
oder

$$\text{Hexadezimal } 9E4A1 = 1 \cdot 16^0 + 10 \cdot 16^1 + 4 \cdot 16^2 + 14 \cdot 16^3 + 9 \cdot 16^4 \\ = 648353 \text{ (dezimal)}$$

#### Binary Coded Decimals, BCD

Häufig müssen Dezimalzahlen in Computerworten dargestellt werden. Es werden dabei für eine Dezimalstelle 4 Bit (4 Dualstellen) benötigt. So lassen sich in einem 8-Bit Computerwort zwei Dezimalstellen unterbringen.  
Beispiele: 13 = 00010011, 57 = 01010111, 99 = 10011001.

#### ASCII-Code

Mit dem ASCII-Code werden Texte, die angezeigt oder ausgedruckt werden sollen, codiert. Jedes Zeichen wird durch eine achtstellige Dualzahl oder durch eine zweistellige Hexadezimalzahl oder durch ein dreistellige Dezimalzahl dargestellt. Es lassen sich 256 Zeichen durch die Zahlen von 0 bis 255 codieren. Die Abkürzung ASCII steht für American Standard Code of Information Interchange. Eine Tabelle der Zuordnung zwischen Zeichen und Code finden Sie in der Bedienungsanleitung des Pocket Computer.

#### Bit und Byte

Das Bit ist eine Dualstelle. Es kann damit von Null bis Eins gezählt, also zwei Zustände dargestellt werden. Das Byte dagegen ist eine achtstellige Dualzahl. Das Byte umfaßt  $2^8 = 256$  Möglichkeiten. Es kann von Null bis 255 gezählt werden.

#### RAM und ROM

RAM und ROM sind Bezeichnungen für Speicherbereiche. RAM (Random Access Memory) ist ein Speicherbereich mit sogenanntem wahlfreiem Zugriff. In diesen Speicherbereich kann Information (Computerworte, Daten, Text) sowohl hineingeschrieben als auch wieder herausgelesen werden. ROM (Read Only Memory) ist dagegen ein Speicherbereich, der nur gelesen aber nicht beschrieben werden kann. Genauer gesagt heißt das, daß das ROM mit Spannungen, die zum normalen Betrieb des Computers benutzt werden, nicht beschrieben werden kann. Das ROM wird im allgemeinen vom Hersteller des Computers mit speziellen Verfahren mit Programmen, z.B. dem Betriebssystem, gefüllt. Der Arbeitsspeicher der Computer dagegen wird aus RAM-Bausteinen aufgebaut.

#### 3.0 Dumpprogramm zur Betrachtung des Speicherinhalts

Mit einem Basicprogramm werden die Hexcode und Zeichen eines Speicherbereiches angezeigt oder ausgedruckt. Auf der Anzeige erscheint auch ganz rechts die jeweilige Quersumme. Es gibt jedoch keinen Zwischenraum zwischen den Zeichen und der Quersumme. Sollte das stören, so läßt sich die Quersumme beseitigen, durch 2125: PRINT ""

#### Anzeige:

Aufruf mit RUN\*D oder GOTO\*D  
Eingabe der Anfangsadresse

Anzeige von je acht Hexcode und der zugehörigen Zeichen. Auf den letzten beiden Stellen der Anzeige erscheint, ohne Zwischenraum zum letzten Zeichen, die zweistellige Hex-Quersumme.

Taste Pfeil nach unten	Nächste Zeile anzeigen
Taste Pfeil nach oben	Letzte Zeile anzeigen
Taste +	Laufende Anzeige vorwärts
Taste -	Laufende Anzeige rückwärts
Taste =	Anhalten der laufenden Anzeige
Taste M	Neue Adresse eingeben
Taste C.CE	Ende, Rückkehr zum Basicinterpreter. Nach ca. 5 Minuten Wartestellung: END (Nicht bei laufender Anzeige)

#### Ausdruck:

Aufruf mit RUN\*P  
Eingabe der Anfangs- und Endadresse  
Ausdruck von je vier Hexcode und der zugehörigen Zeichen mit dem Thermodrucker CE-126P

#### Listing: PC-E500 DUMP

```
2000:*D:P=0:K=7:GOTO 2020
2010:*P:P=1:K=3:PRINT =LPRINT
2020:INPUT "ANF-ADR ";I:G=0:H=1:J=&FFFFF
2030:IF I<&40000 OR I>&FFFFF PRINT "&40000 > ADRESSE > &FFFFF":END
2032:IF I>J END
2035:IF P=1 AND H=1 INPUT "END-ADR ";J:G=1:H=0
2040:IF J<I OR J>&FFFFF PRINT "END < ANF ODER END > &FFFFF":END
2050:PRINT RIGHTS ("0000"+HEX$ I+":",6);:IF P=1 PRINT " ";
2060:T=1:S=0
2070:FOR N=0 TO K
2080:Q=PEEK (I+N):PRINT RIGHTS ("0"+HEX$ Q+" ",3);:S=S+Q
2090:NEXT N
2100:FOR N=0 TO K
2110:Q=PEEK (I+N):IF Q < &20 LET Q = &20
2115:PRINT CHR$ Q;
2120:NEXT N
2125:IF P=1 PRINT "" ELSE PRINT RIGHTS ("0"+HEX$ S,2)
2130:F=ASC INKEY$
```

```
2140:IF F=77 GOTO 2020
2150:IF F=2 END
2160:IF F=43 LET G=1
2165:IF F=45 LET G=2
2170:IF F=61 LET G=0
2180:IF G=1 LET I=I+1+K:GOTO 2030
2185:IF G=2 LET I=I-1-K:GOTO 2030
2190:IF F=5 LET I=I+1+K:GOTO 2030
2200:IF F=4 LET I=I-1-K:GOTO 2030
2210:T=T+1:IF T>9999 END
2220:GOTO 2130
```

#### 4.0 Die Speichereinteilung

##### RAM-Karten, MEM\$="S1", für RAM-Disk F:

8K RAM-Karte:	40000 - 41FFF
16K RAM-Karte:	40000 - 43FFF
32K RAM-Karte:	40000 - 47FFF
64K RAM-Karte:	40000 - 4FFFF

##### Arbeitsspeicher für Variable, Basicprogramm, Funktionstasten, AER usw.:

Nur eingebautes RAM:	B8000 - BFBFF
Eingebautes RAM und RAM-Karte, MEM\$ = "S1":	B8000 - BFBFF
Eingebautes RAM und 8K-RAM-Karte, MEM\$ = "B":	B6000 - BFBFF
Eingebautes RAM und 16K-RAM-Karte, MEM\$ = "B":	B4000 - BFBFF
Eingebautes RAM und 32K-RAM-Karte, MEM\$ = "B":	B0000 - BFBFF
Eingebautes RAM und 64K-RAM-Karte, MEM\$ = "B":	A8000 - BFBFF

Systembereich: BF21B - BFBFF

Systemadressen: BFC00 - BFFFF

ROM mit Betriebssystem und Basicinterpreter: C0000 - FFFFF



## 5.0 Die Variablen

Es gibt numerische Variable und Zeichen- oder Textvariable. Für beide können auch Felder eingerichtet werden. Die numerischen Variablen können mit einfacher oder doppelter Genauigkeit, letztere gekennzeichnet durch ein angehängtes #, verarbeitet werden. Die Namen aller Variablen können bis zu 40(dez) alphanumerische Zeichen enthalten, müssen jedoch stets mit einem Buchstaben beginnen. Die Texte der Zeichenvariablen können bis zu 254(dez) Zeichen umfassen.

Es gibt nur eine Sorte von Standardvariablen, die numerischen Standardvariablen, die nur einfache Genauigkeit haben. Deren Namen bestehen aus nur einem Buchstaben, A bis Z. Für sie sind im Speicher stets 312(dez) Byte reserviert, d.h. 12(dez) Byte für jede Standardvariable. Dieser Speicherbereich liegt jedoch nicht immer an der gleichen Stelle. Es spart Speicher, möglichst die numerischen Standardvariablen zu benutzen.

Die numerischen Variablen einfacher Genauigkeit mit mindestens zwei Zeichen im Namen müssen stets mit einem Buchstaben beginnen. Sie werden in der Bedienungsanleitung als einfache numerische Variable bezeichnet.

Numerische Variable einfacher Genauigkeit können bis zu zehn Stellen enthalten. Ihre Exponenten haben zwei Stellen, reichen also von -99 bis +99. Der mögliche Zahlenbereich reicht von +/-9.99999999 E-99 bis +/-9.99999999 E+99.

Numerische Variable doppelter Genauigkeit können dagegen bis zu zwanzig Stellen enthalten. Der Exponentenbereich ist der gleiche wie bei einfacher Genauigkeit.

Zeichenvariable werden durch ein angehängtes \$-Zeichen von den numerischen unterschieden. Es können hier wieder einzelne Buchstaben als Name benutzt werden, also z.B. A\$. Der Text, der einer Zeichenvariablen zugewiesen wird, darf bis zu 254(dez) Zeichen enthalten.

Felder können sowohl für numerische als auch für Textvariable eingerichtet werden. Die Namen der Felder können auch wieder aus einzelnen Buchstaben bestehen. Die Felder können bis zu 120(dez) Dimensionen haben. Jeder einzelne Index beginnt bei Null zu zählen und kann bis 65534 = &FFFE reichen. Durch DIM A(3) werden die vier Variablen A(0), A(1), A(2) und A(3) eingerichtet.

In den weiteren Abschnitten dieses Kapitels wird erst die Codierung der einzelnen Typen von Variablen im Speicher behandelt und danach die Organisation des gesamten Bereiches für die Variablen im Speicher beschrieben. Im Kopf dieses Bereiches sind die Adressen der Einzelbereiche für die verschiedenen Typen von Variablen angegeben.

## 5.1 Die Codierung von einfach und doppelt genauen Zahlen

Einfach genaue Zahlen können bis zu 10 Stellen haben. Ihre Exponenten reichen von -99 bis +99. Sie werden mit 7 Byte codiert. Die einzelnen Byte haben die folgende Bedeutung:

1. Byte = 00 für positive Zahl.  
= 08 für negative Zahl
2. Byte = Exponent bei positivem Exponent.  
= 00 - Betrag des Exponent bei negativem Exponent.
3. Byte = 1. und 2. Stelle der Zahl.
4. Byte = 3. und 4. Stelle der Zahl.
5. Byte = 5. und 6. Stelle der Zahl.
6. Byte = 7. und 8. Stelle der Zahl.
7. Byte = 9. und 10. Stelle der Zahl.

Beispiele:

+1234567 wird codiert durch: 00 06 12 34 56 70 00  
-1234567 wird codiert durch: 08 06 12 34 56 70 00  
+0.00123 wird codiert durch: 00 FD 12 30 00 00 00

Doppelt genaue Zahlen können bis zu 20 Stellen haben. Ihre Exponenten reichen ebenfalls von -99 bis +99. Sie werden mit 12 Byte codiert. Die einzelnen Byte haben die folgende Bedeutung:

1. Byte = 01 für positive Zahl.  
= 09 für negative Zahl
2. Byte = Exponent bei positivem Exponent.  
= 00 - Betrag des Exponent bei negativem Exponent.
3. Byte = 1. und 2. Stelle der Zahl.
4. Byte = 3. und 4. Stelle der Zahl.
5. Byte = 5. und 6. Stelle der Zahl.
6. Byte = 7. und 8. Stelle der Zahl.
7. Byte = 9. und 10. Stelle der Zahl.
8. Byte = 11. und 12. Stelle der Zahl.
9. Byte = 13. und 14. Stelle der Zahl.
10. Byte = 15. und 16. Stelle der Zahl.
11. Byte = 17. und 18. Stelle der Zahl.
12. Byte = 19. und 20. Stelle der Zahl.

Beispiele:

+123456789012345 wird codiert durch:  
01 0E 12 34 56 78 90 12 34 50 00 00  
-123456789012345 wird codiert durch:  
09 0E 12 34 56 78 90 12 34 50 00 00  
+0.0000123456789 wird codiert durch:  
01 FB 12 34 56 78 90 00 00 00 00 00

### 5.2 Die Standardvariablen, Organisation im RAM

Für die numerischen Standardvariablen mit den aus einem Buchstaben bestehenden Namen A bis Z ist im RAM-Speicher ein Platz von 312(dez) Byte reserviert. Die Benutzung der Standardvariablen kostet also keinen Speicherplatz.

Der Bereich für die numerischen Standardvariablen reicht bei nur eingebautem RAM ursprünglich, also nach ON/Reset und Speicher löschen, im RAM von B8064 bis B819B. Dieser Bereich wird jedoch zu höheren Adressen verschoben, wenn Felder definiert werden. Jede Variable ist mit 12(dez) Byte codiert. Die Codierung entspricht der der Zahlen einfacher Genauigkeit, enthält jedoch noch Zusatzinformationen.

Die einzelnen Byte haben die folgende Bedeutung:

1. Byte = Länge des Namens, 1 bei den Standardvariablen
2. Byte = Länge des Codierbereiches für die einzelne Variable, bei Standardvariablen stets 0C
3. Byte = 00
4. Byte = 00
5. Byte = ASCII Code des Namens der Variable
6. Byte = 00 für positive Zahl.  
= 08 für negative Zahl
7. Byte = Exponent bei positivem Exponent.  
= 00 - Betrag des Exponent bei negativem Exponent.
8. Byte = 1. und 2. Stelle der Zahl.
9. Byte = 3. und 4. Stelle der Zahl.
10. Byte = 5. und 6. Stelle der Zahl.
11. Byte = 7. und 8. Stelle der Zahl.
12. Byte = 9. und 10. Stelle der Zahl.

Anschließend ist der Speicherbereich der numerischen Standardvariablen dargestellt. Es sind einige Beispiele für die Darstellung der Zahlen gegeben. Nach CLEAR haben alle Standardvariablen den Wert Null.

Variable	Adresse	Inhalt
A=0	B8064	01 0C 00 00 41 00 00 00 00 00 00 00
B=1	B8070	01 0C 00 00 42 00 00 10 00 00 00 00
C=-1	B807C	01 0C 00 00 43 08 00 10 00 00 00 00
D=12	B8088	01 0C 00 00 44 00 01 12 00 00 00 00
E=123	B8094	01 0C 00 00 45 00 02 12 30 00 00 00
F=1234567891	B80A0	01 0C 00 00 46 00 09 12 34 56 78 91
G=-1234567891	B80AC	01 0C 00 00 47 08 09 12 34 56 78 91
H=123.456	B80B8	01 0C 00 00 48 00 02 12 34 56 00 00
I=-123.456	B80C4	01 0C 00 00 49 08 02 12 34 56 00 00
J=1.2346E+7	B80D0	01 0C 00 00 4A 00 07 12 34 56 00 00
K=-1.23456E-5	B80DC	01 0C 00 00 4B 08 FB 12 34 56 00 00
L=0	B80E8	01 0C 00 00 4C 00 00 00 00 00 00 00
M=0	B80F4	01 0C 00 00 4D 00 00 00 00 00 00 00
N=0	B8100	01 0C 00 00 4E 00 00 00 00 00 00 00
O=0	B810C	01 0C 00 00 4F 00 00 00 00 00 00 00
P=0	B8118	01 0C 00 00 50 00 00 00 00 00 00 00
Q=0	B8124	01 0C 00 00 51 00 00 00 00 00 00 00

R=0	B8130	01 0C 00 00 52 00 00 00 00 00 00 00
S=0	B813C	01 0C 00 00 53 00 00 00 00 00 00 00
T=0	B8148	01 0C 00 00 54 00 00 00 00 00 00 00
U=0	B8154	01 0C 00 00 55 00 00 00 00 00 00 00
V=0	B8160	01 0C 00 00 56 00 00 00 00 00 00 00
W=0	B816C	01 0C 00 00 57 00 00 00 00 00 00 00
X=0	B8178	01 0C 00 00 58 00 00 00 00 00 00 00
Y=0	B8184	01 0C 00 00 59 00 00 00 00 00 00 00
Z=0	B8190	01 0C 00 00 5A 00 00 00 00 00 00 00
	B819C	

### 5.3 Numerische Variable einfacher Genauigkeit

Die numerischen Variablen einfacher Genauigkeit, abgesehen von den Standardvariablen, haben Namen mit mindestens zwei Zeichen und höchstens 40 Zeichen. Das erste Zeichen muß ein Buchstabe sein. Diese Variablen werden in der Bedienungsanleitung als einfache Variable bezeichnet, im Gegensatz zu den Feldvariablen.

Diese einfachen numerischen Variablen werden im RAM unmittelbar hinter den numerischen Standardvariablen abgelegt. Wenn keine Felder eingerichtet sind, beginnen sie ab der Speicher-Adresse B819C. Sind Felder eingerichtet, so verschiebt sich diese Adresse zu höheren Werten, da Felder vor den numerischen Standardvariablen abgelegt werden.

Ihre Codierung entspricht der der Zahlen einfacher Genauigkeit, jedoch enthält sie zusätzliche Angaben über den Namen der Variablen. Die einzelnen Byte haben folgende Bedeutung:

1. Byte = Länge des Namens, L
2. Byte = Länge des Codierbereiches für die einzelne Variable,  
= 4 + L + 7
3. Byte = 00
4. Byte = 00
5. Byte = ASCII Code des 1. Zeichens des Namens
- .....
- 4+L. Byte = ASCII Code des letzten Zeichens des Namens
- 4+L+1. Byte = 00 für positive Zahl.  
= 08 für negative Zahl
- 4+L+2. Byte = Exponent bei positivem Exponent.  
= 00 - Betrag des Exponent bei negativem Exponent.
- 4+L+3. Byte = 1. und 2. Stelle der Zahl.
- 4+L+4. Byte = 3. und 4. Stelle der Zahl.
- 4+L+5. Byte = 5. und 6. Stelle der Zahl.
- 4+L+6. Byte = 7. und 8. Stelle der Zahl.
- 4+L+7. Byte = 9. und 10. Stelle der Zahl.

Beispiele:

Es seien keine Felder definiert und es wurde im RUN-Mode eingegeben:

ZAHL=1234567

ANFANGSWERT = -0.0056789

Der RAM-Speicher hat dann folgenden Inhalt:

B8190: 01 0C 00 00	B8190 - B819C enthält die Standardvariable Z
B8194: 5A 00 00 00 Z	
B8198: 00 00 00 00	
B819C: 04 0F 00 00	B819C - B81AA enthält die Variable
B81A0: 5A 41 48 4C ZAHL	ZAHL = 1234567
B81A4: 00 06 12 34	
B81A8: 56 70 00	
B81AB: 0B 16 00 00	B81AB - B81C4 enthält die Variable
B81AF: 41 4E 46 41 ANFA	Anfangswert = -0.0056789

B81B3: 4E 47 53 57 NGSW  
B81B7: 45 52 54 08 ERT  
B81BB: FD 56 78 90  
B81C3: 00 00

#### 5.4 Numerische Variable doppelter Genauigkeit

Bezüglich der Namensgebung gilt für die numerischen Variablen doppelter Genauigkeit das gleiche wie für die einfachen numerischen Variablen einfacher Genauigkeit. Doppelte Genauigkeit wird durch das an den Namen angehängte Zeichen # gekennzeichnet.

Im RAM-Speicher werden die Variablen doppelter Genauigkeit unmittelbar hinter den einfachen Variablen einfacher Genauigkeit abgelegt. Die Anfangsadresse des Bereiches für die Variablen doppelter Genauigkeit verschiebt sich zu höheren Adressen, wenn Felder definiert werden oder wenn weitere einfache Variable einfacher Genauigkeit eingegeben werden.

Ihre Codierung entspricht der der Zahlen doppelter Genauigkeit, jedoch enthält sie zusätzliche Angaben über den Namen der Variablen. Die einzelnen Byte haben folgende Bedeutung:

1. Byte = Länge des Namens, L
2. Byte = Länge des Codierbereiches für die einzelne Variable,  
= 4 + L + 12
3. Byte = 00
4. Byte = 00
5. Byte = ASCII Code des 1. Zeichens des Namens  
.....
- 4+L. Byte = ASCII Code des letzten Zeichens des Namens
- 4+L+1. Byte = 01 für positive Zahl.  
= 09 für negative Zahl
- 4+L+2. Byte = Exponent bei positivem Exponent.  
= 00 - Betrag des Exponent bei negativem Exponent.
- 4+L+3. Byte = 1. und 2. Stelle der Zahl.
- 4+L+4. Byte = 3. und 4. Stelle der Zahl.
- 4+L+5. Byte = 5. und 6. Stelle der Zahl.
- 4+L+6. Byte = 7. und 8. Stelle der Zahl.
- 4+L+7. Byte = 9. und 10. Stelle der Zahl.
- 4+L+8. Byte = 11. und 12. Stelle der Zahl.
- 4+L+9. Byte = 13. und 14. Stelle der Zahl.
- 4+L+10. Byte = 15. und 16. Stelle der Zahl.
- 4+L+11. Byte = 17. und 18. Stelle der Zahl.
- 4+L+12. Byte = 19. und 20. Stelle der Zahl.

Beispiele:

Es seien keine Felder definiert und es wurde im RUN-Mode bereits ZAHL=1234567 und ANFANGSWERT = -0.0056789 eingegeben.

Weiterhin wurde eingegeben:  
DOPPELT# = -5678.9012345678901234

Der RAM-Speicher hat dann im Anschluß an den Speicherinhalt der Beispiele für die Variablen einfacher Genauigkeit folgenden Inhalt:

B81C1: 07 17 00 00  
B81C5: 44 4F 50 50 DOPP  
B81C9: 45 4C 54 09 ELT

B81CD: 03 56 78 90  
B81D1: 12 34 56 78  
B81D5: 90 12 34

#### 5.5 Die Zeichenvariablen

Zeichenvariable werden durch ein angehängtes \$-Zeichen von den numerischen unterschieden. Es können hier wieder einzelne Buchstaben als Name benutzt werden, also z.B. A\$. Der Text, der einer Zeichenvariablen zugewiesen wird, darf bis zu 254(dez) Zeichen enthalten.

Ihre Organisation im Speicher erfolgt in zwei Teilen. Für die Textvariablen wird zuerst in einer Liste Name, Textort und Textlänge abgelegt. Erst anschließend an diese Liste aller Textvariablen werden die Texte selbst niedergelegt.

In der Liste der Namen, Textorte und Textlängen werden die einzelnen Textvariablen folgendermaßen codiert:

1. Byte = Länge des Namens(ohne \$), L
2. Byte = Länge des Codierbereiches für die einzelne Variable,  
= 4 + L + 4
3. Byte = 00
4. Byte = 00
5. Byte = ASCII Code des 1. Zeichens des Namens  
.....
- 4+L. Byte = ASCII Code des letzten Zeichens des Namens
- 4+L+1. Byte = low Byte für die Adresse des Textortes, die vom Beginn des Textbereiches ab gezählt wird (dieser ist im Kopf des Variablenbereiches angegeben)
- 4+L+2. Byte = high Byte für die Adresse des Textortes
- 4+L+3. Byte = 0
- 4+L+4. Byte = Textlänge

Beispiele:

Es seien keine Felder definiert und es wurde im RUN-Mode bereits ZAHL=1234567, ANFANGSWERT = -0.0056789 und DOPPELT# = -5678.9012345678901234 eingegeben.

Weiterhin wurde eingegeben:  
A\$ = "PC-E500"  
TAG\$ = "Donnerstag"

Der RAM-Speicher hat dann im Anschluß an den Speicherinhalt der Beispiele für die Variablen einfacher und doppelter Genauigkeit folgenden Inhalt:

B81D8: 01 09 00 00	Definition der Textvariablen
B81DC: 41 00 00 00 A	A\$ = "PC-E500"
B81E0: 07	Platz 00, d.h. unmittelbar hinter den Definitionen
B81E1: 03 0B 00 00	Definition der Textvariablen
B81E5: 54 41 47 07 TAG	Tag\$ = "Donnerstag"
B81E9: 00 00 0A	Platz 07, d.h. hinter den 7 Byte Text der 1. Textvariablen

B81EC: 50 43 2D 45 PC-E  
B81F0: 35 30 30 500

Text zur Textvariablen  
A\$ = "PC-E500"

B81F3: 44 6F 6E 6E Donn  
B81F7: 65 72 73 74 erst  
B81FB: 61 67 ag

Text zur Textvariablen  
Tag\$ = "Donnerstag"

### 5.6 Numerische Feldvariable einfacher Genauigkeit

Für die Namen von numerischen Feldvariablen einfacher Genauigkeit können auch einzelne Buchstaben benutzt werden. Wird mit der DIM-Anweisung ein Feld definiert, so wird die zugehörige Codierung am Anfang des Speicherbereiches für die Variablen abgelegt. Die dort ursprünglich vorhandenen numerischen Standard-Variablen werden zu höheren Adressen verschoben. Anfangs wird für jede Variable der Wert Null eingesetzt. Die Codierung dieser numerischen Felder entspricht der der Zahlen einfacher Genauigkeit, jedoch enthält sie zusätzliche Angaben über den Namen der Variablen und über die Dimensionen des Feldes. Die einzelnen Byte haben folgende Bedeutung:

1. Byte = Länge des Namens, L
2. Byte = Länge des Codierbereiches für das gesamte Feld,  
low Byte
3. Byte = Länge des Codierbereiches für das gesamte Feld,  
medium Byte
4. Byte = Länge des Codierbereiches für das gesamte Feld,  
high Byte  
Gesamtänge =  $4 + L + 1 + 2*D + 7*(\text{Anzahl der Feldelemente})$
5. Byte = ASCII Code des 1. Zeichens des Namens  
.....  
4+L. Byte = ASCII Code des letzten Zeichens des Namens  
4+L+1. Byte = Anzahl der Dimensionen des Feldes, D  
5+L+1. Byte = low Byte des 1. Index  
5+L+2. Byte = high Byte des 1. Index  
.....  
5+L+2D-1. Byte = low Byte des letzten Index  
5+L+2D. Byte = high Byte des letzten Index  
5+L+2D+1. - 5+L+2D+7. Byte = Codierung des Zahlenwertes des ersten Feldelementes  
.....

#### Beispiel:

Es wird im RUN-Mode oder beim Ablauf eines Basicprogrammes eingegeben:  
DIM M(2).

Der RAM-Speicher hat danach folgenden Inhalt:

B8064: 01 1D 00 00	B8064+1D=B8081=nächste Variable
B8068: 4D 01 02 00 M	01=eine Dimension, 0002=Wert des Index
B806C: 00 00 00 00	B806C-B8073 = 7 Plätze für M(0)
B8070: 00 00 00 00	B8074-B8079 = 7 Plätze für M(1)
B8074: 00 00 00 00	
B8078: 00 00 00 00	B807A-B8080 = 7 Plätze für M(2)
B8080: 00 00 00 00	
B8080: 00 01 0C 00	B8081 = Beginn der Stand.-variable A
B8084: 00 41 00 00 A	

Werden nun die Werte für die Variablen eingegeben:  
M(0) = 2345678901, M(1) = -0.003456789012, M(2) = 4567.890123,

so ergibt sich folgender Speicherinhalt:

B8064: 01 1D 00 00	B8064+1D=B8081=nächste Variable
B8068: 4D 01 02 00 M	01=eine Dimension, 0002=Wert des Index
B806C: 00 09 23 45	B806C-B8073 = 7 Plätze für M(0)
B8070: 67 89 01 08	B8074-B8079 = 7 Plätze für M(1)
B8074: FD 34 56 78	
B8078: 90 12 00 03	B807A-B8080 = 7 Plätze für M(2)
B807C: 45 67 89 01	
B8080: 23 01 0C 00	B8081 = Beginn der Stand.-variable A
B8084: 00 41 00 00 A	

Beispiel:

Ein weiteres Beispiel zeige die Verhältnisse bei einem längeren Namen und bei mehreren Dimensionen.

Es wird im RUN-Mode oder beim Ablauf eines Basicprogrammes eingegeben:

```
DIM NUMFELD(1,2)
NUMFELD(0,0)=5006789, NUMFELD(0,1)=5016789, NUMFELD(0,2)=5026789
NUMFELD(1,0)=5106789, NUMFELD(1,1)=5116789, NUMFELD(1,2)=5126789
```

Der RAM-Speicher hat danach anschließend an die Werte des letzten Beispiels folgenden Inhalt:

B8081: 07 3A 00 00	07 = Länge von NUMFELD
B8085: 4E 55 4D 46 NUMF	B8081+3A=B80BB=nächste Variable
B8089: 45 4C 44 02 ELD	02 = Anzahl der Dimensionen
B808D: 01 00 02 00	0001 = 1. Index, 0002 = 2. Index
B8091: 00 06 50 06	B8091-B8097 = 7 Plätze für NUMFELD(0,0)
B8095: 78 90 00 00	B8098-B809E = 7 Plätze für NUMFELD(0,1)
B8099: 06 50 16 78	
B809D: 90 00 00 06	B809F-B80A5 = 7 Plätze für NUMFELD(0,2)
B80A1: 50 26 78 90	
B80A5: 00 00 06 51	B80A6-B80AC = 7 Plätze für NUMFELD(1,0)
B80A9: 06 78 90 00	
B80AD: 00 06 51 16	B80AD-B80B3 = 7 Plätze für NUMFELD(1,1)
B80B1: 78 90 00 00	B80B4-B80BB = 7 Plätze für NUMFELD(1,2)
B80B5: 06 61 26 78	
B80B9: 90 00 01 0C	B80BB = Beginn der Stand.-variable A
B80BD: 00 00 41 08 A	

### 5.7 Numerische Feldvariable doppelter Genauigkeit

Für die Namen von numerischen Feldvariablen doppelter Genauigkeit können auch wieder einzelne Buchstaben benutzt werden. Wird mit der DIM-Anweisung ein doppelt genaues Feld definiert, so wird die zugehörige Codierung hinter den Feldern einfacher Genauigkeit abgelegt. Die numerischen Standard-Variablen werden zu höheren Adressen verschoben. Anfangs wird für jede Variable der Wert Null eingesetzt. Die Codierung dieser numerischen Felder entspricht der der Zahlen doppelter Genauigkeit, jedoch enthält sie zusätzliche Angaben über den Namen der Variablen und über die Dimensionen des Feldes. Die einzelnen Byte haben folgende Bedeutung:

1. Byte = Länge des Namens, L
2. Byte = Länge des Codierbereiches für das gesamte Feld, low Byte
3. Byte = Länge des Codierbereiches für das gesamte Feld, medium Byte
4. Byte = Länge des Codierbereiches für das gesamte Feld, high Byte  
Gesamtänge = 4 + L + 1 + 2\*D + 12\*(Anzahl der Feldelemente)
5. Byte = ASCII Code des 1. Zeichens des Namens  
.....
- 4+L. Byte = ASCII Code des letzten Zeichens des Namens
- 4+L+1. Byte = Anzahl der Dimensionen des Feldes, D
- 5+L+1. Byte = low Byte des 1. Index
- 5+L+2. Byte = high Byte des 1. Index  
.....
- 5+L+2D-1. Byte = low Byte des letzten Index
- 5+L+2D. Byte = high Byte des letzten Index
- 5+L+2D+1. - 5+L+2D+12. Byte = Codierung des Zahlenwertes des ersten Feldelementes  
.....

Beispiel:

Es wird im RUN-Mode oder beim Ablauf eines Basicprogrammes eingegeben:

```
DIM MM#(1)
MM#(0) = -0.01234567890123456789, MM#(1) = 56.789012345678901234
```

Der RAM-Speicher hat dann im Anschluß an die Werte der beiden Beispiele für numerische Felder einfacher Genauigkeit folgenden Inhalt:

B80BB: 02 21 00 00	B80BB+21=B80DC=nächste Variable
B80BF: 4D 4D 01 01 MM	01=eine Dimension, 0001=Wert des Index
B80C3: 00 09 FE 12	B80C4-B80CF = 12 Plätze für MM#(0)
B80C7: 34 56 78 90	
B80CB: 12 34 56 78	
B80CF: 90 01 01 56	B80D0-B80DB = 12 Plätze für MM#(1)
B80D3: 78 90 12 34	
B80D7: 56 78 90 12	
B80DB: 34 01 0C 00	B80DC = Beginn der Stand.-Variable A

### 5.8 Felder für Zeichenvariable

Bei den Feldern für Zeichenvariable geschieht die Organisation im Speicher ähnlich wie bei den einfachen Zeichenvariablen. Erst wird in einem Bereich, der an den Bereich der numerischen Felder doppelter Genauigkeit anschließt, die Definition der Variablen mit Name, Ablageort des Textes und Textlänge niedergelegt. Die Texte selbst befinden sich nach ihrer Eingabe am Ende des Variablenbereiches.

In der Liste der Namen, Textorte und Textlängen werden die einzelnen Textvariablen folgendermaßen codiert:

1. Byte = Länge des Namens(ohne \$), L
2. Byte = Länge des Codierbereiches für das gesamte Feld  
low Byte
3. Byte = Länge des Codierbereiches für das gesamte Feld  
medium Byte
4. Byte = Länge des Codierbereiches für das gesamte Feld  
high Byte  
Gesamtänge =  $4 + L + 1 + 2*D + 3 + 3*(\text{Anzahl der Feldelemente})$
5. Byte = ASCII Code des 1. Zeichens des Namens  
.....
- 4+L. Byte = ASCII Code des letzten Zeichens des Namens
- 4+L+1. Byte = Anzahl der Dimensionen des Feldes, D
- 5+L+1. Byte = low Byte des 1. Index
- 5+L+2. Byte = high Byte des 1. Index  
.....
- 5+L+2D-1. Byte = low Byte des letzten Index
- 5+L+2D. Byte = high Byte des letzten Index
- 5+L+2D+1. Byte = low Byte des Abstandes des Textes des ersten Feldelementes vom Beginn des Text-Bereiches
- 5+L+2D+2. Byte = medium Byte des Abstandes des Textes des ersten Feldelementes vom Beginn des Text-Bereiches
- 5+L+2D+3. Byte = high Byte des Abstandes des Textes des ersten Feldelementes vom Beginn des Text-Bereiches
- 8+L+2D+1. Byte = low Byte des Abstandes des Textes des ersten Feldelementes vom Beginn der Textes dieses Feldes (stets 00)
- 8+L+2D+2. Byte = high Byte des Abstandes des Textes des ersten Feldelementes vom Beginn der Textes dieses Feldes (stets 00)
- 8+L+2D+3. Byte = Länge des Textes des ersten Feldelementes
- 8+L+2D+4. Byte = low Byte des Abstandes des Textes des zweiten Feldelementes vom Beginn der Textes dieses Feldes (= Länge des ersten Elementes)
- 8+L+2D+5. Byte = high Byte des Abstandes des Textes des zweiten Feldelementes vom Beginn der Textes dieses Feldes
- 8+L+2D+6. Byte = Länge des Textes des zweiten Feldelementes  
.....  
.....  
.....

### Beispiel:

Im RUN-Mode oder über ein Basicprogramm wurde im Anschluß an die vorangehenden Beispiele dieses Kapitels eingegeben:

```
DIM ZEICHENFELD$(2)
ZEICHENFELD$(0) = "FELDVARIALE-0"
ZEICHENFELD$(1) = "VARIABLE-1"
ZEICHENFELD$(2) = "DRITTE-VARIABLE-2"
```

Der RAM-Speicher hat dann im Anschluß an den Speicherinhalt der Beispiele für die Feldvariablen einfacher und doppelter Genauigkeit folgenden Inhalt:

B80DC: 0B 1E 00 00 5A 45 49 43	ZEIC	Definition der Zeichen-
B80E4: 48 45 4E 46 45 4C 44 01	HENFELD	felder
B80EC: 02 00 00 00 00 00 00 0E		
B80F4: 0E 00 0A 18 00 11		
B80FA: 01 0C 00 00 41 00 00 00	A	Erste Standardvariable
B8102: 00 00 00 00		

01	ist die Anzahl der Dimensionen
0002	ist der Wert dieser Dimension
00000	ist der Abstand der Texte von dem Beginn der Textbereiche
0000	ist der Abstand des Textes der ersten Feldvariablen vom Beginn der Texte dieses Feldes
OE	ist die Länge der ersten Feldvariablen
000E	ist der Abstand des Textes der zweiten Feldvariablen vom Beginn der Texte dieses Feldes
0A	ist die Länge der zweiten Feldvariablen
0018	ist der Abstand des Textes der dritten Feldvariablen vom Beginn der Texte dieses Feldes
11	ist die Länge der dritten Feldvariablen

## 5.9 Die Organisation der verschiedenen Typen von Variablen im Speicher. Kopf des Variablenbereiches

Im ursprünglichen Zustand des Computers, d.h. nach ON/RESET und Speicher löschen, hat bei nur eingebootem RAM der Beginn des Speichers folgenden Inhalt:

```

B8000: 10 12 10 00 00 01 00 00
B8008: 00 00 00 00 00 00 00 00
B8010: 00 20 18 00 00 00 00 00
B8018: FB 44 41 54 41 20 20 20      DATA
B8020: 20 42 41 53 20 00 00 00      BAS
B8028: 00 3C 71 00 00 00 84 01
B8030: 00 3C 71 00 00 00 00 00
B8038: 00 00 FF 00 01 00 4C 00
B8040: 00 4C 00 00 4C 00 00 4C
B8048: 00 00 4C 00 00 84 01 00
B8050: 84 01 00 84 01 00 84 01
B8058: 00 84 01 00 00 00 00 00
B8060: 00 00 00 00
B8064: 01 0C 00 00 41 00 00 00      A Beginn Standardvariable
.....
B8190: 01 0C 00 00 5A 00 00 00      Z Letzte Standardvariable
B8198: 00 00 00 00
B819C:

```

Der Kopf des Variablenbereiches beginnt hier bei der Adresse B8018 mit dem Text:

**DATA BAS**

Danach stehen Dekremente, bestehend aus drei Byte (low Byte, medium Byte, high Byte), die, wenn sie auf auf die Variablenbereichs-Anfangsadresse, B8018, aufaddiert werden, Adressen ergeben.

Z.B. tritt dort das Dekrement 00713C oder 000184 auf. Diese ergeben, aufaddiert auf die Variablenbereichs-Anfangsadresse:

B8018 + 00713C = BF154,  
B8018 + 000184 = B819C.

B819C ist die erste Adresse hinter den Standardvariablen, im ursprünglichen Zustand.

BF154 ist die Anfangsadresse des Kopfes des Basicprogrammes im ursprünglichen Zustand.

In diesem ursprünglichen Zustand des Variablenkopfes treten viel Werte mehrfach auf. Erst nach Eingabe der verschiedenen Typen von Variablen sind die unterschiedlichen Bedeutungen zu erkennen. Es

gilt: (Angaben in Klammern bedeuten den Inhalt der Adressen, low, medium, high Byte)

B8018 + (B8029-2B): Kopf des Basicprogramm  
 B8018 + (B802E-30): erster Platz hinter den Variablen,  
                           Beginn des Editierbereiches des Basicprogramm  
 B8018 + (B8031-33): Kopf des Basicprogramm  
 B8018 + (B803E-40): Beginn der numerischen Feldvariablen

einfacher Genauigkeit  
 B8018 + (B8041-43): Beginn der numerischen Feldvariablen  
 doppelter Genauigkeit  
 B8018 + (B8044-46): Beginn der Definition der Text-Feldvariablen  
 B8018 + (B8047-49): Beginn der Definition der Text-Feldvariablen  
 B8018 + (B804A-4C): Beginn der num. Standardvariablen  
 B8018 + (B804D-4F): Beginn der numerischen Variablen einfacher  
 Genauigkeit  
 B8018 + (B8050-52): Beginn der numerischen Variablen doppelter  
 Genauigkeit  
 B8018 + (B8053-55): Beginn der Definition der Zeichenvariablen  
 B8018 + (B8056-58): Beginn der Definition der Zeichenvariablen  
 B8018 + (B8059-5B): Beginn der Texte der verschiedenen  
 Textvariablen

An einem ausführlichen Beispiel wird die Organisation des Variablenfeldes demonstriert, und zwar werden alle Beispiele zu den einzelnen vorangend beschriebenen Variablentypen im Zusammenhang gebracht.

#### Beispiel:

Es wird über die Tastatur eingegeben

ZAHL=1234567  
ANFANGSWERT = -0.0056789

DOPPELT# = -5678.9012345678901234

A\$ = "PC-E500"  
TAG\$ = "Donnerstag"

DIM M(2).  
M(0) = 2345678901, M(1) = -0.003456789012, M(2) = 4567.890123,

```
DIM NUMFELD(1,2)
NUMFELD(0,0)=5006789, NUMFELD(0,1)=5016789,NUMFELD(0,2)=5026789
NUMFELD(1,0)=5106789, NUMFELD(1,1)=5116789,NUMFELD(1,2)=5126789
```

```
DIM MM#(1)
MM#(0) = -0.01234567890123456789, MM#(1) = 56.789012345678901234
```

```
DIM ZEICHENFELD$(2)
ZEICHENFELD$(0) = "FELDVARIABLE-0"
ZEICHENFELD$(1) = "VARIABLE-1"
ZEICHENFELD$(2) = "DRITTE-VARIABLE-2"
```

Nach der Eingabe dieser Beispiele ergeben sich aus dem Kopf des Variablenfeldes folgende Bereichsadressen:

B8018 + 003155 = BB16D: Kopf des Basicprogramm  
B8018 + 0002A4 = B82BC: erster Platz hinter den Variablen.

Beginn des Editierbereichs  
Basicprogramm

B8018 + 003155 = BB16D: Kopf des Basicprogramm

B8018 + 00004C = B8064: Beginn der numerischen Feldvariablen einfacher Genauigkeit

B8018 + 0000A3 = B80BB: Beginn der numerischen Feldvariablen doppelter Genauigkeit  
 B8018 + 0000C4 = B80DC: Beginn der Definition der Text-Feldvariablen  
 B8018 + 0000C4 = B80DC: Beginn der Definition der Text-Feldvariablen  
 B8018 + 0000E2 = B80FA: Beginn der num. Standardvariablen  
 B8018 + 00021A = B8232: Beginn der numerischen Variablen einfacher Genauigkeit  
 B8018 + 00023F = B8257: Beginn der numerischen Variablen doppelter Genauigkeit  
 B8018 + 000256 = B826E: Beginn der Definition der Zeichenvariablen  
 B8018 + 000256 = B826E: Beginn der Definition der Zeichenvariablen  
 B8018 + 00026A = B8282: Beginn der Texte der verschiedenen Textvariablen

Das Variablenfeld selbst hat nach der Eingabe der obigen Beispiele folgenden Inhalt:

B8018: FB 44 41 54 41 20 20 20	DATA	Kopf des Variablenfeldes
B8020: 20 42 41 53 20 00 00 00	BAS	
B8028: 00 55 31 00 00 00 A4 02		
B8030: 00 55 31 00 00 00 00 00		
B8038: 00 00 FF 00 01 00 4C 00		
B8040: 00 A3 00 00 C4 00 00 C4		
B8048: 00 00 E2 00 00 1A 02 00		
B8050: 3F 02 00 56 02 00 56 02		
B8058: 00 6A 02 00 00 00 00 00		
B8060: 00 00 00 00		
B8064: 01 1D 00 00 4D 01 02 00 M		numerische Feldvariable
B806C: 00 09 23 45 67 89 01 08		einfacher Genauigkeit
B8074: FD 34 56 78 90 12 00 03		
B807C: 45 67 89 01 23		
B8081: 07 3A 00 00 4E 55 4D 46	NUMF	
B8089: 45 4C 44 02 01 00 02 00	ELD	
B8091: 00 06 50 06 78 90 00 00		
B8099: 06 50 16 78 90 00 00 06		
B80A1: 50 26 78 90 00 00 06 51		
B80A9: 06 78 90 00 00 06 51 16		
B80B1: 78 90 00 00 06 51 26 78		
B80B9: 90 00		
B80BB: 02 21 00 00 4D 4D 01 01	MM	numerische Feldvariable
B80C3: 00 09 FE 12 34 56 78 90		doppelter Genauigkeit
B80CB: 12 34 56 78 90 01 01 56		
B80D3: 78 90 12 34 56 78 90 12		
B80DB: 34		
B80DC: 0B 1E 00 00 5A 45 49 43	ZEIC	Definition der Zeichen-
B80E4: 48 45 4E 46 45 4C 44 01	HENFELD	felder
B80EC: 02 00 00 00 00 00 00 00		
B80F4: 0E 00 0A 18 00 11		
B80FA: 01 0C 00 00 41 00 00 00	A	Erste Standardvariable
B8102: 00 00 00 00		
.....		
.....		
B8226: 01 0C 00 00 5A 00 00 00	Z	Letzte Standardvariable

B822E: 00 00 00 00		
B8232: 04 0F 00 00 5A 41 48 4C	ZAHL	numerische Variable
B823A: 00 06 12 34 56 70 00		einfacher Genauigkeit
B8241: 0B 16 00 00 41 4E 46 41	ANFA	mit mindestens 2 Zeichen
K8249: 4E 47 53 57 45 52 54 08	NGSWERT	im Namen
B8251: FD 56 78 90 00 00		
B8257: 07 17 00 00 44 4F 50 50	DOPP	
B825F: 45 4C 54 09 03 56 78 90	ELT	numerische Variable
B8267: 12 34 56 78 90 12 34		doppelter Genauigkeit
B826E: 01 09 00 00 41 29 00 00	A	Definition der Zeichen-
B8276: 07		variablen
B8277: 03 0B 00 00 54 41 47 30	TAG	
B827F: 00 00 0A		
B8282: 46 45 4C 44 56 41 52 49	FELDvari	Texte der Textvariablen
B828A: 41 42 4C 45 2D 30	ABLE-0	
B8290: 56 41 52 49 41 42 4C 45	VARIABLE	
B8298: 2D 31	-1	
B829A: 44 52 49 54 54 45 2D 56	DRITTE-V	
B82A2: 41 52 49 41 42 4C 45 2D	ARIABLE-	
B82AA: 32	2	
B82AB: 50 43 2D 45 35 30 30	PC-E500	
B82B2: 44 6F 6E 6E 65 72 73 74	Donnerst	
B82BA: 61 67	ag	
B82BC: FB 54 45 58 54 20 20 20	TEXT	
B82C4: 20 42 41 53 20	BAS	Beginn des Editier-
		bereiches für das
		Basicprogramm

## 6.0 Das Basicprogramm

Für das Basicprogramm stehen beim PC-E500 bei nur eingebautem RAM ursprünglich, d.h. nach ON/RESET und Speicher löschen, 28600 Byte zur Verfügung, was sich aus der Abfrage FRE0 ergibt.

Dieser Bereich wird durch verschiedene andere Eingaben reduziert. Dies sind z.B. Variable, insbesondere Feldvariable, Belegungen der Funktionstasten, AER-Funktionen und ENG-Berechnungen.

Im folgenden wird der Aufbau des Basicprogrammes und seine Organisation im RAM-Speicher im einzelnen beschrieben.

## 6.1 Aufbau des Basicprogrammes

Ein Basicprogramm kann sowohl im Text- als auch im Basic-Modus eingegeben, bzw. von einem in den anderen Modus umgewandelt werden.

Im Textmodus wird es im Speicher Zeichen für Zeichen abgelegt, ohne verändert zu werden. Dies ist sinnvoll, wenn das Programm, z.B. über die serielle Schnittstelle an einen PC übergeben werden soll.

Im Basic-Modus werden die Befehle durch die Token codiert. Nur diese codierte Darstellung des Basicprogrammes kann vom Basic-Interpreter verarbeitet werden.

Wie im Kapitel über die Variablen erläutert, ist im Kopf des Variablenbereiches der Beginn des Basicprogrammes, genauer gesagt, der Beginn des Kopfes des Basicprogrammes, angegeben. Das Basicprogramm selbst beginnt mit einem OD. Jede Zeile beginnt mit der Zeilennummer, die in zwei Byte als Dual- bzw. Hexzahl dargestellt ist. Im nächsten Byte ist die Länge der Zeile angegeben. Danach kommt der Zeileninhalt. Basicbefehle sind mit ihren Token codiert. Diese bestehen aus zwei Byte. Das erste Byte ist stets FE. Die Bedeutung des zweiten Byte ist in der Tokentabelle gegeben. Die unterschiedliche Codierung der Zahlen ist im nächsten Abschnitt erläutert. Abgeschlossen wird eine Zeile mit OD, dem Code für die Return-Taste. Die nächste Zeile beginnt dann wieder mit der Zeilennummer. Am Ende des Basicprogrammes, nach dem letzten OD, steht der Code FF.

Als Beispiel sei das erste Programm aus dem Abschnitt Programmieren der Bedienungsanleitung demonstriert:

```
10: INPUT A
20: PRINT A*A
30: END
```

Im RAM ist dieses Basicprogramm folgendermaßen codiert:

```
OD
00 0A 04 FE 61 41 OD
00 14 06 FE 60 41 2A 41 OD
00 1E 03 FE 5A 0D
FF
```

Es bedeutet:

OD = Beginn des Basicprogramm  
00 0A = erste Zeilennummer, 10  
04 = Länge der Zeile, Abstand bis zum Beginn der nächsten Zeile  
FE 61 = Token für den Basicbefehl INPUT  
41 = A  
OD = Ende der ersten Zeile  
00 14 = zweite Zeilennummer, 20  
06 = Länge der Zeile, Abstand bis zum Beginn der nächsten Zeile  
FE 60 = Token für den Basicbefehl PRINT  
41 = A  
2A = \*

41 = A  
0D = Ende der zweiten Zeile  
00 1E = dritte Zeilennummer, 30  
03 = Länge der Zeile, Abstand bis zum Beginn der nächsten Zeile  
FE 5A = Token für den Basicbefehl END  
0D = Ende der dritten Zeile  
FF = Ende des Basicprogramm

## 6.2 Die Codierung von Zahlen im Basicprogramm

Die Codierung von Zahlen im Basicprogramm weicht etwas von dem bei den Variablen beschriebenen Schema ab.

So wird z.B. codiert:

A=1234567 durch: 41 3D 1D 00 06 12 34 56 70 00  
A =

A=-1234567 durch: 41 3D 2D 1D 00 06 12 34 56 70 00  
A = -

A=123456789012345# durch:  
41 3D 1D 01 0E 12 34 56 78 90 12 34 50 00 00  
A =

A=-123456789012345# durch:  
41 3D 2D 1D 01 0E 12 34 56 78 90 12 34 50 00 00  
A = -

Es wird also vor der Codierung der Zahlen das Kennbyte 1D in das Basicprogramm eingetragen, außerdem wird das Minus-Zeichen explizit im Text geschrieben und nicht durch 08 bzw. 09 im ersten Byte der Zahl gekennzeichnet. Die Eintragung von A = Zahl im Basicprogramm bedeutet allerdings nicht, daß auch der Variablen A dieser Wert zugewiesen wird. Dies geschieht erst beim Ablauf des Basicprogrammes.

Diese Codierung von Zahlen im Basicprogramm gilt nicht in jedem Fall. Zahlenangaben nach GOTO, GOSUB werden bei der Eingabe in Hexzahlen umgewandelt und folgendermaßen codiert:

GOTO 10       durch: FE 2B 1F 00 OA                   FE2B = Token von GOTO  
GOTO 2000      durch: FE 2B 1F 07 D0                   &07D0 = 2000

Hierbei tritt als Vorcode vor der Zahl das Byte 1F auf. Die Dezimalzahl wird in eine vierstellige Hexzahl umgewandelt, in die Form, in der auch die Zeilenzahlen im Basicprogramm codiert sind.

Schließlich sei noch darauf hingewiesen, daß Eintragungen von Hexzahlen ins Basicprogramm explizit übernommen und nicht codiert werden:

POKE &B8010,&0 wird: FE 32 26 42 38 30 31 30 2C 26 30  
POKE    & B 8 0 1 0 , & 0

Durch Eingabe von Hexzahlen lässt sich also Speicherplatz sparen, denn die Hexzahl &0 benötigt nur zwei Byte, während die Dezimalzahl 0 entsprechend dem obigen Schema mit 8 Byte codiert wird.

### 6.3 Die Tokentabelle

Die Tabelle der Token (Codierung der Basicbefehle im Programm) findet sich im ROM ab Adresse &F5A91. Hier ist jeweils ein Byte angegeben. Im Basicprogramm haben die Token noch den Vorcode FE. In den folgenden Tabellen sind die Token sowohl in alphabetischer Reihenfolge als auch in einer Hextabelle angegeben.

#### Tokentabelle, alphabetisch

ABS	99,	ACS	9E,	AER	BE,	AHC	8E
AHS	8D,	AHT	8F,	AND	A1,	APPEND	72
ARUN	74,	AS	73,	ASN	9D,	ATN	9F
AUTO	1A,	AUTOGOTO	75,	BASIC	36,	BDATA\$	0C
BEEP	29,	BTEXT\$	0B,	CALL	31,	CHAIN	67
CHR\$	F0,	CIRCLE	6F,	CLEAR	2E,	CLOAD	16
CLOSE	22,	CLS	50,	COLOR	44,	CONSOLE	24
CONT	12,	COPY	3D,	COS	96,	CROTATE	6E
CSAVE	20,	CSIZE	43,	CUB	BF,	CUR	89
DATA	5E,	DECI	84,	DEFDBL	46,	DEFSGN	47
DEG	9B,	DEGREE	26,	DELETE	1B,	DIM	30
DMS	9C,	DSKF	B1,	ELSE	76,	END	5A
EOF	B0,	ERASE	3A,	ERL	C1,	ERN	C0
ERROR	78,	EVAL	A7,	EXP	93,	FACT	90
FILES	1C,	FOR	57,	FRE	AF,	GCURSOR	68
GLCURSOR	6C,	GOSUB	62,	GOTO	2B,	GPRINT	33
GRAD	28,	GRAPH	41,	HCS	8B,	HEX	85
HEXS	F2,	HSN	8A,	HTN	8C,	IF	56
INIT	1D,	INKEY\$	E9,	INPUT	61,	INT	98
KEY	79,	KILL	3C,	LEN	D2,	LEFT\$	EB
LET	58,	LF	42,	LFILES	3B,	LINE	69
LIST	14,	LLINE	6A,	LLIST	15,	LN	91
LOAD	18,	LOC	B3,	LOCATE	51,	LOF	B2
LOG	92,	LPRINT	64,	LTEXT	40,	MDF	80
MEM\$	0D,	MERGE	17,	MIDS	EA,	NAME	3E
NCR	B6,	NEW	11,	NEXT	5B,	NOT	A3
NPR	B7,	ON	55,	OPEN	21,	OPEN\$	E8
OR	A2,	OUTPUT	71,	PAINT	70,	PASS	13
PAUSE	5F,	PEEK	A4,	PI	AE,	POINT	AD
POKE	32,	POL	82,	PRESET	35,	PRINT	60
PSET	34,	RADIAN	27,	RANDOMIZE	25,	RPC	87
READ	5D,	REC	81,	REM	59,	RENUM	19
RESTORE	66,	RESUME	77,	RETURN	65,	RIGHT\$	EC
RLINE	6B,	RND	A0,	ROT	83,	RUN	10
SAVE	23,	SET	3F,	SGN	9A,	SIN	95
SORGN	6D,	SQR	94,	SQU	88,	STEP	53
STOP	5C,	STR\$	F1,	TAN	97,	TEN	86
TEXT	37,	THEN	54,	TO	52,	TROFF	2D
TRON	2C,	USING	2F,	VAL	D1,	WAIT	2A
XOR							

### 6.4 Tokentabelle, low

	&00	&10	&20	&30	&40	&50	&60	&70
&0	RUN	CSAVE	DIM	LTEXT	CLS	PRINT	PAINT	
&1	NEW	OPEN	CALL	GRAPH	LOCATE	INPUT	OUTPUT	
&2	CONT	CLOSE	POKE	LF	TO	GOSUB	APPEND	
&3	PASS	SAVE	GPRINT	CSIZE	STEP		AS	
&4	LIST	CONSOLE	PSET	COLOR	THEN	LPRINT	ARUN	
&5	LLIST	RANDOMIZE	PRESET		ON	RETURN	AUTO-GOTO	
&6	CLOAD	DEGREE	BASIC	DEFDBL	IF	RESTORE	ELSE	
&7	MERGE	RADIAN	TEXT	DEFSGN	FOR	CHAIN	RESUME	
&8	LOAD	GRAD			LET	GCURSOR	ERROR	
&9	RENUM	BEEP			REM	LINE	KEY	
&A	AUTO	WAIT	ERASE		END	LLINE		
&B	BTEXT\$	DELETE	GOTO	LFILES		NEXT	RLINE	
&C	BDATA\$	FILES	TRON	KILL		STOP	GL-CURSOR	
&D	MEM\$	INIT	TROFF	COPY		READ	SORGN	
&E			CLEAR	NAME		DATA	CROTATE TATE	
&F			USING	SET		PAUSE	CIRCLE	

### 6.5 Tokentabelle, high

&80	&90	&A0	&B0	&C0	&D0	&E0	&FO
&0 MDF	FACT	RND	EOF	ERN	ASC		CHR\$
&1 REC	LN	AND	DSKF	ERL	VAL		STR\$ POKE
&2 POL	LOG	OR	LOF		LEN		PEEK CALL
&3 ROT	EXP	NOT	LOC				EVAL
&4 DECI	SQR	PEEK					NAME
&5 HEX	SIN	XOR					
&6 TEN	COS		NCR				
&7 RCP	TAN	EVAL	NPR				
&8 SQU	INT					OPEN\$	
&9 CUR	ABS					INKEY\$	
&A HSN	SGN					MID\$	
&B HCS	DEG					LEFT\$	
&C HTN	DMS					RIGHT\$	
&D AHS	ASN	POINT					
&E AHC	ACS	PI	AER				
&F AHT	ATN	FRE	CUB				

### 6.6 Unbekannte Basicbefehle

Unter unbekannten Basicbefehlen sollen solche verstanden sein, deren Token in der Tokenliste im ROM des PC-E500 vorhanden sind, die aber in der Bedienungsanleitung nicht beschrieben sind. Dies sind:

diese drei Basicbefehle werden im Kapitel über die Maschinensprache beschrieben.

wandelt eine als Zeichenvariable eingegebene Zahl in ihren numerischen Wert um.  
 dient zur Änderung eines Dateinamens. Z.B.  
 NAME"E:PROGRAMM.BAS"AS"PRGR.BAS"  
 ändert den Namen der Datei PROGRAMM.BAS in der Ramdisk E:  
 in PRGR.BAS

### 6.7 Die Organisation des Basicprogrammes im Speicher

Vor dem Basicprogramm gibt es einen Kopf, aus dem das Ende des Basicprogrammes und der Anfang des nächstfolgenden Datenbereiches, der Funktionstasten, zu entnehmen ist. Außerdem enthält er das Passwort, falls eins für das Basicprogramm im Arbeitsspeicher definiert ist.

#### Ende des Basicprogrammes:

Anfangsadresse + Inhalt von (Anfangsadresse des Kopfes + 29/30/31) = Endadresse des Basicprogramms, FF

Ist die Adresse des Kopfes des Basicprogramm zu BEDA1 (= B8018 + 006D89) zu entnehmen.

Dem Kopf des Basicprogrammes:  
 BEDA1: FB 54 45 58 54 20 20 20 TEXT  
 BEDA9: 20 42 41 53 22 00 00 00 BAS  
 BEDB1: 00 E9 03 00 00 00 E9 03  
 BEDB9: 00 E9 03 00 00 00 00 00  
 BEDC1: 00 00 FF 00 00 00 34 00  
 BEDC9: 00 E8 03 00 44 55 4D 50 DUMP  
 BEDD1: 2E 42 41 53 .BAS

sind folgende Informationen zu entnehmen:

#### Beginn des Bereiches der Funktionstasten:

Anfangsadresse + Inhalt von (Anfangsadresse des Kopfes + 11/12/13) = Anfangsadresse des Kopfes für die Funktionstasten, FF

Das Basicprogramm im Arbeitsspeicher ist durch das Passwort DUMP.BAS geschützt. Dies ergibt sich aus 22 auf BEDAD statt 20 ohne Passwortschutz. Das Passwort DUMP.BAS steht auf BEDCD bis BEDD4, d.h. unmittelbar vor dem Beginn des Basicprogrammes.

#### Passwort:

Wenn ein Passwort definiert ist, steht hinter BAS nicht 20 sondern 22 und das Passwort steht auf den letzten acht Byte des Kopfes, also unmittelbar vor dem ersten OD des Basicprogrammes.

Die Adresse des Endes des Basicprogrammes (FF) ergibt sich zu:  
 BEDA1 + 0003E8 = BF189  
 (BEDCA/B/C)

Bei nur eingebautem RAM beginnt im ursprünglichen Zustand, d.h. nach ON/RESET und Speicher löschen, in dem noch kein Basicprogramm vorhanden ist, der Kopf für das Basicprogramm bei der Adresse BF154 und hat folgenden Inhalt:

BF154: FB 54 45 58 54 20 20 20 TEXT  
 BF15C: 20 42 41 53 20 00 00 00 BAS  
 BF164: 00 36 00 00 00 00 36 00  
 BF16C: 00 36 00 00 00 00 00 00  
 BF174: 00 00 FF 00 00 00 34 00  
 BF17C: 00 35 00 00 00 00 00 00  
 BF184: 00 00 00 00 00 FF

BF184: FB 46 F UNCKEY

BF154 = Beginn des Kopfes des Basicprogramm selbst befindet sich zwischen BEDD5 und BF189:  
 des Basicprogramm  
 BEDD5: 0D 07 DO 1F 2A 44 3A 50 7DO = 2000 dezimal,  
 BEDDD: 3D 1D 00 00 00 00 00 00 erste Zeilennummer  
 .....  
 BF17E: 5A 0D 08 AC 06 FE 2B 00  
 BF186: 84 02 0D FF FB 46 55 4E FUN

BF188 = OD vor einem Basicprogramm  
 BF189 = FF hinter einer Basicprogramm  
 BF18A = FB, Beginn des Kopfes für Funktionstaste

Wird nun ein Basicprogramm eingegeben, so wird es zwischen dem 0 und dem FF eingefügt. Das FF auf der Adresse BF189 bleibt dabei auf seinem Platz, während das Basicprogramm davor aufgebaut wird. Aus dem Kopf des Variablenbereiches ergab sich, daß ursprünglich für den Aufbau des Basicprogramm der Speicher ab der Adresse B819C zu Verfügung steht. Dieser Bereich von B819C bis BF153 umfaßt 28600(dez) = 6FB8 Byte, was dem Wert von FRE0 im ursprünglichen Zustand bei nur eingebautem RAM entspricht.

Zur Demonstration wurde das in einem früheren Kapitel angegebene Dump-Programm eingegeben und mit dem Passwort "DUMP.BAS" geschützt:

#### Aus dem Variablenkopf:

B8018: FB 44 41 54 41 20 20 20 DATA  
 B8020: 20 42 41 53 20 00 00 00 BAS  
 B8028: 00 89 6D 00 00 00 84 01

## 7.0 RAM-Disketten, RAM-Karten

Bei nur eingebautem RAM mit 32 KByte kann nur die RAM-Diskette mit dem Basic-Befehl INIT initialisiert werden.

Stellt eine RAM-Karte (8-, 16-, 32- oder 64-KByte) zur Verfügung, so gibt es mehrere Möglichkeiten ihrer Verwendung. Diese verschiedenen Möglichkeiten werden mit dem Basic-Befehl MEM\$ festgelegt.

MEM\$="S1"

Die RAM-Karte wird für die RAM-Diskette F verwandt.

MEM\$="S2"

Die RAM-Karte wird für die Variablen und das Basic-Programm verwandt. Für die RAM-Diskette E und die sonstigen Funktionen des PC-E500 wird das eingebaute RAM benutzt.

MEM\$="B"

Der Speicher der RAM-Karte wird dem Arbeitsspeicher des eingebauten RAM zugeschlagen, d.h. der Umfang des Arbeitsspeichers wird um die KByte der RAM-Karte erhöht. Die RAM-Diskette F steht in diesem Falle nicht zur Verfügung.

Zur Verwendung dieser verschiedenen Möglichkeiten studiere man ausführlich den Abschnitt RAM-Karten der Bedienungsanleitung.

Eine weitere RAM-Diskette G befindet sich im ROM des PC-E500 ab C0018. Diese enthält Dateien und Basicprogramme, die vom PC-E500 bei der Technikersoftware benutzt werden. D.h. bei vielen Funktionen arbeitet der PC-E500 nicht mit Maschinenprogrammen sondern mit bei Bedarf in einen bestimmten Bereich des Arbeitsspeichers geladenen Basicprogrammen.

Im folgenden ist die Organisation der auf den RAM-Disketten gespeicherten Dateien/Basicprogramme beschrieben. Zur Erkundung dieser Organisation stand nur eine 16K RAM-Karte zur Verfügung. So sind die Angaben in diesem Kapitel für höhere Kapazitäten (über 42K) nicht durch Tests überprüft.

## 7.1 RAM-Disk F

Die RAM-Disk F steht bei Vorhandensein einer RAM-Karte und der Speicherorganisation MEM\$="S1" zur Verfügung. Sie befindet sich im Speicherbereich für die RAM-Karten, d.h. bei

8K RAM-Karte von 40000 bis 41FFF

16K RAM-Karte von 40000 bis 43FFF

32K RAM-Karte von 40000 bis 47FFF

64K RAM-Karte von 40000 bis 4FFFF

Die RAM-Disk F muß mit INIT"F:XK" initialisiert werden. X kann bis zu eins weniger betragen als der Kapazität der RAM-Karte entspricht.

### Kopf der RAM-Disk

Nach INIT"F:15K" (maximale Möglichkeit für 16K RAM-Karte) hat der Kopf der RAM-Disk F folgende Gestalt:

40018:	FB	52	41	4D	46	49	4C	45	RAMFILE
40020:	20	20	20	20	20	00	00	00	
40028:	00	00	3C	00	00	00	00	3C	
40030:	00	00	3C	00	00	00	00	00	
40038:	00	00	FF	00	07	01	3C	00	
40040:	00	00	F4	00	01	07	03	00	
40048:	01	01	00	01	20	00	06	00	
40050:	37	00	01	02	00				

Die Werte auf den einzelnen Adressen haben folgende Bedeutung:

Adresse	Wert
4002B/A/9	003C00 Speicherumfang der RAM-Disk F (hier 15 K)
40030/2F/E	"
40033/2/1	"
	Kapazität von F: Wert auf 4002A/2F/32/3E
	2K 08
	3K 0C
	4K 10
	5K 14
	6K 18
	7K 1C
	8K 20
	9K 24
	10K dezimal 28
	11K dezimal 2C
	12K dezimal 30
	...
	wächst für jedes KByte um 04
4003A	FF
4003C	07
4003F/E/D	003C01 Speicherumfang der RAM-Disk F + 1
40042	F4 = Wert am Beginn der FAT, Adresse 40118
	Kapazität von F: Wert auf 40042 und 40118
	2K F0
	3-4K F1
	5-8K F2

		9-12K dezimal	F3
		13-16K dezimal	F4
		17-20K dezimal	F5
		21-24K dezimal	F6
		25-28K dezimal	F7
		29K... dezimal	F8
40044	01		
40045	07		
40046	03		
40048	01		
40049	01		
4004B	01		
4004C	20	= Anzahl der möglichen Dateien, Anzahl der möglichen Eintragungen im Inhaltsverzeichnis Kapazität von F: Anzahl der möglichen Dateien	
		2K	05
		3-4K	08
		5-8K	10 = 16 dezimal
		9-12K dezimal	18 = 24 dezimal
		13-16K dezimal	20 = 32 dezimal
		17-20K dezimal	28 = 40 dezimal
		21-24K dezimal	30 = 48 dezimal
		25-28K dezimal	38 = 56 dezimal
		29K... dezimal	40 = 64 dezimal
4004E/D	0600	= Speicherumfang des Directory + 200 Kapazität von F: Wert von 4004E/D Directory	
		2-4K	0300
		5-8K	0400
		9-12K dezimal	0500
		13-16K dezimal	0600
		17-20K dezimal	0700
		21-24K dezimal	0800
		25-28K dezimal	0900
		29K... dezimal	0A00
40050/4F	3700	= 100 + DSKF"F:", falls F: leer Kapazität von F: Wert von 40050/4F DSKF"F:"	
		2K	0600
		3K	0A00
		4K	0E00
		5K	1100
		6K	1500
		7K	1900
		8K	1D00
		9K	2000
		10K dezimal	2400
		11K dezimal	2800
		12K dezimal	2C00
		13K dezimal	2F00
		14K dezimal	3300
		15K dezimal	3700
		16K dezimal	3B00
		17K dezimal	3E00
		18K dezimal	4200
		19K dezimal	4600
		20K dezimal	4A00
		21K dezimal	4D00
		22K dezimal	5100

23K dezimal	5500	5400
24K dezimal	5900	5800
25K dezimal	5C00	5B00
26K dezimal	6000	5F00
27K dezimal	6400	6300
28K dezimal	6800	6700
29K dezimal	6B00	6A00
30K dezimal	6F00	6E00
31K dezimal	7300	7200
32K dezimal	7700	7600
33K dezimal	7B00	7A00
34K dezimal	7F00	7E00
35K dezimal	8300	8200

0052  
0053

01  
02

#### FAT, File Allocation Table

Die File Allocation Table macht Aussagen über die Anordnung der Daten der Dateien in der RAM-Disk, sie beginnt bei 40118 mit dem Wert von 40042, und zwar:

Kapazität von F:	Wert auf 40118/9/A
2K	F0 00 00
3-4K	F1 00 00
5-8K	F2 00 00
9-12K dezimal	F3 00 00
13-16K dezimal	F4 00 00
17-20K dezimal	F5 00 00
21-24K dezimal	F6 00 00
25-28K dezimal	F7 00 00
29K... dezimal	F8 00 00

Dieser Anfangswert der FAT bestimmt die Anfangsadresse für die Ablage der Texte der Dateien. Genauere Angaben hierzu finden sich im Abschnitt über Texte der Dateien.

Die Ablage des Textes geschieht in Blöcken von 100 = 256 dezimal. Die Anordnung, bzw. Reihenfolge, dieser Blöcke spiegelt sich in der FAT wieder.

S werden stets zwei Blockangaben in einer speziellen Form zu drei Byte zusammengefaßt. Sei die Angabe für einen Block 123 und die für den nächsten Block 456, so wird dies in den drei zugehörigen Byte zu 23 61 45.

Auf 40118 - 4011A steht der Vorspann zur FAT.

Auf 4011B - 4011D stehen die Angaben zu Block 002 und 003.

Auf 4011E - 40120 stehen die Angaben zu Block 004 und 005.

Die erste Blocknummer für die Ablage der Daten einer Datei ist im Directory angegeben. Auf dem zugehörigen Platz in der FAT steht die Nummer des Blockes, der als nächstes für die Ablage der Daten der Datei benutzt wird, usw. Wenn kein weiterer Block für die Da-

ten nötig ist, erscheint bei dem letzten Block die Angabe FF0. Bei den Blöcken, die nicht belegt sind, enthält der zugehörige Platz die Angabe 000.

An einem ausführlichen Beispiel seien diese Verhältnisse demonstriert.

Beispiel:

INIT"F:15K"  
COPY"G:PRIME.111"TO"F:" Länge von PRIME.111 = 990 = 3DE  
COPY"G:NORMAL.411"TO"F:" Länge von NORMAL.411 = 732 = 2DC  
KILL"F:PRIME.111"  
COPY"G:DIFF.132"TO"F:" Länge von DIFF.132 = 1346 = 542

- 1) INIT"F:15K" initialisiert F: mit 15 KByte.  
DSKF"F:" gibt mit 13824 = 3600 den verfügbaren Speicher an.
- 2) COPY"G:PRIME.111"TO"F:" kopiert PRIME.111 von G: nach F:.  
DSKF"F:" gibt mit 12800 = 3200 die Belegung von 4 Blöcken an,  
denn es ist 3600 - 3200 = 400.

Directory:

40218: 50 52 49 4D 45 20 20 20 PRIME  
40220: 31 31 31 20 00 00 00 00 111  
40228: 00 00 00 00 00 00 00 00  
40230: 00 00 02 00 DE 03 00 00 0002 = Beginn der Daten b  
Block 2  
0003DE = Länge der Datei,  
Block 2, 3, 4 und

FAT:

40118: F4 00 00 03 40 00 05 00  
40120: FF 00 00 00 00 00 00 00  
PRIME.111 beginnt auf Bl.  
4011B-1D = Block (2,3): 003,004 = nach Block 2 weiter bei 3  
nach Block 3 weiter bei 4  
4011E-20 = Block (4,5): 005,FF0 = nach Block 4 weiter bei 5  
nach Block 5 nicht weiter,  
Dateiende von PRIME.111

Daten von PRIME.111 auf:

40618 - 40717 = Block 2  
40718 - 40817 = Block 3  
40818 - 40917 = Block 4  
40918 - 409F5 = Block 5, zum Teil

- 3) COPY"G:NORMAL.411"TO"F:" kopiert NORMAL.411 von G: nach F:.  
DSKF"F:" gibt mit 12032 = 2F00 die Belegung von 3 Blöcken an,  
denn es ist 3200 - 2F00 = 300.

Directory:

40218: 50 52 49 4D 45 20 20 20 PRIME  
40220: 31 31 31 20 00 00 00 00 111  
40228: 00 00 00 00 00 00 00 00  
40230: 00 00 02 00 DE 03 00 00  
40238: 4E 4F 52 4D 41 4C 20 20 NORMAL  
40240: 34 31 31 20 00 00 00 00 411  
40248: 00 00 00 00 00 00 00 00  
40250: 00 00 06 00 DC 02 00 00 0006 = Beginn der Daten b

Block 6  
0002DC = Länge der Datei,  
Block 6, 7 und 8

FAT:

40118: F4 00 00 03 40 00 05 00  
40120: FF 07 80 00 F0 OF 00 00  
PRIME.111 beginnt auf Bl. 2  
4011B-1D = Block (2,3): 003,004 = nach Block 2 weiter bei 3  
nach Block 3 weiter bei 4  
4011E-20 = Block (4,5): 005,FF0 = nach Block 4 weiter bei 5  
nach Block 5 nicht weiter,  
Dateiende von PRIME.111.  
NORMAL.411 beginnt auf Bl. 6  
40121-23 = Block (6,7): 007,008 = nach Block 6 weiter bei 7  
nach Block 7 weiter bei 8  
40124-26 = Block (8,9): FF0,000 = nach Block 8 nicht weiter,  
Dateiende von NORMAL.411

Daten von PRIME.111 auf:

40618 - 40717 = Block 2  
40718 - 40817 = Block 3  
40818 - 40917 = Block 4  
40918 - 409F5 = Block 5, zum Teil  
Daten von NORMAL.411 auf:  
40A18 - 40B17 = Block 6  
40B18 - 40C17 = Block 7  
40C18 - 40CF3 = Block 8, zum Teil

KILL"F:PRIME.111"

DSKF"F:" gibt mit 12032 = 3300 das Löschen von 4 Blöcken an,  
denn es ist 3300 - 2F00 = 400.

Directory:

40218: E5 52 49 4D 45 20 20 20 RIME "P" ist mit E5 über-  
schrieben  
40220: 31 31 31 20 00 00 00 00 111  
40228: 00 00 00 00 00 00 00 00  
40230: 00 00 02 00 DE 03 00 00  
40238: 4E 4F 52 4D 41 4C 20 20 NORMAL  
40240: 34 31 31 20 00 00 00 00 411  
40248: 00 00 00 00 00 00 00 00  
40250: 00 00 06 00 DC 02 00 00

FAT:

40118: F4 00 00 00 00 00 00 00  
40120: 00 07 80 00 F0 OF 00 00

NORMAL.411 beginnt auf Bl. 6  
40121-23 = Block (6,7): 007,008 = nach Block 6 weiter bei 7  
nach Block 7 weiter bei 8  
40124-26 = Block (8,9): FF0,000 = nach Block 8 nicht weiter,  
Dateiende von NORMAL.411

Daten von PRIME.111 auf:

40618 - 40717 = Block 2  
40718 - 40817 = Block 3  
40818 - 40917 = Block 4

40918 - 409F5 = Block 5, zum Teil  
Daten von NORMAL.111 auf:  
40A18 - 40B17 = Block 6  
40B18 - 40C17 = Block 7  
40C18 - 40CF3 = Block 8, zum Teil

- 5) COPY"G:DIFF.132"TO"F:" kopiert DIFF.132 von G: nach F:.  
DSKF"F:" gibt mit 11520 = 2D00 die Belegung von 6 Blöcken an  
denn es ist 3300 - 2D00 = 600.

Directory:

40218: 44 49 46 46 20 20 20 20 DIFF  
40220: 31 33 32 20 00 00 00 00 132  
40228: 00 00 00 00 00 00 00 00  
40230: 00 00 02 00 42 05 00 00  
40238: 4E 4F 52 4D 41 4C 20 20 NORMAL  
40240: 34 31 31 20 00 00 00 00 411  
40248: 00 00 00 00 00 00 00 00  
40250: 00 00 06 00 DC 02 00 00

FAT:

40118: F4 00 00 03 40 00 05 90  
40120: 00 07 80 00 F0 AF 00 F0  
40128: 0F 00 00 00 00 00 00 00

DIFF.132 beginnt auf Bl. 2  
4011B-1D = Block (2,3): 003,004 = nach Block 2 weiter bei 3  
nach Block 3 weiter bei 4  
4011E-20 = Block (4,5): 005,009 = nach Block 4 weiter bei 5  
nach Block 5 weiter bei 9  
NORMAL.411 beginnt auf Bl. 6  
40121-23 = Block (6,7): 007,008 = nach Block 6 weiter bei 7  
nach Block 7 weiter bei 8  
40124-26 = Block (8,9): FF0,00A = nach Block 8 nicht weiter  
Dateiende von NORMAL.411  
nach Block 9 weiter bei A  
40127-29 = Block (A,B): FF0,000 = nach Block A nicht weiter  
Dateiende von DIFF.132

Daten von DIFF.132 auf:

40618 - 40717 = Block 2  
40718 - 40817 = Block 3  
40818 - 40917 = Block 4  
40918 - 40A17 = Block 5  
40D18 - 40E17 = Block 9  
40E18 - 40E59 = Block A, zum Teil  
Daten von NORMAL.111 auf:  
40A18 - 40B17 = Block 6  
40B18 - 40C17 = Block 7  
40C18 - 40CF3 = Block 8, zum Teil

Das Inhaltsverzeichnis, Directory

Das Inhaltsverzeichnis beginnt auf 40218. Für jede Eintragung stehen &20 Byte zur Verfügung. Die Länge des Speicherbereiches für das Inhaltsverzeichnis ist gleich &20 \* Zahl der Dateien (Wert von 4004C).

Der Speicherbereich für das Inhaltsverzeichnis umfaßt je nach Kapazität der RAM-Disk 100, 200, ..., 800 Byte, und zwar:

Kapazität von F:	Bereich und Umfang des Directory	Anzahl der möglichen Dateien
1K	40218 - 40317 = 100 Byte	05
2K	40218 - 40317 = 100 Byte	08
4K	40218 - 40417 = 200 Byte	10 = 16 dezimal
8K	40218 - 40517 = 300 Byte	18 = 24 dezimal
16K	40218 - 40617 = 400 Byte	20 = 32 dezimal
32K	40218 - 40717 = 500 Byte	28 = 40 dezimal
64K	40218 - 40817 = 600 Byte	30 = 48 dezimal
128K	40218 - 40917 = 700 Byte	38 = 56 dezimal
256K	40218 - 40A17 = 800 Byte	40 = 64 dezimal
512K... dezimal		

Die Eintragung für die erste Datei hat folgende Anordnung:

40218 bis 1F = Name der Datei, hat der Name weniger als 8 Zeichen, so werden die restlichen Plätze mit 20 aufgefüllt

40220 bis 22 = Extension, hat die Extension weniger als 3 Zeichen, so werden die restlichen Plätze mit 20 aufgefüllt

40223 = &20, falls Datei nicht geschützt  
&21, falls Datei geschützt, mit SET

40224 bis 31 = 00

40233/2 = Nummer des Blockes bei dem die Ablage der Daten der Datei beginnt

40236/5/4 = Länge der Daten der Datei, wie auch bei FILES angegeben

40237 = 00

40238 .... Eintragung für die 2. Datei

Beispiel:

Nach INIT"F:15K" wurde die Datei PRIME.111 von G: nach F:

kopiert. Im Directory ergibt sich die Eintragung:

40218: 50 52 49 4D 45 20 20 20 PRIME

40220: 31 31 31 20 00 00 00 00 111

40228: 00 00 00 00 00 00 00 00 00

40230: 00 00 02 00 DE 03 00 00 00002 = Beginn der Daten bei Block 2

0003DE = Länge der Datei

Daten der Dateien

Die Daten der Dateien werden in Blöcken von 100 = 256(dezimal) Byte abgespeichert. Im Directory wird der Anfangsblock für die Ablage der Daten angegeben. Der Zählbeginn für diese Blockangabe ist in der folgenden Tabelle gegeben:

Kapazität der RAM-Disk	Wert am Beginn der FAT	Zählbeginn für die Blockangabe im Directory	Datenbeginn, Block 2
1K	F0 00 00	40118	40318
2K	F1 00 00	40118	40318
4K	F2 00 00	40218	40418

9-12K dezimal	F3 00 00	40318
13-16K dezimal	F4 00 00	40418
17-20K dezimal	F5 00 00	40518
21-24K dezimal	F6 00 00	40615
25-28K dezimal	F7 00 00	40718
29K... dezimal	F8 00 00	40818

### Beispiel:

INIT" F:15K"

Handelt es sich um ein Basicprogramm, das aus dem Arbeitsspeicher bei der Ablage im Zwischencode hat das Basicprogramm einen Vormit

SAVE "F:Dateiname.Extension", A  
nach F: gegeben wurde, so wird es als Text (ohne Token abgespeichert.

Basic programs

**SAVE "F:END.BAS",A** Übergabe nach F: in ASCII-Format

40118: F4 00 00 F0 0F 00 00 00 FAT

Datenbeginn auf 40418 + 02 Blöcke = 40618

40618: 32 30 20 45 4E 44 0D 0A 20 END Daten der Datei  
40620: 1A 00 00 00 00 00 00 00 00 40618 + 09 = 40621 = erster freier Platz

Handelt es sich um ein Basicprogramm, das aus dem Arbeitsspeicher mit  
SAVE "F:Dateiname.Extension"  
nach F: gegeben wurde, so wird es im Zwischencode (mit Token) abgespeichert. Hier wird vor dem Basicprogramm ein Kopf abgelegt, der mit FF 00 00 00 34 00 beginnt. Danach kommt auf den nächsten beiden Byte ein Wert, der der um &21 erhöhten Länge der Datei entspricht. Danach dann wieder 10 (dezimal) mal 00.

20:END Basicprogramm

**SAVE "F:END.BAS"** Übergabe nach F  
im Zwischencode

40118: F4 00 00 F0 0F 00 00 00 FATA

40218: 45 4E 44 20 20 20 20 20 20 END Eintragung im Directory  
40220: 42 41 53 20 00 00 00 00 00 BAS  
40228: 00 00 00 00 00 00 00 00 00  
40230: 00 00 02 00 1A 00 00 00 00 0002 = Blockangabe für

Datenbeginn  
00001A = Länge der Datei

40818 Datenbeginn auf 40418 + 02 Blöcke = 40618

40918: 0618: FF 00 00 00 34 00 00 00 31  
40A18: 0620: 00 00 00 00 00 00 00 00 00  
0628: 00 OD 00 14 03 FE 5A  
0630: OD FF

## Daten der Datei

40618 + 1A = 40632 =  
erster freier Platz

Bei der Ablage im Zwischencode hat das Basicprogramm einen Vor-  
spann von 12 Byte (= 18 dezimal), der mit FF beginnt.

KILL, SAVR

wird eine Datei mit KILL gelöscht, so wird die FAT und das Directory geändert.

In der FAT wird die zugehörige Serie von Zahlen durch Nullen ersetzt. Im Directory wird das erste Zeichen des Namens durch E5 ersetzt. Die sonstigen Daten im Directory und im Datenbereich bleiben erhalten.

Obwohl also die Daten der gelöschten Datei erhalten bleiben, genügt es nicht auf dem ersten Zeichen des Directory statt des E5 das richtige Zeichen einzupoken. Es müssen auch die Angaben in der FAT wieder hergestellt werden. Dies ist möglich, da die Eintragung in Directory bis auf das erste Zeichen des Namens erhalten bleibt. Mit Hilfe der Angabe des ersten Blockes für die Datenablage und die Länge der Datei lässt sich die FAT durch Einpoken rekonstruieren.

Wird nach KILL mit SAVE oder COPY wieder eine Datei in die RAM-Disk gegeben, so werden die Daten der neuen auf die Plätze der früheren Daten der gelöschten Dateien gegeben. Nach KILL kann also der Text einer neu auf die RAM-Disk gegebenen Datei in nicht aufeinander folgenden Blöcken gespeichert sein.

## 7.2 RAM-Disk E

Für die RAM-Disk E gilt im Prinzip das gleiche wie für die RAM-Disk F. Während jedoch die RAM-Disk F stets bei der gleichen Adresse, 40018, beginnt, hat die RAM-Disk E, je nach vorhandener RAM-Karte und eingestellter Speicherkonfiguration unterschiedliche Anfangsadressen:

Vorhandenes RAM mit Speicherkonfiguration, MEM\$

Nur eingebautes RAM:

Eingebautes RAM und RAM-Karte, MEM\$ = "S1":

Eingebautes RAM und RAM-Karte, MEM\$ = "S2":

Eingebautes RAM und 8K-RAM-Karte, MEM\$ = "B":

Eingebautes RAM und 16K-RAM-Karte, MEM\$ = "B":

Eingebautes RAM und 32K-RAM-Karte, MEM\$ = "B":

Eingebautes RAM und 64K-RAM-Karte, MEM\$ = "B":

Anfangsadresse der RAM-Disk E

B8018

B8018

B8018

B6018

B4018

B0018

A8018

## 7.3 RAM-Disk G

Die RAM-Disk G: ist eine RAM-Disk, die sich im ROM befindet. Ihre Dateien und Basicprogramme werden vom Betriebssystem für interne Berechnungen, z.B. bei der Techniker-Software, benutzt.

Sie beginnt bei C0018 mit dem folgenden Kopf:

```
C0018: FB 52 41 4D 46 49 4C 45  RAMFILE
C0020: 20 21 20 20 21 00 00 00
C0028: 00 00 F8 01 00 00 00 F8
C0030: 01 00 F8 01 00 00 00 00
C0038: 00 00 FF 00 07 01 F8 01
C0040: 00 00 FF 40 00 01 01 00
C0048: 01 01 00 01 46 00 52 00
C0050: 00 00 FF 40 00
```

Die FAT beginnt bei:

```
C0058: FF 00 00 F0 0F FF F0 0F
C0060: FF F0 0F FF F0 0F FF F0
....
```

Das Directory beginnt bei:

```
COBD8: 4D 41 54 48 20 20 20 20 MATH
COBE0: 31 23 23 20 00 00 00 00 1##
COBE8: 00 00 00 00 00 00 00 00
COBF0: 00 00 02 00 22 00 00 00
....
```

Die Texte beginnen bei:

```
C1498: 4D 46 0D 0A 22 20 22 0D MF " "
C14A0: 0A 22 20 20 20 4D 41 " MA
C14A8: 54 48 45 4D 41 54 49 43 THEMATIC
C14B0: 53 22 0D 0A 22 20 22 0D S" " "
....
```

Mit FILES"G:" erhält man das zugehörige Inhaltsverzeichnis:

MATH	.1##	34
INT	.11#	30
EQU	.12#	31
DI&INT	.13#	40
FORM	.14#	31
GRAPH	.15#	31
S-FUNC	.16#	35
SCI	.2##	30
PHYS	.21#	30
CHEM	.22#	31
EARTH	.23#	33
BIO	.24#	30
ENG	.3##	32
ELEC	.31#	39
MECH	.32#	39
STAT	.4##	32

DISTR	.41#	43
EDIT	.5#	29
COMM	.00\$	1243
DISP500	.00\$	1014
DISP501	.00\$	1585
DISP502	.00\$	1046
SCUBIC	.12\$	1313
SBDI	.42\$	4054
PRIME	.111	990
GCMLCM	.112	778
CUBIC	.121	1251
NEWTON	.122	1138
BISECT	.123	1274
INTGRL	.131	1491
DIFF	.132	1346
INTPLT	.133	946
FACT	.141	1967
TRIG	.142	3111
INTFRM	.143	2598
GREEK	.144	1435
FUNC	.151	2129
DATA	.152	2933
FIGURE	.153	8652
GAMMA	.161	986
CONST	.211	2325
M-CONV	.212	10128
MOTION	.213	947
PERIOD	.221	8130
ELECTR	.222	8483
ISOTOP	.223	15405
METEO	.231	3802
PLANET	.232	2725
AMINO	.241	5405
COMPLX	.311	2435
EE-FRM	.312	940
ELEMAG	.313	951
LAPLAC	.314	2739
ME-FRM	.321	4615
NORMAL	.411	732
t	.412	928
CHI	.413	974
F	.414	1193
EDITOR	.51#	2155
DEMO	.50#	2370

#### 8.0 Weitere Datenbereiche im RAM

Bisher wurden die wichtigsten Datenbereiche und ihre Organisation im RAM beschrieben, nämlich der Bereich für die verschiedenen Variablen, der Bereich für das Basicprogramm und die Bereiche für die verschiedenen RAM-Disk.

Es gibt weiterhin Datenbereiche für die Funktionstasten, den AER-Modus und den ENG-Modus (Techniker Software). Der STAT-Modus für statistische Berechnungen und MATRIX-Modus für Matrizen Berechnungen belegen keinen eigenen Datenbereich. Die beiden letzteren verwenden Variable, hauptsächlich numerische Feldvariable, die in der üblichen Weise im Variablenbereich abgelegt werden.

### 8.1 Funktionstasten, Organisation im Arbeitsspeicher

Die Funktionstasten sind im RAM unmittelbar hinter dem Basicprogramm abgelegt. Im ursprünglichen Zustand, d.h. nach ON/RESET und Speicher löschen, mit der vom System vorgegebenen Belegung der Funktionstasten ist folgender Speicherinhalt vorhanden:

```

BF18A: FB 46 55 4E 43 4B 45 59 FUNCKEY
BF192: 20 20 20 20 20 00 00 00
BF19A: 00 69 00 00 00 00 69 00
BF1A2: 00 69 00 00 00 00 00 00
BF1AA: 00 00 FF 00 02 00 69 00
BF1B2: 00 52 55 4E 20 FF 4C 49 RUN LI
BF1BA: 53 54 20 FF 43 4F 4E 54 ST CONT
BF1C2: 20 FF 53 41 56 45 20 FF SAVE
BF1CA: 4C 4F 41 44 20 FF 49 4E LOAD IN
BF1D2: 50 55 54 20 FF 50 52 49 PUT PRI
BF1DA: 4E 54 20 FF 47 4F 54 4F NT GOTO
BF1E2: 20 FF 47 4F 53 55 42 20 GOSUB
BF1EA: FF 52 45 54 55 52 4E 20 RETURN
BF1F2: FF FB 41 45 52 20 20 AER

```

Der Kopf des Funktionstasten-Bereiches beginnt wieder mit FB und einem kennzeichnenden Namen. Im Kopf tritt viermal der Hexwert 69 auf. Dieser ergibt, aufaddiert auf die Anfangsadresse des Funktionstasten-Bereiches, BF18A + 000069 = BF1F3. Dies ist die Anfangsadresse des nächsten Datenbereiches, des AER-Bereiches. Bei dem Wert 000069 handelt es sich also um die Länge des Funktionstasten-Bereiches. Die einzelnen Tastenbelegungen sind durch ein FF voneinander getrennt.

Belegt man nun eine Funktionstaste anders, z.B.:

KEY4,"Funktions-Tasten ", so wird dies anstelle von "SAVE " in diesen RAM-Bereich einge-tragen. Es werden zusätzlich C Speicherplätze benötigt. Dazu wird die Anfangsadresse um C vermindert und auch der Bereich für das Basicprogramm um C Plätze nach vorn geschoben. Der verfügbare Speicherbereich, mit FRE0 abfragbar, wird ebenso um C = 12 dez. vermindert. Die im Kopf des Funktionstasten-Bereiches angegebene Länge, die Anfangs 000069 betrug, wird um C auf 000075 erhöht. Es ergibt sich der folgende Speicherinhalt:

```

BF17E: FB 46 55 4E 43 4B 45 59 FUNCKEY
BF186: 20 20 20 20 20 00 00 00
BF18E: 00 75 00 00 00 00 75 00
BF196: 00 75 00 00 00 00 00 00
BF19E: 00 00 FF 00 02 00 75 00
BF1A6: 00 52 55 4E 20 FF 4C 49 RUN LI
BF1AE: 53 54 20 FF 43 4F 4E 54 ST CONT
BF1B6: 20 FF 46 75 6E 6B 74 69 Funkti
BF1BE: 6F 6E 73 2D 54 61 73 74 ons-Tast
BF1C6: 65 6E 20 FF 4C 4F 41 44 en LOAD
BF1CE: 20 FF 49 4E 50 55 54 20 INPUT
BF1D6: FF 50 52 49 4E 54 20 FF PRINT
BF1DE: 47 4F 54 4F 20 FF 47 4F GOTO GO
BF1E6: 53 55 42 20 FF 52 45 54 SUB RET
BF1EE: 55 52 4E 20 FF FB 41 45 URN AE

```

### 8.2 AER-Eingaben, Organisation im RAM

Der AER-Modus erlaubt algebraische Ausdrücke unter einem Namen einzugeben und bei Bedarf im RUN-Modus zur Durchführung abzurufen. Auch von einem Basicprogramm aus können diese Ausdrücke abgerufen werden.

Die AER-Eingaben werden im RAM im AER-Bereich abgelegt. Im ursprünglichen Zustand, d.h. nach ON/RESET, Speicher löschen befindet sich der Kopf des AER-Bereiches auf den Adressen BF1F3 bis BF21A und hat folgende Gestalt:

```

BF1F3: FB 41 45 52 20 20 20 20 AER
BF1FB: 20 20 20 20 20 00 00 00
BF203: 00 27 00 00 00 00 27 00
BF20B: 00 27 00 00 00 00 00 00
BF213: 00 00 FF 00 04 01 FF FF

```

Die auftretenden Dekremente 000027 weisen addiert zur Anfangsadresse des Kopfes, BF1F3, auf das Ende des Kopfes, BF21A.

Gibt man nun im AER-Modus das erste Beispiel der Bedienungsanleitung (01.AERA, F(A,B,C) = B\*C\*SIN A/2) ein, so erhält man im RAM:

```

BF1D3: FB 41 45 52 20 20 AER
BF1DB: 20 20 20 20 20 00
BF1E3: 00 00 00 45 00 00 00 00
BF1EB: 45 00 00 45 00 00 00 00
BF1F3: 00 00 00 00 FF 00 04 01
BF1FB: 01 9C 04 41 45 52 41 00 AERA
BF203: 41 2C 42 2C 43 00 42 2A A,B,C B*
BF20B: 43 2A FE 95 41 2F 1D 00 C* A/
BF213: 00 20 00 00 00 00 FF FF

```

Der Endpunkt des AER-Bereiches blieb erhalten. Der Kopf wurde zu niedrigeren Adressen verschoben, um die Eingabe aufzunehmen. Das Dekrement 27 im Kopf ohne Inhalt hat sich geändert in 45. Dies zeigt wieder auf das Bereichsende BF21A, wenn man es auf den Beginn, BF1D5, aufaddiert. Im codierten Text erkennt man vor dem Namen AERA seine Länge 4 und die anderen Eingaben. Der Sinus ist, wie im Basicprogramm mit seinem Token codiert. Die Zahl 2 ist wie üblich in 7 Byte codiert und hat als Kennung 1D davor, wie in einem Basicprogramm.

Gibt man nun noch das zweite Beispiel der Bedienungsanleitung (02.PLANCK, 6.626076 E-34) ein, so ergibt sich der folgende RAM-Inhalt:

```

BF1C3: FB 41 45 52 20 20 20 AER
BF1CB: 20 20 20 20 20 00 00
BF1D3: 00 00 56 00 00 00 00 56
BF1DB: 00 00 56 00 00 00 00 00
BF1E3: 00 00 00 FF 00 04 01 01
BF1EB: 1C 04 41 45 52 41 00 41 AERA A
BF1F3: 2C 42 2C 43 00 42 2A 43 ,B,C B*C
BF1FB: 2A FE 95 41 2F 1D 00 00 * A/
BF203: 20 00 00 00 00 02 8F 06

```

BF20B: 50 4C 41 4E 43 4B 1D 00 PLANCK  
BF213: DE 66 26 07 60 00 9F FF

Wieder ist das Ende des AER-Bereiches auf der Adresse BF21A geblieben und der ganze Bereich zu niedrigeren Adressen verschoben. Der Title 01 hat eine Länge von 1C, was vor der Länge des Namens steht, der Title 02 hat eine Länge von 0F. Hier, beim letzten Title, ist, wie auch beim vorangehenden Beispiel, 80 aufaddiert.

### 8.3 Organisation der Technikersoftware (ENG-Modus)

Für die Funktionen der Techniker Software werden Basicprogramme benutzt, die sich in der RAM-Disk G befinden. Diese werden bei Bedarf in den Arbeitsspeicher hinter den AER-Bereich geladen. Das ist allerdings nur möglich, wenn genug freier Speicherraum zur Verfügung steht. Ist das nicht der Fall, weil der Arbeitsspeicher durch ein umfangreiches Basicprogramm belegt ist, weil im AER-Mode zahlreiche Ausdrücke gespeichert sind oder weil eine umfangreiche RAM-Disk E: definiert wurde, so ergibt sich beim Versuch einer Funktion des ENG-Mode zu benutzen die Fehlermeldung Out of memory in program. In der Bedienungsanleitung ist am Beginn der Beschreibung der Techniker-Software in einer Tabelle angegeben wieviel Speicherraum die einzelnen Funktionen benötigen. Die Funktion Stabile Isotope hat mit 16 700 den größten Speicherbedarf. Mit FREO lässt sich abschätzen, ob die gewünschte Funktion nutzbar ist. Kehrt man nach der Verwendung einer Funktion des Techniker Software mit BREAK zum Hauptmenü zurück, so wird der von der Funktion belegte Speicherplatz wieder freigegeben.

Im folgende wird die Organisation des Arbeitsspeichers für die Primezahl Funktion genauer dargestellt.

Im Ausgangszustand, d.h. nach ON/RESET und Speicher löschen beginnt der AER-Bereich bei BF1F3 mit FB und endet bei BF21A mit FF.

Wird nun vom Hauptmenü die Primzahlfunktion aufgerufen mit ENG, MATH, INT und PRIME, so wird der ENG-Kopf gebildet und die Programme G:PRIME.111 und G:COMN.00\$ in den Arbeitsspeicher geladen. Hierzu wird der AER- und die vorangehenden Bereiche um 8C7 = 2247 dezimal nach vorn geschoben, um den benötigten Platz zu schaffen. Der ENG-Bereich beginnt in diesem Fall wie folgt:

```
BE953: FB 45 4E 47 20 20 20 20 20 ENG
BE95B: 20 24 24 24 20 00 00 00 $$$
BE963: 00 C7 08 00 00 00 C7 08
BE96B: 00 C7 08 00 00 00 00 00
BE973: 00 00 FF 00 00 00 34 00
BE97B: 00 C6 08 00 00 00 00 00
BE983: 00 00 00 00 OD 00 0A 1F
BE98B: 3A 27 1E 1C 20 50 52 49      PRI
BE993: 4D 45 2E 31 31 31 20 28 ME.111

BED51: 0D C3 50 1E 2A 43 47 20
BED59: 3A 27 1E 17 20 43 4F 4D      COM
BED61: 4E 2E 30 30 24 20 28 43 N.00$
```

Bei BF21A endet der ENG-Bereich mit FF und anschließend beginnt der Systembereich.

Im Kopf des ENG-Bereiches tritt mit 8C7 = 2247 seine Länge und damit der belegte Speicherraum auf. Zusätzlich wird noch Raum für Variable benötigt.

#### 8.4 Statistische- und Matrix-Berechnungen

Für die statistischen Berechnungen im STAT-Modus und die Matrizenberechnungen im Matrix-Modus werden keine besonderen Felder im Arbeitsspeicher belegt. Die Variablen, Datenfelder und Matrizen werden wie numerische Feldvariable einfacher Genauigkeit behandelt und in dem im Kapitel über die Variablen beschriebenen Schema im Variablenfeld abgelegt.

#### 9.0 Zusammenspiel der Datenbereiche im RAM

Bei der ausschließlichen Benutzung des eingebauten RAM, d.h. ohne eine zusätzliche RAM-Karte, beginnt der Arbeitsspeicher bei B8000. Am Anfang, beginnend bei B8018, befindet sich der Bereich für die RAM-Disk E. Daran anschließend liegt der Datenbereich für die verschiedenen Variablen, der auch bei B8018 beginnt, wenn keine RAM-Disk E initialisiert wurde. Danach kommt der Editierbereich bzw. Ablagebereich für das Basicprogramm. Hinter dem Basicprogramm kommen die Speicherbereiche für die Funktionstasten, den AER-Modus, und den ENG-Modus. Zum Schluß kommt der Systembereich und die Systemadressen, die Systemadressen beginnen stets bei BFC00. Wird Speicherbereich für Maschinenprogramme reserviert, so geschieht dies zwischen Systembereich und Systemadressen, d.h. Basicprogramm, Funktionstasten, AER-Modus, ENG-Modus und der Systembereich werden zu niedrigeren Adressen hin verschoben.

RAM-Disk E

-----  
Variable

-----  
Basicprogramme

-----  
Funktionstasten

-----  
AER-Modus

-----  
ENG-Modus

-----  
Systembereich

-----  
Maschinenprogramme

-----  
Systemadressen

B8018 -

BFC00 - BFFFF

Durch eine RAM-Disk E und durch die Benutzung von Variablen wird der Bereich für das Basicprogramm (und damit FRE0) von vorn her verkleinert, weil diese beiden Bereiche vor dem Basicprogramm angeordnet sind. Der Fixpunkt hierfür ist die Adresse B8018. Durch Erweiterung der Definitionen für Funktionstasten, durch Bildung von Formeln im AER-Modus und durch Benutzung der Technikersoftware, wird der Bereich für das Basicprogramm (und damit FRE0) von hinten her verkleinert. Der Fixpunkt hierfür ist der Beginn des Systembereiches, die Adresse BF21B.

Diese Bedingungen wurden vorangehend im einzelnen für den Fall, daß nur das eingebaute RAM benutzt wird, beschrieben.

Wird Speicher für Maschinenprogramme freigemacht, so verschiebt sich der Systembereich zu niedrigeren Adressen.

Wird der RAM-Speicher durch eine RAM-Karte erweitert, so hängt die Speicherorganisation von dem verwendeten Speichermodus, MEM\$ = "S1", "S2" oder "B" ab.

Für MEM\$ = "S1" ist Speicherverwaltung im wesentlichen die gleiche wie bei nur eingebautem RAM. Es steht lediglich zusätzlich

der Speicherraum (40000 ...) der RAM-Karte für die RAM-Diskette F: zur Verfügung.

Für MEM\$ = "B" bleibt ebenfalls die Speicherverwaltung im wesentlichen erhalten. Hier wird lediglich die untere Grenze des Arbeitsspeichers und damit der Bereich für die RAM-Diskette E und den Datenbereich für die Variablen entsprechend der Kapazität der RAM-Karte zu niedrigeren Adressen verschoben.

Für MEM\$ = "S2" ändert sich die Speicherverwaltung wesentlich. Hier werden die Variablen und das Basicprogramm im Speicherbereich der RAM-Karte (40000 ...) bei etwas anderer Organisation gehalten. Alle anderen Funktionen werden im eingebauten RAM verwaltet.

#### 10.0 Systembereich und Systemadressen

Der Systembereich für Puffer usw. liegt nach ON/RESET, Speicherlöschen von BF21B - BFBFF.

Wurde Speicherplatz für Maschinenprogramme reserviert, so liegt er entsprechend zu niedrigeren Adressen verschoben, also vor dem reservierten Speicherplatz.

Ein fester Bereich für Systemadressen reicht dagegen stets von BFC00 bis BFFFF. Der für Maschinenprogramme reservierte Speicherplatz liegt stets davor.

Systembereich: BF21B - BFBFF

Nach Reservierung von Speicherplatz für  
Maschinenprogramme entsprechend zu niedrigeren  
Adressen verschoben

BF87F-82                    Werte für Basic-Befehl AUTO  
BF882/1                    Startzeilennummer  
BF880/7F                    Ergänzungswert

BF8E3/2                    Wert für Basic-Befehl WAIT

Systemadressen: BFC00 - BFFFF

BFC15-7                    BFC17/6/5 = Anfangsadresse des Arbeitsspeichers.  
Bei nur eingebautem RAM: B8000.  
Bei 16K-RAM-Karte, MEM\$ = "B",: B4000.

BFC1B-D                    BFC1D/C/B = Beginn des Systembereiches hinter dem  
AER-Bereich.  
Nach ON/RESET, Speicher Löschen: BF21B.  
Nach Reservierung von Speicherplatz für  
Maschinenprogramme entsprechend zu niedrigeren  
Adressen verschoben.

BFC5C                    Sonderanzeigeelement DEG gesetzt:  
Bit 0 (letztes Bit) gesetzt  
Bit 1 (vorletztes Bit) nicht gesetzt  
Sonderanzeigeelement RAD gesetzt:  
Bit 0 (letztes Bit) nicht gesetzt  
Bit 1 (vorletztes Bit) nicht gesetzt  
Sonderanzeigeelement GRAD gesetzt:  
Bit 0 (letztes Bit) nicht gesetzt  
Bit 1 (vorletztes Bit) gesetzt

##### 1. Zeichensatz/ 2. Zeichensatz

BFC8A	55	00
BFC8B	24	FD
BFC8C	0F	0D
BFC8D	95	40
BFC8E	26	FF
BFC8F	0F	0D
BFC93	15	C0
BFC94	25	FD

BFC95	OF            OD Nach ON/RESET, Speicher löschen ist der erste Zeichensatz gültig. Im Bereich von 20 bis 7F unterscheiden sie sich nicht. Über 7F unterscheiden sie sich auf der Anzeige, auf dem Thermostrucker CE-126P werden sie nicht dargestellt.
BFC97	00, falls Sonderanzeigeelement BATT nicht gesetzt 01, falls Sonderanzeigeelement BATT gesetzt
BFC99	Sonderanzeigeelement CAPS gesetzt: Bit 3 (viertletztes Bit) gesetzt Sonderanzeigeelement CAPS nicht gesetzt: Bit 3 (viertletztes Bit) nicht gesetzt
BFC9A	Sonderanzeigeelement PRINT an: Bit 0 (letztes Bit) gesetzt. Sonderanzeigeelement PRINT aus: Bit 0 (letztes Bit) nicht gesetzt.
BFCA1	00: Anzeige schwarz auf weiß FF: Anzeige weiß auf schwarz
BFD1A,B,C	Adresse in BFD1C/B/A gibt die untere Grenze des für Maschinenprogramme reservierte Speicherbereiches, der bis BFBFF reicht, an. Nach ON-RESET: (BFD1A)=BFC00 Siehe Abschnitt Speicherreservierung für Maschinenprogramme
BFD1D/E	00/0C, falls CE-126P nicht angeschlossen 02/14, falls CE-126P angeschlossen
BFD25	Wert für Basic-Befehl CONSOLE, Voreinstellung &27
BFD33/4 BFD3B/C	Werte für Basic-Befehl OPEN
BFD33 BFD34 BFD3B BFD3C	Voreinstellung: 3C Voreinstellung: 21 Voreinstellung: 01 Voreinstellung: 1A
Baudrate 300 600 1200 2400 4800 9600	Höherwertiges Halbbyte von BFD33 1 2 3 4 5 6
Stopppbit 1 2	Bit "1" von BFD33 nicht gesetzt gesetzt
Wortlänge 8 Bit	Bit "2" von BFD33 nicht gesetzt

7 Bit	gesetzt	
Parität	Bit "4", E            nicht gesetzt O            gesetzt N            gesetzt	Bit "8" von BFD33 nicht gesetzt nicht gesetzt gesetzt
Umstellungs code	Bit "1" von BFD34	
N            nicht gesetzt S            gesetzt		
XON	Bit "2", N            nicht gesetzt X            gesetzt	Bit "4" von BFD34 nicht gesetzt gesetzt
Begrenzer	BFD3B	
C            01 F            02 L            03		
Dateiende	BFD3C, Voreinstellung 1A	
	Diese OPEN-Werte sind Spiegel, Einpoken genügt nicht.	

## 11.0 Zeichensätze

Es gibt zwei Zeichensätze. Der erste Zeichensatz ist nach ON/RESET und Speicher löschen eingeschaltet. Der zweite Zeichensatz wird vom PC-E500 in speziellen Fällen benutzt, z.B. bei der Technikersoftware, im Hauptmenü nach: ENG, MATH, EQU, CUBIC.

Geht man von hier mit Break, Break, Break wieder zum Hauptmenü zurück, so ist wieder zum ersten Zeichensatz zurückgeschaltet.

Kehrt man jedoch statt mit Break mit Reset zurück, so bleibt der zweite Zeichensatz eingeschaltet.

Die beiden Zeichensätze lassen sich auch durch Einpoken bestimmter Werte in den Bereich der Systemadressen einschalten:

1. Zeichensatz      2. Zeichensatz

BFC8A	55	00
BFC8B	24	FD
BFC8C	0F	0D
BFC8D	95	40
BFC8E	26	FF
BFC8F	0F	0D
BFC93	15	C0
BFC94	25	FD
BFC95	0F	0D

Die Darstellung der beiden Zeichensätze in BASIC, also mit

PRINT CHR\$ A,

ergibt für die Werte A = 0 bis A = 1F keine Anzeige. Lediglich mit dem Wert A = 7 wird bei beiden Zeichensätzen ein BEEB erzeugt.

Oberhalb 1F, also ab 20 wird beim 1. Zeichensatz der IBM Zeichensatz erzeugt. Beim 2. Zeichensatz sind die Zeichen für A = 20 bis A = 7F die gleichen wie beim 1. Zeichensatz. Darüber, von A = 80 bis A = FF unterscheiden sich die beiden Zeichensätze. Die anschließenden Tabellen listet die beiden Zeichensätze auf, so wie sie auf der Anzeige erscheinen.

Auf dem Drucker CE-126P werden die Zeichen oberhalb von A = 7F nicht dargestellt.

## 1. Zeichensatz:

Tabelle der Zeichen auf der Anzeige (IBM Zeichensatz, ab &20)

Hex-zahl	Zeichen	Hex-zahl	Zeichen	Hex-zahl	Zeichen	Hex-zahl	Zeichen
20		21	!	22	"	23	#
24	\$	25	%	26	&	27	,
28	(	29	)	2A	*	2B	+
2C	,	2D	-	2E	.	2F	/
34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;
3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C
44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K
4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S
54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[
5C	\	5D	]	5E	^	5F	-
60	,	61	a	62	b	63	c
64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k
6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s
74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{
7C	-	7D	}	7E	~	7F	△
80	ç	81	ä	82	é	83	å
84	ä	85	ë	86	à	87	ç
88	ê	89	ë	8A	è	8B	å
8C	í	8D	í	8E	ä	8F	ö
90	é	91	æ	92	ó	93	ú
94	ö	95	ø	96	ü	97	ù
98	ÿ	99	ÿ	9A	ü	9B	ç
9C	ƒ	9D	ÿ	9E	ƒ	9F	ú
A0	á	A1	í	A2	ó	A3	ú
A4	ñ	A5	ñ	A6	ä	A7	ë
A8	ç	A9	í	AA	æ	AB	ø
AC	é	AD	í	AE	»	AF	«
B0	ö	B1	ø	B2	»	B3	«
B4	ÿ	B5	ÿ	B6	—	B7	—
B8	ƒ	B9	ÿ	BA	—	BB	—
BC	—	BD	—	BE	—	BF	—
C0	—	C1	—	C2	—	C3	—
C4	—	C5	+	C6	—	C7	—
C8	—	C9	=	CA	—	CB	—
CC	—	CD	=	CE	—	CF	—
D0	—	D1	—	D2	—	D3	—
D4	—	D5	—	D6	—	D7	—
D8	+	D9	—	DA	—	DB	—
DC	—	DD	—	DE	—	DF	—

1. Zeichensatz:

Tabelle der Zeichen auf der Anzeige (IBM Zeichensatz, ab &20)

Hex-zahl	Zeichen	Hex-zahl	Zeichen	Hex-zahl	Zeichen	Hex-zahl	Zeichen
E0	α	E1	β	E2	Γ	E3	π
E4	Σ	E5	σ	E6	μ	E7	τ
E8	Φ	E9	Θ	EA	Ω	EB	δ
EC	∞	ED	φ	EE	ε	EF	η
F0	≡	F1	±	F2	≥	F3	≤
F4	ʃ	F5	J	F6	÷	F7	≈
F8	·	F9	.	FA	.	FB	✓
FC	n	FD	²	FE	-	FF	

2. Zeichensatz:

Tabelle der Zeichen auf der Anzeige

Hex-zahl	Zeichen								
20		21	!	22	"	23	#	24	,
24	\$	25	%	26	&	27	/	28	(
28	)	29	-	2A	*	2B	+	2C	,
2C	,	2D	-	2E	.	2F	/	34	4
34	35	35	5	36	6	37	7	38	8
38	9	39	9	3A	:	3B	;	3C	<
3C	=	3D	>	3E	>	3F	?	40	@
40	A	41	B	42	C	43	G	44	D
44	E	45	F	46	G	47	K	48	H
48	I	49	J	4A	N	4B	O	4C	L
4C	M	4D	R	4E	S	4F	W	50	P
50	Q	51	V	52	Z	53	[	54	T
54	U	55	Y	56	^	57	]	58	X
58	Y	59	Z	5A	b	5B	c	5C	\
5C	a	5D	j	5E	f	5F	g	60	ı
60	e	61	n	62	o	63	k	64	d
64	i	65	m	66	s	67	l	68	h
68	r	69	u	6A	t	6B	o	70	p
70	v	71	q	72	v	73	w	74	t
74	z	75	y	76	z	77	{	78	x
78	}	79	~	7A	z	7B	Γ	7C	i
7C	Δ	7D	A	7E	~	7F	H	80	E
80	81	85	E	82	B	83	N	84	Θ
84	88	89	I	86	Z	87	Φ	88	M
88	8D	91	N	8E	K	8F	Φ	90	Π
90	95	95	P	92	Σ	93	Φ	94	Y
94	99	99	Φ	96	X	97	Φ	98	Ω
98	9D	9D	N	9A	η	9B	Φ	9C	μ
9C	9A	9A	Φ	9E	δ	9F	Φ	A0	α
A0	A1	A1	ε	A2	β	A3	Φ	A4	δ
A4	A5	A5	ε	A6	γ	A7	Φ	A8	θ
A8	A9	A9	γ	AA	κ	AB	Φ	AC	μ
AC	AD	AD	γ	AE	ξ	AF	Φ	B0	π
B0	B1	B1	φ	B2	σ	B3	Φ	B4	ν
B4	B5	B5	φ	B6	χ	B7	Φ	B8	ω
B8	B9	B9	φ	BA	τ	BB	Φ	BC	◆
BC	BD	BD	φ	BE	λ	BF	Φ	C0	◆
C0	C1	C1	φ	C2	μ	C3	Φ	C4	γ
C4	C5	C5	φ	C6	σ	C7	Φ	CC	+
CC	C9	C9	φ	CA	ε	CB	Φ	DO	ʃ
DO	CD	CD	φ	CE	◦	CF	Φ	D4	ʒ
D4	D1	D1	φ	D2	◦	D3	Φ	D8	ʒ
D8	D5	D5	φ	D6	◦	D7	Φ	DC	q
DC	D9	D9	φ	DA	◦	DB	Φ		
	DD	DD	φ	DE	◦	DF			

2. Zeichensatz:

Tabelle der Zeichen auf der Anzeige

Hex-zahl	Zeichen	Hex-zahl	Zeichen	Hex-zahl	Zeichen	Hex-zahl	Zeichen
E0	n	E1	t	E2	*	E3	
E4	>	E5	<	E6	>	E7	↑
E8	[	E9	]	EA	.	EB	✓
EC	z	ED	A	EE	ß	EF	e
F0	ç	F1	þ	F2	ä	F3	ö
F4	N	F5	ñ	F6	ö	F7	/
F8	ß	F9	:	FA	^	FB	→
FC	±	FD	↑	FE	↓	FF	↔

**12.0 Der Mikroprozessor des PC-E500 und seine Architektur**

Die CPU des PC-E500 ist eine 8-Bit CMOS CPU und trägt, laut Service Manual die Bezeichnung SC-62015B01.

Der Adressraum für das äußere RAM/ROM umfaßt 1 MByte. D.h. die Adressierung geschieht mit 20 Bit oder 5 Hexstellen.

Ein CPU internes, inneres RAM hat 256 Byte.

Die Zykluszeit wurde über den WAIT Befehl zu ca. 1,3 Mikrokunden gemessen.

Die Architektur der CPU

Die CPU Register

Es gibt 1-Bit-, 8-Bit-, 16-Bit- und 20-Bit-Register:

<u>1</u>	<u>1</u>	<u>F / Z/Cy/</u>	F Flag Z Zero-Flag Cy Carry-Flag
<u> </u>	<u> </u>	<u>A 8/</u>	A Akkumulator
<u> </u>	<u> </u>	<u>IL 8/</u>	IL Counter, low
<u> </u>	<u> </u>	<u>B 8/ A 8/</u>	BA Hilfs-Akkumul. mit Akkumul.
<u> </u>	<u> </u>	<u>IH 8/ IL 8/</u>	I Zähler
<u> </u>	<u> </u>	<u>X 20/</u>	X Pointer
<u> </u>	<u> </u>	<u>Y 20/</u>	Y Pointer
<u> </u>	<u> </u>	<u>U 20/</u>	U Daten Stackp.
<u> </u>	<u> </u>	<u>S 20/</u>	S CALL Stackp.
<u> </u>	<u> </u>	<u>PS 4/ PC 16/</u>	PC Programmcount. 16 Bit = 64 KB
<u> </u>	<u> </u>		PS Segmentregist. 4 Bit = 16 Seg

1-Bit-Register

Z, das Zero-Flag, wird bei arithmetischen/logischen Funktionen Eins gesetzt, wenn das Ergebnis Null war. War das Ergebnis nicht Null, so wird das Zero-Flag Null gesetzt.

Cy, das Carry-Flag, wird bei arithmetischen Funktionen Eins gesetzt, wenn es einen Überlauf oder ein borrow gab, anderenfalls wird es Null gesetzt.

Beide Register sind im Flag-Register, F, zusammengefaßt und können so auf den Stack gegeben oder von dort geholt werden.

/ 0 / 0 / 0 / 0 / 0 / 0 / Z/ Cy/

#### 8-Bit-Register

A, der Akkumulator, ist das Hauptregister, in dem häufig die Ergebnisse von arithmetischen oder logischen Operationen erscheinen.

B, der Hilfsakkumulator, kann nur zusammen mit dem Akkumulator als Doppelregister BA angesprochen werden.

IH ist das höherwertige Byte des 2-Byte Register I. Es kann nur mit IL zusammen angesprochen werden.

IL ist das geringerwertige Byte des 2-Byte Register I. Es kann einzeln angesprochen werden. Wird es mit einem Immediate geladen, so erhält IH den Wert Null.

F Flagregister.

#### 16-Bit-Register

BA faßt als Doppelregister den Hilfs-Akkumulator, B, und den Akkumulator, A, zusammen.

I ist ein 16-Bit Zähler, dessen geringerwertiges Byte einzeln als IL angesprochen werden kann.

#### 20-Bit-Register

X und Y sind 20-Bit Pointer. Sie können das äußere RAM mit seinem Adressbereich von sechzehn 64KB-Segmenten direkt adressieren.

U ist der Stackpointer für Daten, die mit den Befehlen PUSHU-/POPU- auf den U-Stack gegeben/geholt werden.

S ist der Stackpointer für die Rückkehradressen der CALL/RET-Befehle.

Beide Stackpointer, U und S, müssen bei der Rückkehr von einem Maschinenprogramm in den Basicinterpreter den gleichen Wert haben wie vor dem Ablauf des Maschinenprogrammes.

PC ist der 16-Bit Programmcounter, der zusammen mit dem 4-Bit Page Segment Register, PS, insgesamt also 20 Bit, den Speicherraum von 1 MByte adressieren kann.

#### Register des Inneren CPU-RAM

Es gibt ein CPU-internes RAM mit 256 Byte. Davon stehen 236 Byte (Adressen 00 - EB) zur freien Verfügung. Die restlichen 20 (Adresse EC - FF) sind für spezielle Ein-Byte Register reserviert, wovon (EC) = BP, (ED) = PX und (EE) = PY spezielle Pointer für das interne RAM sind.

EC = BP = Basis Pointer für inneres RAM  
ED = PX = PX Pointer für inneres RAM  
EE = PY = PY Pointer für inneres RAM

EF = AMC = Adress Modify Control  
F0 = KOL = Key Output Buffer  
F1 = KOH = Key Output Buffer  
F2 = KI = Key Input Buffer  
F3 = EOL = E Port Output Buffer  
F4 = EOH = E Port Output Buffer  
F5 = EIL = E Port Input Buffer  
F6 = EIL = E Port Input Buffer  
F7 = UCR = UART Control Register  
F8 =USR = UART Status Register  
F9 = RXD = UART Recieve Buffer  
FA = TXD = UART Transmit Buffer  
FB = IMR = Interrupt Mask Register  
FC = ISR = Interrupt Status Register  
FD = SCR = System Control Register  
FE = LCC = LCD Contrast Control  
FF = SSR = System Status Register

## 12.1 Der Befehlssatz

Der Befehlssatz der CPU ist recht umfangreich. Es werden nicht nur die 256 Möglichkeiten des 8-Bit Codes genutzt sondern es gibt auch Mehrbytebefehle.

Der Befehlssatz der CPU kann in folgende Gruppen unterteilt werden:

- Transportbefehle
- Laden der Register der CPU oder des inneren RAM mit Zahlenwerten, Immediate
- Transport zwischen den Registern der CPU
- Transport innerhalb des inneren RAM
- Transport zwischen den CPU-Registern und dem inneren RAM
- Transport zwischen den CPU-Registern und dem äußeren RAM
- Transport zwischen dem inneren RAM und dem äußeren RAM
- Blocktransfer
- Transport zwischen den Registern der CPU und dem Stack
- Sprungbefehle
- Unterprogrammbefehle
- Additions/Subtraktions Befehle
- Pointer Additions Befehle
- Inkrement/Dekrement Befehle
- Logische Befehle
- Bit-Test Befehle
- Vergleich Befehle, Compare
- Carry Flag Control Befehle
- Swap- Rotate- Shift-Befehle
- Reset und Interrupt Befehle
- CPU Control Befehle

Diese Gruppen von Maschinenbefehlen werden im folgenden in komprimierter Form beschrieben.

Für eine ausführliche, mit zahlreichen Beispielen versehene Beschreibung sei auf das Maschinensprachendbuch verwiesen. Dort wird auch ein Disassembler und ein Breakpointmonitor für das Aus- testen von selbst entwickelten Maschinenprogrammen abgedruckt. Außerdem werden dort die Maschinenroutinen für die Bedienung der Anzeige, der Tastatur und des Druckers behandelt.

## 12.2 Der Adressraum

### Das innere RAM

Das innere RAM umfaßt 256 Byte. Davon stehen 236 (00 - EB) frei zur Verfügung. Die restlichen 20 (EC - FF) sind für spezielle Register reserviert.

00 - EC	frei benutzbar
EB - FF	spezielle Register

### Das äußeren RAM

Der Programmcounter (inklusive PS Register) hat 20 Bit. Der durch den Programmcounter adressierbare Speicherbereich beträgt also sechzehn 64K-Segmente. Dies entspricht einem Adressraum von einem MByte, darstellbar durch fünf Hexstellen oder 2,5 Byte.

Von diesem Bereich sind reserviert:

00000 - 03FFF	SH-26 Display
04000 - 07FFF	weiterer Display Bereich
....	
....	
FFFFA	Interrupt Vektor, low Byte
FFFFB	Interrupt Vektor, high Byte
FFFC	Interrupt Vektor, Extension(Segment)
FFFD	Reset Vektor, low Byte
FFFE	Reset Vektor, high Byte
FFFFF	Reset Vektor, Extension(Segment)

### 12.3 Die Adressierung

#### Die Adressierung im inneren RAM

Die 256 (dezimal) Register des inneren RAM können auf verschiedene Arten adressiert werden. Da sie über ihre Adressen angesprochen werden, werden stets runde Klammern ( ) benutzt. Der Inhalt einer Speicherstelle des äußeren RAM wird dagegen durch die in eckige Klammern [ ] gesetzte Adresse bezeichnet.

Die Adressierung im inneren RAM kann direkt durch ein Byte erfolgen oder relativ in Bezug auf die Pointer BP (Base Pointer), PX oder PY.

(*)	bedeutet irgend eine der folgenden Adressierungen
(n)	direkte Adressierung
(BP+n)	Adresse = BP+n. Falls BP+n > FF, Adresse = BP+n -100
(PX+n)	Adresse = PX+n. Falls PX+n > FF, Adresse = PX+n -100
(PY+n)	Adresse = PY+n. Falls PY+n > FF, Adresse = PY+n -100
(BP+PX)	Adresse = BP+PX. Falls BP+PX > FF, Adresse = BP+PX-100
(BP+PY)	Adresse = BP+PY. Falls BP+PY > FF, Adresse = BP+PY-100

PY wird im zweiten Argument eines Befehls benutzt, wenn im ersten Argument bereits PX auftrat.

Die übliche Adressierungsart ist (BP+n), sie erfordert keinen Vorcode. Alle anderen werden durch einen Vorcode bestimmt. Ein Vorcode erhöht die Zykluszahl um eins.

#### Die Adressierung im äußeren RAM

Der Speicherraum von 1 MByte im äußeren RAM wird mit 20 Bit oder fünf Hexstellen adressiert.

Die Adressierung kann direkt, durch eine Zahlenangabe, oder indirekt, über den Inhalt eines Registers, erfolgen. Weiterhin kann sie short, near oder far sein.

Short bedeutet eine Adressierung mit einem Byte (zweistellige Hexzahl oder 8-Bit-Register) relativ zum Programmcounter um plus/minus 255.

Near bedeutet Adressierung mit zwei Byte (vierstellige Hexzahl oder 16-Bit-Register) im durch die erste Hexstelle des Programmcounters (PS-Register) vorgegebenen Speichersegment.

Far bedeutet Adressierung mit 2,5 Byte (fünfstellige Hexzahl oder 20-Bit-Register), also Adressierung im gesamten adressierbaren Speicherbereich.

Für die indirekte Adressierung kann entweder ein 3-Byte CPU-Register benutzt werden oder drei aufeinanderfolgende Register des inneren RAM.

Bei der indirekten Adressierung durch ein 3-Byte CPU-Register, kann bei der Ausführung des Befehls auch der Registerinhalt inkrementiert werden (nach der Befehlausführung) oder dekrementiert

(vor der Befehlausführung). Weiterhin kann die Adresse auch durch den Registerinhalt plus oder minus einem Offset gegeben sein:

[*]	bedeutet eine der folgenden Adressierungen: Inhalt der Speicherzelle mit der Adresse lmn, die erste vier Bit von 1 werden nicht beachtet.
[lmn]	Inhalt der Speicherzelle deren Adresse durch den Inhalt eines 3-Byte Registers (20 Bit) gegeben ist.
[r <sub>3</sub> ]	Inhalt der Speicherzelle deren Adresse durch den Inhalt eines 3-Byte Registers (20 Bit) gegeben ist. Nach der Ausführung des Befehls wird der Inhalt des Registers entsprechend erhöht.
[r <sub>3</sub> ++]	Bei 1-Byte Transfer um 1. Bei 2-Byte Transfer um 2. Bei 3-Byte Transfer um 3.
[--r <sub>3</sub> ]	Inhalt der Speicherzelle deren Adresse durch den Inhalt eines 3-Byte Registers (20 Bit) gegeben ist. Vor der Ausführung des Befehls wird der Inhalt des Registers entsprechend vermindert Bei 1-Byte Transfer um 1. Bei 2-Byte Transfer um 2. Bei 3-Byte Transfer um 3.
[r <sub>3</sub> +n]	Inhalt der Speicherzelle deren Adresse durch den um n erhöhten Inhalt eines 3-Byte Registers (20 Bit) gegeben ist.
[r <sub>3</sub> -n]	Inhalt der Speicherzelle deren Adresse durch den um n verminderten Inhalt eines 3-Byte Registers (20 Bit) gegeben ist.

Bei der indirekten Adressierung durch drei Byte des inneren RAM werden diese drei Byte in umgekehrter Reihenfolge benutzt. Das erste Byte ist das low Byte der Adresse, das zweite Byte ist das medium Byte und das dritte Byte das high Byte. Beim letzteren werden die ersten vier Bit ignoriert. Es können die Pointer des inneren RAM oder auch ein Offset benutzt werden:

[*]	bedeutet eine der folgenden Adressierungen: Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei n beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte.
[(n)]	Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei BP+n beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte.
[(BP+n)]	Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei PX+n beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte.
[(PY+n)]	Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei PY+n beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte.
[(BP+PX)]	Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei BP+PX beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte.

- $[(m)+/-n]$  high Adressbyte.  
Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei m beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte. Die so erhaltene Adresse wird noch um n erhöht/vermindert.
- $[(BP+m)+/-n]$  Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei BP+m beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte. Die so erhaltene Adresse wird noch um n erhöht/vermindert.
- $[(PX+m)+/-n]$  Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei PX+m beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte. Die so erhaltene Adresse wird noch um n erhöht/vermindert.
- $[(PY+m)+/-n]$  Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei PY+m beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte. Die so erhaltene Adresse wird noch um n erhöht/vermindert.
- $[(BP+PX)+/-n]$  Inhalt der Speicherzelle deren Adresse gegeben ist durch die bei BP+PX beginnenden drei Zellen des inneren RAM und zwar in der Ordnung: low, medium, high Adressbyte. Die so erhaltene Adresse wird noch um n erhöht/vermindert.

Beispiel für  $[(m)+n]$

Es sei:  $[63510] = A7$   
 Es sei:  $(08) = 12, (09) = 34, (0A) = 56$   
 $MV A, [(08)+FE]$  bewirkt:  $A = [63412+FE] = [63510] = A7$

#### 12.4 Kurzbeschreibung der Befehle

Abkürzung für die CPU-Register:

$R_1 =$	0	1					
Register	A, IL, BA,	I, X, Y, U, S					
$R_2 =$	2	3					
Register	A, IL, BA,	I, X, Y, U, S					
$R_3 =$	4	5	6	7			
Register	A, IL, BA,	I, X, Y, U, S					
$r_1 =$	0	1					
Register	A, IL, BA,	I, X, Y, U, S					
$r_2 =$	2	3					
Register	A, IL, BA,	I, X, Y, U, S					
$r_3 =$	4	5	6	7			
Register	A, IL, BA,	I, X, Y, U, S					

Die Vorcode für die verschiedenen Adressierungsarten im inneren RAM sind:

Für die Vorcode, \*\*, bei nur einem Argument aus dem inneren RAM:

(*)	**			
(N)	30			
(BP+N)	kein			
(PX+N)	34			
(BP+PX)	24			

Für die Vorcode, \*#, bei zwei Argumenten aus dem inneren RAM:

(*)/(#)	(n)	(BP+n)	(PY+n)	(BP+PY)
(N)	32	30	33	31
(BP+N)	22	kein	23	21
(PX+N)	36	34	37	35
(BP+PX)	26	24	27	25

Ein Vorcode erhöht die Zykluszahl um eins.

Immediate Befehle, laden der Register der CPU oder des inneren RAM mit Zahlenwerten

#### Laden eines CPU-Registers

Befehl	Code	Funktion		C/Z	Zy
	1./2.	weitere			
MV A,n	08	n	n --> A		
MV IL,n	09	n	n --> IL, 00 --> IH	./.	2
				./.	3

Befehl	Code	Funktion	C/Z	Zy
	1./2.	weitere		
	Byte	Byte		
MV BA,mn	0A	n,m mn --> BA	./.	3
MV I,mn	0B	n,m mn --> I	./.	3
MV X,lmn	0C	n,m,l lmn --> X	./.	4
MV Y,lmn	0D	n,m,l lmn --> Y	./.	4
MV U,lmn	0E	n,m,l lmn --> U	./.	4
MV S,lmn	0F	n,m,l lmn --> S	./.	4

#### Laden eines Registers des inneren RAM

MV (*),n	**CC	N,n n --> (N)	./.	3/4
MVW (*),mn	**CD	N,n,m n --> (N) m --> (N+1)	./.	4/5
MVP (*),lmn	**DC	N,n,m,l n --> (N) m --> (N+1) l --> (N+2)	./.	5/6

#### Transport zwischen den CPU-Registern

Befehl	Code	Funktion	C/Z	Zy
	1./2.	weitere		
	Byte	Byte		
MOV A,B	74	B --> A	./.	1
MOV B,A	75	A --> B	./.	1
MOV R <sub>2</sub> ,r <sub>2</sub>	FD	R <sub>2</sub> r <sub>2</sub> r <sub>2</sub> --> R <sub>2</sub>	./.	2
MOV R <sub>3</sub> ,r <sub>3</sub>	FD	R <sub>3</sub> r <sub>3</sub> r <sub>3</sub> --> R <sub>3</sub>	./.	2
EX A,B	DD	A <--> B	./.	3
EX R <sub>2</sub> ,r <sub>2</sub>	ED	R <sub>2</sub> r <sub>2</sub> r <sub>2</sub> <--> R <sub>2</sub>	./.	4
EX R <sub>3</sub> ,r <sub>3</sub>	ED	R <sub>3</sub> r <sub>3</sub> r <sub>3</sub> <--> R <sub>3</sub>	./.	4

#### Transport zwischen den Registern des inneren RAM

Befehl	Code	Funktion	C/Z	Zy
	1./2.	weitere		
	Byte	Byte		
MV (*),(#)	*#C8	N,n (#) --> (*)	./.	6/7
EX (*),(#)	*#C0	N,n (#) <--> (*)	./.	7/8

#### Transport von zwei Byte (Wort)

MVW (*),(#)	*#C9	N,n (#),(#+1) --> (*),(*+1)	./.	8/9
EXW (*),(#)	*#C1	N,n (*),(*+1) <--> (#),(#+1)	./.	10/11

#### Transport von drei Byte

Befehl	Code	Funktion	C/Z	Zy
	1./2.	weitere		
	Byte	Byte		
MVP (*),(#)	*#CA	N,n (#),(#+1),(#+2) --> (*),(*+1),(*+2)	./.	10/11
EXP (*),(#)	*#C2	N,n (#),(#+1),(#+2) <--> (*),(*+1),(*+2)	./.	13/14

#### Blocktransport

##### Inkrementierender Blocktransport

MVL (*),(#)	*#CB	N,n (#)...(#+I-1) --> (*)...(*+I-1)	./.	5/6+2*I
EXL (*),(#)	*#C3	N,n (#)...(#+I-1) <--> (*)...(*+I-1)	./.	5/6+3*I

##### Dekrementierender Blocktransport

MVLD (*),(#)	*#CF	N,n (#-I+1)...(#) --> (*-I+1)...(*)	./.	5/6+2*I
--------------	------	--	-----	---------

#### Transport zwischen CPU-Registern und dem inneren RAM

Befehl	Code	Funktion	C/Z	Zy
	1./2.	weitere		
	Byte	Byte		
MV A,(*)	**,80	n (*) --> A	./.	3/4
MV IL,(*)	**,81	n (*) --> IL	./.	4/5
MV BA,(*)	**,82	n (**+1),(*) --> BA	./.	4/5
MV I,(*)	**,83	n (**+1),(*) --> I	./.	4/5
MV X,(*)	**,84	n (**+2),(**+1),(*) --> X	./.	5/6
MV Y,(*)	**,85	n (**+2),(**+1),(*) --> Y	./.	5/6
MV U,(*)	**,86	n (**+2),(**+1),(*) --> U	./.	5/6
MV S,(*)	**,87	n (**+2),(**+1),(*) --> S	./.	5/6
MV (*),A	**,A0	N A --> (*)	./.	3/4
MV (*),IL	**,A1	N IL --> (*)	./.	3/4
MV (*),BA	**,A2	N BA --> (**+1),(*)	./.	4/5
MV (*),I	**,A3	N I --> (**+1),(*)	./.	4/5
MV (*),X	**,A4	N X --> (**+2),(**+1),(*)	./.	5/6
MV (*),Y	**,A5	N Y --> (**+2),(**+1),(*)	./.	5/6
MV (*),U	**,A6	N U --> (**+2),(**+1),(*)	./.	5/6
MV (*),S	**,A7	N S --> (**+2),(**+1),(*)	./.	5/6

Transport zwischen den CPU-Registern und den Speicherplätzen des äußeren RAM

Transport: CPU Register ins äußere RAM (direkt adressiert)

Befehl	Code 1./2. Byte	Funktion weitere Byte	C/Z	Zy
MV [lmn], A	A8	n,m,l    A --> [lmn]	./.	5
MV [lmn], IL	A9	n,m,l    IL --> [lmn]	./.	5
MV [lmn], BA	AA	n,m,l    BA --> [lmn+1],[lmn]	./.	6
MV [lmn], I	AB	n,m,l    I --> [lmn+1],[lmn]	./.	6
MV [lmn], X	AC	n,m,l    X --> [lmn+2],[lmn+1],[lmn]	./.	7
MV [lmn], Y	AD	n,m,l    Y --> [lmn+2],[lmn+1],[lmn]	./.	7
MV [lmn], U	AE	n,m,l    U --> [lmn+2],[lmn+1],[lmn]	./.	7
MV [lmn], S	AF	n,m,l    S --> [lmn+2],[lmn+1],[lmn]	./.	7

Transport: Äußeres RAM (direkt adressiert) in CPU Register

Befehl	Code	Funktion	C/Z	Zy
MV A,[lmn]	88	n,m,l    [lmn] --> A	./.	6
MV IL,[lmn]	89	n,m,l    [lmn] --> IL	./.	6
MV BA,[lmn]	8A	n,m,l    [lmn+1],[lmn] --> BA	./.	7
MV I,[lmn]	8B	n,m,l    [lmn+1],[lmn] --> I	./.	7
MV X,[lmn]	8C	n,m,l    [lmn+2],[lmn+1],[lmn] --> X	./.	8
MV Y,[lmn]	8D	n,m,l    [lmn+2],[lmn+1],[lmn] --> Y	./.	8
MV U,[lmn]	8E	n,m,l    [lmn+2],[lmn+1],[lmn] --> U	./.	8
MV S,[lmn]	8F	n,m,l    [lmn+2],[lmn+1],[lmn] --> S	./.	8

Transport: CPU Register ins äußere RAM (indirekt adressiert)

Befehl	Code 1./2. Byte	Funktion weitere Byte	C/Z	Zy
<b>1-Byte Quelle:</b>				
MV [R <sub>3</sub> ], r <sub>1</sub>	Br	OR    r <sub>1</sub> --> [R <sub>3</sub> ]	./.	4
MV [R <sub>3</sub> +r], r <sub>1</sub>	Br	2R    r <sub>1</sub> --> [R <sub>3</sub> ], R <sub>3</sub> --> R <sub>3</sub> +1	./.	4
MV [-R <sub>3</sub> ], r <sub>1</sub>	Br	3R    R <sub>3</sub> --> R <sub>3</sub> -1, r <sub>1</sub> --> [R <sub>3</sub> ]	./.	5
MV [R <sub>3</sub> +N], r <sub>1</sub>	Br	8R,N    r <sub>1</sub> --> [R <sub>3</sub> +N]	./.	6
MV [R <sub>3</sub> -N], r <sub>1</sub>	Br	CR,N    r <sub>1</sub> --> [R <sub>3</sub> -N]	./.	6
MV [(*)], r <sub>1</sub>	**, B(8+r)	00,N    r <sub>1</sub> --> [(*)]	./.	9/10
MV [(*)+N], r <sub>1</sub>	**, B(8+r)	80,M,N    r <sub>1</sub> --> [(*)+N]	./.	11/12
MV [(*)-N], r <sub>1</sub>	**, B(8+r)	C0,M,N    r <sub>1</sub> --> [(*)-N]	./.	11/12
<b>2-Byte Quelle:</b>				
MV [R <sub>3</sub> ], r <sub>2</sub>	Br	OR    r <sub>2</sub> --> [R <sub>3</sub> +1],[R <sub>3</sub> ]	./.	5
MV [R <sub>3</sub> +r], r <sub>2</sub>	Br	2R    r <sub>2</sub> --> [R <sub>3</sub> +1],[R <sub>3</sub> ]	./.	5
MV [-R <sub>3</sub> ], r <sub>2</sub>	Br	3R    R <sub>3</sub> --> R <sub>3</sub> -2	./.	6
MV [R <sub>3</sub> +N], r <sub>2</sub>	Br	8R,N    r <sub>2</sub> --> [R <sub>3</sub> +1],[R <sub>3</sub> ]	./.	7
MV [R <sub>3</sub> -N], r <sub>2</sub>	Br	CR,N    r <sub>2</sub> --> [R <sub>3</sub> -N+1],[R <sub>3</sub> -N]	./.	7
MV [(*)], r <sub>2</sub>	**, B(8+r)	00,N    r <sub>2</sub> --> [(*)+1],[(*)]	./.	10/11
MV [(*)+N], r <sub>2</sub>	**, B(8+r)	80,M,N    r <sub>2</sub> --> [(*)+N+1],[(*)+N]	./.	12/13
MV [(*)-N], r <sub>2</sub>	**, B(8+r)	C0,M,N    r <sub>2</sub> --> [(*)-N+1],[(*)-N]	./.	12/13
<b>3-Byte Quelle:</b>				
Bei 3-Byte Quelle ist r <sub>3</sub> = S nicht zugelassen				
MV [R <sub>3</sub> ], r <sub>3</sub>	Br	OR    r <sub>3</sub> --> [R <sub>3</sub> +2],[R <sub>3</sub> +1],[R <sub>3</sub> ]	./.	6

Befehl	Code 1./2. Byte	Funktion weitere Byte	C/Z	Zy
MV [R <sub>3</sub> ++], r <sub>3</sub>	Br	2R    r <sub>3</sub> --> [R <sub>3</sub> +2],[R <sub>3</sub> +1],[R <sub>3</sub> ], ./.	./.	7
MV [-R <sub>3</sub> ], r <sub>3</sub>	Br	3R    R <sub>3</sub> --> R <sub>3</sub> -3, r <sub>3</sub> --> [R <sub>3</sub> +2],[R <sub>3</sub> +1],[R <sub>3</sub> ]	./.	8
MV [R <sub>3</sub> +N], r <sub>3</sub>	Br	8R,N    r <sub>3</sub> --> [R <sub>3</sub> +2],[R <sub>3</sub> +1],[R <sub>3</sub> ]	./.	8
MV [R <sub>3</sub> -N], r <sub>3</sub>	Br	CR,N    r <sub>3</sub> --> [R <sub>3</sub> -N+2],[R <sub>3</sub> -N+1],[R <sub>3</sub> -N]	./.	8
MV [(*)], r <sub>3</sub>	**, B(8+r)	00,N    r <sub>3</sub> --> [(*)+2],[(*)+1],[(*)]	./.11/12	
MV [(*)+N], r <sub>3</sub>	**, B(8+r)	80,M,N    r <sub>3</sub> --> [(*)+N+2],[(*)+N+1],[(*)+N]	./.13/14	
MV [(*)-N], r <sub>3</sub>	**, B(8+r)	C0,M,N    r <sub>3</sub> --> [(*)-N+2],[(*)-N+1],[(*)-N]	./.13/14	

Transport: vom äußeren RAM (indirekt adressiert) in CPU Register

<b>1-Byte Ziel:</b>				bei IL ein Zyklus mehr
MV R <sub>1</sub> ,[r <sub>3</sub> ]	9R	Or    [r <sub>3</sub> ] --> R <sub>1</sub>	./.	4
MV R <sub>1</sub> ,[r <sub>3</sub> ++]	9R	2r    [r <sub>3</sub> ] --> R <sub>1</sub> , r <sub>3</sub> --> r <sub>3</sub> +1	./.	4
MV R <sub>1</sub> ,[-r <sub>3</sub> ]	9R	3r    r <sub>3</sub> --> r <sub>3</sub> -1, [r <sub>3</sub> ] --> R <sub>1</sub>	./.	5
MV R <sub>1</sub> ,[r <sub>3</sub> +n]	9R	8r,n    [r <sub>3</sub> +n] --> R <sub>1</sub>	./.	6
MV R <sub>1</sub> ,[r <sub>3</sub> -n]	9R	Cr,n    [r <sub>3</sub> -n] --> R <sub>1</sub>	./.	6
MV R <sub>1</sub> ,[(*)]	**, 9(8+R)	00,n    [(*)] --> R <sub>1</sub>	./.	9
MV R <sub>1</sub> ,[(*)+n]	**, 9(8+R)	80,m,n    [(*)+n] --> R <sub>1</sub>	./.11/12	
MV R <sub>1</sub> ,[(*)-n]	**, 9(8+R)	C0,m,n    [(*)-n] --> R <sub>1</sub>	./.11/12	
<b>2-Byte Ziel:</b>				
MV R <sub>2</sub> ,[r <sub>3</sub> ]	9R	0r    [r <sub>3</sub> +1],[r <sub>3</sub> ] --> R <sub>2</sub>	./.	5
MV R <sub>2</sub> ,[r <sub>3</sub> ++]	9R	2r    [r <sub>3</sub> +1],[r <sub>3</sub> ] --> R <sub>2</sub>	./.	5
MV R <sub>2</sub> ,[-r <sub>3</sub> ]	9R	3r    r <sub>3</sub> --> r <sub>3</sub> -2	./.	6
MV R <sub>2</sub> ,[r <sub>3</sub> +n]	9R	8r,n    [r <sub>3</sub> +1],[r <sub>3</sub> ] --> R <sub>2</sub>	./.	7
MV R <sub>2</sub> ,[r <sub>3</sub> -n]	9R	Cr,n    [r <sub>3</sub> -n+1],[r <sub>3</sub> -n] --> R <sub>2</sub>	./.	7
MV R <sub>2</sub> ,[(*)]	**, 9(8+R)	00,n    [(*)+1],[(*)] --> R <sub>2</sub>	./.10/11	
MV R <sub>2</sub> ,[(*)+n]	**, 9(8+R)	80,m,n    [(*)+n+1],[(*)+n] --> R <sub>2</sub>	./.12/13	
MV R <sub>2</sub> ,[(*)-n]	**, 9(8+R)	C0,m,n    [(*)-n+1],[(*)-n] --> R <sub>2</sub>	./.12/13	
<b>3-Byte Ziel:</b>				Bei 3-Byte Ziel ist R <sub>3</sub> = S nicht zugelassen
MV R <sub>3</sub> ,[r <sub>3</sub> ]	9R	0r    [r <sub>3</sub> +2],[r <sub>3</sub> +1],[r <sub>3</sub> ] --> R <sub>3</sub>	./.	6
MV R <sub>3</sub> ,[r <sub>3</sub> ++]	9R	2r    [r <sub>3</sub> +2],[r <sub>3</sub> +1],[r <sub>3</sub> ] --> R <sub>3</sub>	./.	7
MV R <sub>3</sub> ,[-r <sub>3</sub> ]	9R	3r    r <sub>3</sub> --> r <sub>3</sub> -3	./.	8
MV R <sub>3</sub> ,[r <sub>3</sub> +n]	9R	8r,n    [r <sub>3</sub> +2],[r <sub>3</sub> +1],[r <sub>3</sub> ] --> R <sub>3</sub>	./.	8
MV R <sub>3</sub> ,[r <sub>3</sub> -n]	9R	Cr,n    [r <sub>3</sub> -n+2],[r <sub>3</sub> -n+1],[r <sub>3</sub> -n] --> R <sub>3</sub>	./.	8
MV R <sub>3</sub> ,[(*)]	**, 9(8+R)	00,n    [(*)+2],[(*)+1],[(*)]	./.11/12	
MV R <sub>3</sub> ,[(*)+n]	**, 9(8+R)	80,m,n    [(*)+n+2],[(*)+n+1],	./.13/14	
MV R <sub>3</sub> ,[(*)-n]	**, 9(8+R)	C0,m,n    [(*)-n+2],[(*)-n+1],	./.13/14	

Transport zwischen den Registern des inneren RAM und dem äußeren RAM

Transport vom inneren RAM ins äußere RAM (direkt adressiert)

Befehl	Code 1./2.weitere Byte Byte	Funktion	C/Z	Zy
<u>1 Byte</u>				
MV [LMN],(*)	**D8 N,M,L,n	(*) --> [LMN]	./. 6/7	
<u>2 Byte</u>				
MVW [LMN],(*)	**D9 N,M,L,n	(*),(*+1) --> [LMN],[LMN+1]	./. 7/8	
<u>3 Byte</u>				
MVP [LMN],(*)	**DA N,M,L,n	(*),(*+1),(*+2) --> [LMN],[LMN+1],[LMN+2]	./. 8/9	
<u>Blocktransfer</u>				
I bestimmt die Länge des Blockes				
MVL [LMN],(*)	**DB N,M,L,n	(*)... --> [LMN]... Zy = 6/7 + 2*I		

Transport vom äußeren RAM (direkt adressiert) ins innere RAM

Befehl	Code 1./2.weitere Byte Byte	Funktion	C/Z	Zy
<u>1 Byte</u>				
MV (*),[lmn]	**D0 N,n,m,l	[lmn] --> (*)	./. 7/8	
<u>2 Byte</u>				
MVW (*),[lmn]	**D1 N,n,m,l	[lmn],[lmn+1] --> (*),(*+1)	./. 8/9	
<u>3 Byte</u>				
MVP (*),[lmn]	**D2 N,n,m,l	[lmn],[lmn+1],[lmn+2] --> (*),(*+1),(*+2)	./. 9/10	
<u>Blocktransfer</u>				
I bestimmt die Länge des Blockes				
MVL (*),[lmn]	**D3 N,n,m,l	[lmn]... --> (*)... Zy = 6/7 + 2*I		

Transport vom inneren RAM ins äußere RAM (indirekt adressiert)

Befehl	Code 1./2.weitere Byte Byte	Funktion	C/Z	Zy
<u>1 Byte</u>				
MV [R <sub>3</sub> ],(*)	**E8 0R <sub>3</sub> ,n	(*) --> [R <sub>3</sub> ]	./. 6/7	
MV [R <sub>3</sub> ++],(*)	**E8 2R <sub>3</sub> ,n	(*) --> [R <sub>3</sub> ]	./. 6/7	
<u>2 Byte</u>				
MV [--R <sub>3</sub> ],(*)	**E8 3R <sub>3</sub> ,n	R <sub>3</sub> --> R <sub>3</sub> +1 R <sub>3</sub> --> R <sub>3</sub> -1 (*) --> [R <sub>3</sub> ]	./. 7/8	
MV [R <sub>3</sub> +N],(*)	**E8 8R <sub>3</sub> ,n,N	(*) --> [R <sub>3</sub> +N]	./. 8/9	
MV [R <sub>3</sub> -N],(*)	**E8 CR <sub>3</sub> ,n,N	(*) --> [R <sub>3</sub> -N]	./. 8/9	
MV [(*)],(#)	*#F8 00,N,n	(#) --> [(*)]	./. 11/12	
MV [(*)+N],(#)	*#F8 80,M,n,N	(#) --> [(*)+N]	./. 13/14	
MV [(*)-N],(#)	*#F8 C0,M,n,N	(#) --> [(*)-N]	./. 13/14	
<u>3 Byte</u>				
MVW [R <sub>3</sub> ],(*)	**E9 0R <sub>3</sub> ,n	(*),(*+1) --> [R <sub>3</sub> ],[R <sub>3</sub> +1]	./. 7/8	
MVW [R <sub>3</sub> ++],(*)	**E9 2R <sub>3</sub> ,n	(*),(*+1) --> [R <sub>3</sub> ],[R <sub>3</sub> +1], R <sub>3</sub> --> R <sub>3</sub> +2	./. 7/8	
MVW [--R <sub>3</sub> ],(*)	**E9 3R <sub>3</sub> ,n	R <sub>3</sub> --> R <sub>3</sub> -2, (*),(*+1) --> [R <sub>3</sub> ],[R <sub>3</sub> +1]	./. 10/11	

Befehl	Code 1./2.weitere Byte Byte	Funktion	C/Z	Zy
MVW [R <sub>3</sub> +N],(*)	**E9 8R <sub>3</sub> ,n,N	(*),(*+1) --> [R <sub>3</sub> +N],[R <sub>3</sub> +N+1]	./. 9/10	
MVW [R <sub>3</sub> -N],(*)	**E9 CR <sub>3</sub> ,n,N	(*),(*+1) --> [R <sub>3</sub> -N],[R <sub>3</sub> -N+1]	./. 9/10	
MVW [(*)],(#)	*#F9 00,N,n	(#),(#+1) --> [(*)],[(*)+1]	./. 12/13	
MVW [(*)+N],(#)	*#F9 80,M,n,N	(#),(#+1) --> [(*)+N],[(*)+N+1]	./. 14/15	
MVW [(*)-N],(#)	*#F9 C0,M,n,N	(#),(#+1) --> [(*)-N],[(*)-N+1]	./. 14/15	
<u>3 Byte</u>				
MVP [R <sub>3</sub> ],(*)	**EA 0R <sub>3</sub> ,n	(*),(*+1),(*+2) --> [R <sub>3</sub> ],[R <sub>3</sub> +1],[R <sub>3</sub> +2]	./. 8/9	
MVP [R <sub>3</sub> ++],(*)	**EA 2R <sub>3</sub> ,n	(*),(*+1),(*+2) --> [R <sub>3</sub> ],[R <sub>3</sub> +1],[R <sub>3</sub> +2]	./. 9/10	
MVP [--R <sub>3</sub> ],(*)	**EA 3R <sub>3</sub> ,n	R <sub>3</sub> --> R <sub>3</sub> +3 R <sub>3</sub> --> R <sub>3</sub> -3, (*),(*+1),(*+2) --> [R <sub>3</sub> ],[R <sub>3</sub> +1],[R <sub>3</sub> +2]	./. 10/11	
MVP [R <sub>3</sub> +N],(*)	**EA 8R <sub>3</sub> ,n,N	(*),(*+1),(*+2) --> [R <sub>3</sub> +N],[R <sub>3</sub> +N+1],[R <sub>3</sub> +N+2]	./. 10/11	
MVP [R <sub>3</sub> -N],(*)	**EA CR <sub>3</sub> ,n,N	(*),(*+1),(*+2) --> [R <sub>3</sub> -N],[R <sub>3</sub> -N+1],[R <sub>3</sub> -N+2]	./. 10/11	
MVP [(*)],(#)	*#FA 00,N,n	(#),(#+1),(#+2) --> [(*)],[(*)+1],[(*)+2]	./. 13/14	
MVP [(*)+N],(#)	*#FA 80,M,n,N	(#),(#+1),(#+2) --> [(*)+N],[(*)+N+1],[(*)+N+2]	./. 15/16	
MVP [(*)-N],(#)	*#FA C0,M,n,N	(#),(#+1),(#+2) --> [(*)-N],[(*)-N+1],[(*)-N+2]	./. 15/16	
<u>Blocktransfer</u>				
I bestimmt die Länge des Blockes				

Befehl	Code 1./2.weitere Byte Byte	Funktion	C/Z	Zy
MVL [X],(BP+n)	Man benutze:	MVL [X+00],(BP+n)		
MVL [R <sub>3</sub> ++],(*)	**EB 2R <sub>3</sub> ,n	Funktion: (*) --> [R <sub>3</sub> ], R <sub>3</sub> --> R <sub>3</sub> +1, I --> I-1, bis I = 0000 (*+1) --> [R <sub>3</sub> ], R <sub>3</sub> --> R <sub>3</sub> +1, I --> I-1, bis I = 0000	./. 5/6+2*I	
MVL [--R <sub>3</sub> ],(*)	**EB 3R <sub>3</sub> ,n	Funktion: R <sub>3</sub> --> R <sub>3</sub> -1, (*) --> [R <sub>3</sub> ], I --> I-1, bis I = 0000 R <sub>3</sub> --> R <sub>3</sub> -1, (*+1) --> [R <sub>3</sub> ], I --> I-1, bis I = 0000	./. 7/8+2*I	
MVL [R <sub>3</sub> +N],(*)	*#F5E 8R <sub>3</sub> ,n,N	Funktion: (*) --> [R <sub>3</sub> +N], I --> I-1, bis I = 0000 (*+1) --> [R <sub>3</sub> +N+1], I --> I-1, bis I = 0000	./. 5/6+2*I	
MVL [R <sub>3</sub> -N],(*)	*#F5E CR <sub>3</sub> ,n,N	Funktion: (*) --> [R <sub>3</sub> -N], I --> I-1, bis I = 0000 (*+1) --> [R <sub>3</sub> -N+1], I --> I-1, bis I = 0000	./. 5/6+2*I	
MVL [(*)],(#)	*#FB 00,N,n	Funktion: (#) --> [(*)], I --> I-1, bis I = 0000 (#+1) --> [(*)+1], I --> I-1, bis I = 0000	./. 10/11+2*I	
MVL [(*)+N],(#)	*#FB 80,M,n,N	Funktion: (#) --> [(*)+N], I --> I-1, bis I = 0000 (#+1) --> [(*)+N+1], I --> I-1, bis I = 0000	./. 12/13+2*I	

Befehl	Code	C/Z	Zy
MVL [(*)-N], (#)	*#FB C0,M,n,N	./. 12/13+2*I	
Funktion:	(#) --> [(*)-N], I --> I-1, bis I = 0000		
	(#+1) --> [(*)-N+1], I --> I-1, bis I = 0000		

#### Transport vom äußeren RAM (indirekt adressiert) ins innere RAM

Befehl	Code	Funktion	C/Z	Zy
1_Byte				
MV (*),[r <sub>3</sub> ]	**EO 0r <sub>3</sub> ,N	[r <sub>3</sub> ] --> (*)	./. 6/7	
MV (*),[r <sub>3</sub> ++]	**EO 2r <sub>3</sub> ,N	[r <sub>3</sub> ] --> (*), r <sub>3</sub> --> r <sub>3</sub> +1	./. 6/7	
MV (*),[-r <sub>3</sub> ]	**EO 3r <sub>3</sub> ,N	r <sub>3</sub> --> r <sub>3</sub> -1, [r <sub>3</sub> ] --> (*)	./. 7/8	
MV (*),[r <sub>3</sub> +n]	**EO 8r <sub>3</sub> ,N,n	[r <sub>3</sub> +n] --> (*)	./. 8/9	
MV (*),[r <sub>3</sub> -n]	**EO Cr <sub>3</sub> ,N,n	[r <sub>3</sub> -n] --> (*)	./. 8/9	
MV (*),[(#)]	#FO 00,N,n	[(#)] --> (*)	./. 11/12	
MV (*),[(#+n)]	#FO 80,N,m,n	[(#+n)] --> (*)	./. 13/14	
MV (*),[(#)-n]	#FO C0,N,m,n	[(#)-N] --> (*)	./. 13/14	
2[Byte]				
MVW (*),[r <sub>3</sub> ]	**E1 0r <sub>3</sub> ,N	[r <sub>3</sub> ],[r <sub>3</sub> +1] --> (*),(*+1)	./. 7/8	
MVW (*),[r <sub>3</sub> ++]	**E1 2r <sub>3</sub> ,N	[r <sub>3</sub> ],[r <sub>3</sub> +1] --> (*),(*+1), r <sub>3</sub> --> r <sub>3</sub> +2	./. 7/8	
MVW (*),[--r <sub>3</sub> ]	**E1 3r <sub>3</sub> ,N	r <sub>3</sub> --> r <sub>3</sub> -2, [r <sub>3</sub> ],[r <sub>3</sub> +1] --> (*),(*+1)	./. 10/11	
MVW (*),[r <sub>3</sub> +n]	**E1 8r <sub>3</sub> ,N,n	[r <sub>3</sub> +n],[r <sub>3</sub> +n+1] --> (*),(*+1)	./. 9/10	
MVW (*),[r <sub>3</sub> -n]	**E1 Cr <sub>3</sub> ,N,n	[r <sub>3</sub> -n],[r <sub>3</sub> -n+1] --> (*),(*+1)	./. 9/10	
MVW (*),[(#)]	#F1 00,N,n	[(#)],[(#+1)] --> (*),(*+1)	./. 12/13	
MVW (*),[(#+n)]	#F1 80,N,m,n	[(#+N)],[(#+N+1)] --> (*),(*+1)	./. 14/15	
MVW (*),[(#)-n]	#F1 C0,N,m,n	[(#)-n],[(#)-n+1] --> (*),(*+1)	./. 14/15	
3[Byte]				
MVP (*),[r <sub>3</sub> ]	**E2 0r <sub>3</sub> ,N	[r <sub>3</sub> ],[r <sub>3</sub> +1],[r <sub>3</sub> +2] --> (*),(*+1),(*+2)	./. 8/9	
MVP (*),[r <sub>3</sub> ++]	**E2 2r <sub>3</sub> ,N	[r <sub>3</sub> ],[r <sub>3</sub> +1],[r <sub>3</sub> +2] --> (*),(*+1),(*+2)	./. 9/10	
MVP (*),[--r <sub>3</sub> ]	**E2 3r <sub>3</sub> ,n	r <sub>3</sub> --> r <sub>3</sub> +3 [r <sub>3</sub> ],[r <sub>3</sub> +1],[r <sub>3</sub> +2] --> (*),(*+1),(*+2)	./. 10/11	
MVP (*),[r <sub>3</sub> +n]	**E2 8r <sub>3</sub> NnnN	[r <sub>3</sub> +n],[r <sub>3</sub> +n+1],[r <sub>3</sub> +n+2] ./. 10/11		
MVP (*),[r <sub>3</sub> -n]	**E2 Cr <sub>3</sub> ,N,n	[r <sub>3</sub> -n],[r <sub>3</sub> -n+1],[r <sub>3</sub> -n+2] ./. 10/11		
MVP (*),[(#)]	#F2 00,N,n	[(#)],[(#+1)],[(#+2)] ./. 13/14		
MVP (*),[(#+n)]	#F2 80,N,m,n	[(#+n)],[(#+n+1)],[(#+n+2)] --> (*),(*+1),(*+2) ./. 15/16		
MVP (*),[(#)-n]	#F2 C0,N,m,n	[(#)-n],[(#)-n+1],[(#)-n+2] --> (*),(*+1),(*+2) ./. 15/16		

<u>Blocktransfer</u>			
I bestimmt die Länge des Blockes	Code	C/Z	Zy
Befehl	1./.weitere		
Byte Byte			
MVL (BP+N),[X]	Man benutze: MVL (BP+N),[X+00]	./. 5/6+2*I	
MVL (*),[r <sub>3</sub> ++]	**E3 2r <sub>3</sub> ,N	./. 5/6+2*I	
Funktion: [r <sub>3</sub> ] --> (*), r <sub>3</sub> --> r <sub>3</sub> +1, I --> I-1, bis I = 0000	[r <sub>3</sub> ] --> (*+1), r <sub>3</sub> --> r <sub>3</sub> +1, I --> I-1, bis I = 0000		
MVL (*),[--r <sub>3</sub> ]	**E3 3r <sub>3</sub> ,N	./. 7/8+2*I	
Funktion: r <sub>3</sub> --> r <sub>3</sub> -1, [r <sub>3</sub> ] --> (*), I --> I-1, bis I = 0000	r <sub>3</sub> --> r <sub>3</sub> -1, [r <sub>3</sub> ] --> (*+1), I --> I-1, bis I = 0000		
MVL (*),[r <sub>3</sub> +n]	**56 8r <sub>3</sub> ,N,n	./. 5/6+2*I	
Funktion: [r <sub>3</sub> +n] --> (*), I --> I-1, bis I = 0000	[r <sub>3</sub> +n+1] --> (*+1), I --> I-1, bis I = 0000		
MVL (*),[r <sub>3</sub> -n]	**56 Cr <sub>3</sub> ,N,n	./. 5/6+2*I	
Funktion: [r <sub>3</sub> -n] --> (*), I --> I-1, bis I = 0000	[r <sub>3</sub> -n+1] --> (*+1), I --> I-1, bis I = 0000		
MVL (*),[(#)]	*#F3 00,N,n	./. 10/11+2*I	
Funktion: [(#)] --> (*), I --> I-1, bis I = 0000	[(#+1)] --> (*+1), I --> I-1, bis I = 0000		
MVL (*),[(#+n)]	*#F3 80,N,m,n	./. 12/13+2*I	
Funktion: [(#+n)] --> (#), I --> I-1, bis I = 0000	[(#+n+1)] --> (#+1), I --> I-1, bis I = 0000		
MVL (*),[(#)-n]	*#F3 C0,N,m,n	./. 12/13+2*I	
Funktion: [(#)-n] --> (*), I --> I-1, bis I = 0000	[(#)-n+1] --> (*+1), I --> I-1, bis I = 0000		

#### Transport zwischen CPU-Registern und dem Stack

Befehl	Code	Funktion	C/Z	Zy
U-Stack:				
PUSHU				
PUSHU A	28	U --> U-1, A --> [U]	./. 3	
PUSHU IL	29	U --> U-1, IL --> [U]	./. 3	
PUSHU BA	2A	U --> U-1, B --> [U]	./. 4	
		U --> U-1, A --> [U]		
PUSHU I	2B	U --> U-1, IH --> [U]	./. 4	
		U --> U-1, IL --> [U]		
PUSHU X	2C	U --> U-1, Xhigh --> [U]	./. 5	
		U --> U-1, Xmedium --> [U]		
		U --> U-1, Xlow --> [U]		
PUSHU Y	2D	U --> U-1, Vhigh --> [U]	./. 5	
		U --> U-1, Vmedium --> [U]		
		U --> U-1, Vlow --> [U]		
PUSHU F	2E	U --> U-1, F --> [U]	./. 3	
PUSHU IMR	2F	U --> U-1, IMR --> [U]	./. 3	
POPU				
POPU A	38	[U] --> A, U --> U+1	./. 2	
POPU IL	39	[U] --> IL, U --> U+1	./. 3	
POPU BA	3A	[U] --> B, U --> U+1	./. 3	

Befehl	Code	Funktion	C/Z	Zy
POPUP I	3B	[U] --> IL, U --> U+1 [U] --> IH, U --> U+1	./. .	3
POPUP X	3C	[U] --> Xlow, U --> U+1 [U] --> Xmedium, U --> U+1 [U] --> Xhigh, U --> U+1	./. .	4
POPUP Y	3D	[U] --> Ylow, U --> U+1 [U] --> Ymedium, U --> U+1 [U] --> Yhigh, U --> U+1	./. .	4
POPUP F	3E	[U] --> F, U --> U+1	./. .	2
POPUP IMR	3F	[U] --> IMR, U --> U+1	./. .	2
S-Stack				
PUSHS F	4F	S --> S-1, F --> [S]	./. .	3
POPS F	5F	[S] --> F, S --> S+1	./. .	2
PUSHS Reg = MV [-S],Reg				
POPS Reg = MV Reg,[S++]				

#### Sprungbefehle

Befehl	Code	Funktion	C/Z	Zy
		1./2. weitere Byte		
		Byte		

#### Direkter far-Sprung:

JPF lm	03	n,m,l	l --> PS (Segmenreg.) 1.Hexstelle ohne Bedeutung mn --> PC	./. .	5
--------	----	-------	--	-------	---

#### Indirekte far-Sprünge:

JP r <sub>3</sub>	11	r <sub>3</sub>	[r <sub>3</sub> ] --> PS/PC	./. .	4
JP (*)	**10	n	(n) --> PS/PC	./. .	6/7

#### Unbedingter near-Sprung:

JP mn	02	n,m	mn --> PC	./. .	4
-------	----	-----	-----------	-------	---

#### Bedingte near Sprünge:

JPZ mn	14	n,m	mn --> PC, falls Z = 1 Z = 0, kein Sprung	./. .	4
JPNZ mn	15	n,m	mn --> PC, falls Z = 0 Z = 1, kein Sprung	./. .	4
JPC mn	16	n,m	mn --> PC, falls Cy = 1 Cy = 0, kein Sprung	./. .	4
JPNC mn	17	n,m	mn --> PC, falls Cy = 0 Cy = 1, kein Sprung	./. .	4

#### Unbedingte relative short-Sprünge:

JR +n	12	n	PC --> PC + n PC = PC hinter dem JR-Befehl	./. .	3
-------	----	---	---	-------	---

Befehl	Code	1. weitere Byte	weitere Byte	Funktion	C/Z	Zy
JR -n	13	n		PC --> PC - n PC = PC hinter dem JR-Befehl	./. .	3
<u>Bedingte relative short-Sprünge:</u>						
JRZ +n	18	n		PC --> PC+n, falls Z = 1 Z = 0, kein Sprung	./. .	3
JRZ -n	19	n		PC --> PC-n, falls Z = 1 Z = 0, kein Sprung	./. .	3
JRNZ +n	1A	n		PC --> PC+n, falls Z = 0 Z = 1, kein Sprung	./. .	3
JRNZ -n	1B	n		PC --> PC-n, falls Z = 0 Z = 1, kein Sprung	./. .	3
JRC +n	1C	n		PC --> PC+n, falls C = 1 C = 0, kein Sprung	./. .	3
JRC -n	1D	n		PC --> PC-n, falls C = 1 C = 0, kein Sprung	./. .	3
JRNC +n	1E	n		PC --> PC+n, falls C = 0 C = 1, kein Sprung	./. .	3
JRNC -n	1F	n		PC --> PC-n, falls C = 0 C = 1, kein Sprung	./. .	3

#### Unterprogramm-Befehle

##### Far-CALLE/RETF:

Befehl	Code	1./2. weitere Byte	weitere Byte	Funktion	C/Z	Zy
CALLF lm	05	n,m,l		S --> S-1, PS --> [S] S --> S-1, PChigh --> [S] S --> S-1, PClow --> [S] 1 --> PS (Segmenreg.) 1.Hexstelle ohne Bedeutung mn --> PC	./. .	8
RETF	07			[S] --> PClow, S --> S+1 [S] --> PChigh, S --> S+1 [S] --> PS, S --> S+1	./. .	5

##### Near-CALL/RET:

CALL mn	04	n,m		S --> S-1, PChigh --> [S] S --> S-1, PClow --> [S] mn --> PC	./. .	6
RET	06			[S] --> PClow, S --> S+1 [S] --> PChigh S --> S+1	./. .	4

### Additions/Subtraktions-Befehle

#### Addition/Subtraktion eines Immediate

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
ADD A,n	40 n	A + n --> A	C/Z	3
ADC A,n	50 n	A + n + C --> A	C/Z	3
SUB A,n	48 n	A - n --> A	C/Z	3
SBC A,n	58 n	A - n - C --> A	C/Z	3
ADD (*),n	**41 N,n	(*) + n --> (*)	C/Z	4/5
ADC (*),n	**51 N,n	(*) + n + C --> (*)	C/Z	4/5
SUB (*),n	**49 N,n	(*) - n --> (*)	C/Z	4/5
SBC (*),n	**59 N,n	(*) - n - C --> (*)	C/Z	4/5

#### Addition/Subtraktion zwischen dem Akkumulator und dem inneren RAM

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
ADD A,(*)	**42 n	A + (*) --> A	C/Z	4/5
ADC A,(*)	**52 n	A + (*) + C --> A	C/Z	4/5
SUB A,(*)	**4A n	A - (*) --> A	C/Z	4/5
SBC A,(*)	**5A n	A - (*) - C --> A	C/Z	4/5
ADD (*),A	**43 N	(*) + A --> (*)	C/Z	4/5
ADC (*),A	**53 N	(*) + A + C --> (*)	C/Z	4/5
SUB (*),A	**4B N	(*) - A --> (*)	C/Z	4/5
SBC (*),A	**5B N	(*) - A - C --> (*)	C/Z	4/5

#### Addition/Subtraktion zwischen CPU Registern

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
1. Argument: 1-Byte Register				
ADD A,A	46 00	A + A --> A	C/Z	3
SUB A,A	4E 00	A - A --> A	C/Z	3
ADD A,IL	46 01	A + IL --> A	C/Z	3
SUB A,IL	4E 01	A - IL --> A	C/Z	3
ADD IL,A	46 10	IL + A --> IL	C/Z	3
SUB IL,A	4E 10	IL - A --> IL	C/Z	3
ADD IL,IL	46 11	IL + IL --> IL	C/Z	3
SUB IL,IL	4E 11	IL - IL --> IL	C/Z	3
1. Argument: 2-Byte Register				
ADD R <sub>2</sub> ,r <sub>1</sub>	44 R <sub>2</sub> r <sub>1</sub>	R <sub>2</sub> + r <sub>1</sub> --> R <sub>2</sub>	C/Z	5
ADD R <sub>2</sub> ,r <sub>2</sub>	44 R <sub>2</sub> r <sub>2</sub>	R <sub>2</sub> + r <sub>2</sub> --> R <sub>2</sub>	C/Z	5
SUB R <sub>2</sub> ,r <sub>1</sub>	4C R <sub>2</sub> r <sub>1</sub>	R <sub>2</sub> - r <sub>1</sub> --> R <sub>2</sub>	C/Z	5
SUB R <sub>2</sub> ,r <sub>2</sub>	4C R <sub>2</sub> r <sub>2</sub>	R <sub>2</sub> - r <sub>2</sub> --> R <sub>2</sub>	C/Z	5
1. Argument: 3-Byte Register				
ADD R <sub>3</sub> ,r <sub>1</sub>	45 R <sub>3</sub> r <sub>1</sub>	R <sub>3</sub> + r <sub>1</sub> --> R <sub>3</sub>	C/Z	7
ADD R <sub>3</sub> ,r <sub>2</sub>	45 R <sub>3</sub> r <sub>2</sub>	R <sub>3</sub> + r <sub>2</sub> --> R <sub>3</sub>	C/Z	7
ADD R <sub>3</sub> ,r <sub>3</sub>	45 R <sub>3</sub> r <sub>3</sub>	R <sub>3</sub> + r <sub>3</sub> --> R <sub>3</sub>	C/Z	7
SUB R <sub>3</sub> ,r <sub>1</sub>	4D R <sub>3</sub> r <sub>1</sub>	R <sub>3</sub> - r <sub>1</sub> --> R <sub>3</sub>	C/Z	7
SUB R <sub>3</sub> ,r <sub>2</sub>	4D R <sub>3</sub> r <sub>2</sub>	R <sub>3</sub> - r <sub>2</sub> --> R <sub>3</sub>	C/Z	7
SUB R <sub>3</sub> ,r <sub>3</sub>	4D R <sub>3</sub> r <sub>3</sub>	R <sub>3</sub> - r <sub>3</sub> --> R <sub>3</sub>	C/Z	7

### Binäre Block-Addition/Subtraktion, Akkumulator zu/von einem Block im inneren RAM

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
ADCL (*),A	**55 N	(*+I-1),...,(*) + 00,...,RegA --> (*+I-1),...,(*)	C/Z	4/5+I
SBCL (*),A	**5D N	(*+I-1),...,(*) - 00,...,RegA --> (*+I-1),...,(*)	C/Z	4/5+I

### Binäre Block-Addition/Subtraktion eines Blockes im inneren RAM zu/von einem anderen Block im inneren RAM

ADCL (*),(#)	*#54 N,n	(*+I-1),...,(*) + (#+I-1),...,(#) --> (*+I-1),...,(#)	C/Z	5/6+2I
SBCL (*),(#)	*#5C N,n	(*+I-1),...,(*) - (#+I-1),...,(#) --> (*+I-1),...,(#)	C/Z	5/6+2I

### Dezimale Block-Addition/Subtraktion, Akkumulator zu/von einem Block im inneren RAM

DADL (*),A	30C5 N	(*-I+1),...,(*),BCD + 00,...,RegA,BCD --> (*-I+1),...,(*)	C/Z	4/5+I
DSBL (*),A	30D5 N	(*-I+1),...,(*),BCD - 00,...,RegA,BCD --> (*-I+1),...,(*)	C/Z	4/5+I

### Dezimale Block-Addition/Subtraktion von einem Block im inneren RAM zu/von einem anderen Block im inneren RAM

DADL (*),(#)	*#C4 N,n	(*-I+1),..,(*),BCD + (#-I+1),..,(#),BCD --> (*-I+1),..,(*)	C/Z	5/6+2I
DSBL (*),(#)	*#D4 N,n	(*-I+1),..,(*),BCD - (#-I+1),..,(#),BCD --> (*-I+1),..,(*)	C/Z	5/6+2I

### Pointer Addition Befehle

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
PMDF (N),A	3057 N	(N) + A --> (N)	/.	5
PMDF (BP+N),A	57 N	(BP+N) + A --> (BP+N)	/.	4
PMDF (PX+N),A	3457 N	(PX+N) + A --> (PX+N)	/.	5
PMDF (BP+PX),A	2457 00	(BP+PX) + A --> (BP+PX)	/.	5
PMDF (N),n	3047 N,n	(N) + n --> (N)	/.	5
PMDF (BP+N),n	47 N,n	(BP+N) + n --> (BP+N)	/.	4

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
PMDF (PX+N),n	3447	N,n (PX+N) + n --> (PX+N)	./. 5	
PMDF (BP+PX),n	2447	00,n (BP+PX) + n --> (BP+PX)	./. 5	

#### Inkrement/Dekrement Befehle

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
INC A	6C 00	A + 1 --> A	./Z 3	
DEC A	7C 00	A - 1 --> A	./Z 3	
INC IL	6C 01	IL + 1 --> IL	./Z 3	
DEC IL	7C 01	IL - 1 --> IL	./Z 3	
INC BA	6C 02	BA + 1 --> NA	./Z 3	
DEC BA	7C 02	BA - 1 --> NA	./Z 3	
INC I	6C 03	I + 1 --> I	./Z 3	
DEC I	7C 03	I - 1 --> I	./Z 3	
INC X	6C 04	X + 1 --> X	./Z 3	
DEC X	7C 04	X - 1 --> X	./Z 3	
INC Y	6C 05	Y + 1 --> Y	./Z 3	
DEC Y	7C 05	Y - 1 --> Y	./Z 3	
INC U	6C 06	U + 1 --> U	./Z 3	
DEC U	7C 06	U - 1 --> U	./Z 3	
INC S	6C 07	S + 1 --> S	./Z 3	
DEC S	7C 07	S - 1 --> S	./Z 3	
INC (*)	**6D n	(*) + 1 --> (*)	./Z 3/4	
DEC (*)	**7D n	(*) - 1 --> (*)	./Z 3/4	

#### Logische Befehle

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
AND A,n	70 n	A AND n --> A	./Z 3	
OR A,n	78 n	A OR n --> A	./Z 3	
XOR A,n	68 n	A XOR n --> A	./Z 3	
AND (*),n	**71 N,n	(*) AND n --> (*)	./Z 4/5	
OR (*),n	**79 N,n	(*) OR n --> (*)	./Z 4/5	
XOR (*),n	**69 N,n	(*) XOR n --> (*)	./Z 4/5	
AND [LMN],n	72 N,M,L,n	[LMN] AND n --> [LMN]	./Z 7	
OR [LMN],n	7A N,M,L,n	[LMN] OR n --> [LMN]	./Z 7	
XOR [LMN],n	6A N,M,L,n	[LMN] XOR n --> [LMN]	./Z 7	
AND (*),A	**73 N	(*) AND A --> (*)	./Z 4/5	
OR (*),A	**7B N	(*) OR A --> (*)	./Z 4/5	
XOR (*),A	**6B N	(*) XOR A --> (*)	./Z 4/5	
AND A,(*)	**77 n	A AND (*) --> A	./Z 4/5	
OR A,(*)	**7F n	A OR (*) --> A	./Z 4/5	
XOR A,(*)	**6F n	A XOR (*) --> A	./Z 4/5	
AND (*),#	**76 N,n	(*) AND (#) --> (*)	./Z 6/7	

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
OR (*),(#)	*#7E N,n	(*) OR (#) --> (*)	./Z 6/7	
XOR (*),(#)	*#6E N,n	(*) XOR (#) --> (*)	./Z 6/7	

#### Bit-Test Befehle

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
TEST A,n	64 n	A AND n --> Z	./Z 3	
TEST (*),n	**65 N,n	(*) AND n --> Z	./Z 4/5	
TEST [LMN],n	66 N,M,L,n	[LMN] AND n --> Z	./Z 6	
TEST (*),A	**67 N	(*) AND A --> Z	./Z 4/5	

#### Vergleich Befehle, Compare

##### Compare mit Immediate

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
CMP A,n	60 n	A - n --> C,Z	C/Z 3	
CMP (*),n	**61 N,n	(*) - n --> C,Z	C/Z 4/5	
CMP [LMN],n	62 N,M,L,n	[LMN] - n --> C,Z	C/Z 6	

##### Compare inneres RAM mit CPU Register

CMP (*),A	**63 N	(*) - A --> C,Z	C/Z 4/5
CMPW (*),r <sub>2</sub>	**D6 r <sub>2</sub> ,N	(*+1),(*) - r <sub>2</sub> --> C,Z	C/Z 7/8
CMPP (*),r <sub>3</sub>	**D7 r <sub>3</sub> ,N	(*+2),(*+1),(*) - r <sub>3</sub> --> C,Z	C/Z 9/10

##### Compare inneres RAM mit innerem RAM

CMP (*),(#)	*#B7 N,n	(*) - (#) --> C,Z	C/Z 6/7
CMPW (*),(#)	*#C6 N,n	(*+1),(*) - (#+1),(#) --> C,Z	C/Z 8/9
CMPP (*),(#)	*#C7 N,n	(*+2),(*+1),(*) - (#+2),(#+1),(#) --> C,Z	C/Z 10/11

#### Carry Flag Control Befehle

Befehl	Code	Funktion	C/Z	Zy
	1./2. weitere Byte			
SC	97	1 --> C	1/. 1	
RC	9F	0 --> C	0/. 1	

Swap- Rotate- und Shift-Befehle

Befehl	Code 1./2. Byte	Funktion	C/Z	Zy
SWAP A	EE	Bit 7 - 4 <--> Bit 3 - 0	.Z	3

Rotate Befehle

ROR A	E4	Bit 0 --> Cy, rotate right	C/Z	2
ROR (*)	**E5 n	Bit 0 --> Cy, rotate right	C/Z	3/4
ROL A	E6	Bit 7 --> Cy, rotate left	C/Z	2
ROL (*)	**E7 n	Bit 7 --> Cy, rotate left	C/Z	3/4

Bit-Shift Befehle

SHR A	F4	bit-shift r. through Carry	C/Z	2
SHR (*)	**F5 n	bit-shift r. through Carry	C/Z	3/4
SHL A	F6	bit-shift l. through Carry	C/Z	2
SHL (*)	**F7 n	bit-shift l. through Carry	C/Z	3/4

Digit-Shift Befehle

DSRL (*)	**FC n	digit shift right, block	C/Z	4/5+I
DSLL (*)	**EC n	digit shift left, block	C/Z	4/5+I

Reset und Interrupt Befehle

Befehl	Code 1./2. Byte	Funktion	C/Z	Zy
RESET	FF	[FFFFD] --> PClow [FFFFE] --> PChigh [FFFFF] --> PS	./.	6
INT	FE	S --> S-1, PS --> [S] S --> S-1, PChigh --> [S] S --> S-1, PClow --> [S] S --> S-1, F --> [S] S --> S-1, IMR --> [S] [FFFFA] --> PClow [FFFFB] --> PChigh [FFFFC] --> PS	./.	12
RETI	01	IMR --> [S], S --> S+1 F --> [S], S --> S+1 PClow --> [S], S --> S+1 PChigh --> [S], S --> S+1 PS --> [S], S --> S+1	./.	7

CPU Control Befehle

Befehl	Code 1./2. Byte	Funktion	C/Z	Zy
NOP	00	no operation	./.	1
WAIT	EF	1+I Zyklen warten	./.	1+I
TCL	CE	Timer zurücksetzen	./.	1
HALT	DE	Anhalten, sub-timer läuft weiter. On pin high, 1 pin(K0 to K7) high, sub timer überlauf oder IRQ bei Bit 7 von FD = 1 bewirken Ausführung des nächsten Befehls. Reset pin high bewirkt RESET. Anhalten.		
OFF	DF	On pin high, 1 pin(K0 to K7) high oder IRQ bei Bit 7 von FD = 1 bewirken Ausführung des nächsten Befehls. Reset pin high bewirkt RESET.		

### 13.0 Aufbau und Aufruf von Maschinenprogrammen

Ganz zum Anfang sei auf einen wichtigen Punkt hingewiesen, der am Ende dieses Abschnittes genauer ausgeführt wird:

Vor Rückkehr eines Maschinenprogrammes zum Basicinterpreter muß das Carry Flag zurückgesetzt sein, anderenfalls erscheint eine unsinnige Fehlermeldung oder der PC-E500 kommt außer Kontrolle und es muß die RESET Taste betätigt werden.

Maschinenprogramme können mit dem Basicbefehl CALL sowohl im RUN-Modus, also direkt von der Tastatur, als auch von einem Basicprogramm aus aufgerufen werden.

Zur Erstellung und zur Speicherung von Maschinenprogrammen verwendet man die Basicbefehle PEEK, POKE, CSAVEM und CLOADM. Mit PEEK kann man den Speicherinhalt direkt betrachten. POKE gestattet Zahlen (0 bis 255 dezimal) in Speicherplätze einzugeben. Mit CSAVEM bzw. CLOADM können Maschinenprogramme, genauer gesagt Inhalte von Speicherbereichen, auf Cassette gespeichert bzw. von Cassette ins RAM geladen werden. Da diese Befehle in der Bedienungsanleitung des PC-E500 nicht enthalten sind, werden sie im folgenden Abschnitt beschrieben.

Für die Programmierung in Maschinensprache benötigt man einen Speicherbereich, der durch die Basicprogrammierung oder andere Funktionen des PC-E500 nicht gestört wird. Diese Reservierung von Specherraum für die Programmierung von Maschinenprogrammen wird ebenfalls in einem der nächsten Abschnitte beschrieben.

Das Maschinenprogramm selbst wird aus in dem Kapitel über die Maschinensprache beschriebenen Maschinenbefehlen aufgebaut. Zur Rückkehr zum Basicinterpreter, sei es zur Tastatur oder zum aufrufenden Basicprogramm, sind zwei Maschinenbefehle am Ende des Maschinenprogrammes nötig. Dies sind die Befehle RC (Reset Carry, Code 9F) und RETF (Return Far, Code 07).

### 13.1 Basicbefehle zur Maschinenprogrammierung

Zur Eingabe von Maschinenprogrammen, ihren Aufruf aus einem Basicprogramm oder von der Tastatur und ihre Speicherung auf Cassette sind im Basicinterpreter die Befehle PEEK, POKE, CALL, CSAVEM und CLOADM vorhanden.

Auch die Basicbefehle SAVE und LOAD dürfen durch Anhängen von M in der gleichen Weise wie CSAVEM und CLOAD für das Sichern und Laden von Maschinenprogrammen nutzbar sein.

#### PEEK

PEEK numerischer Ausdruck  
Es wird der Inhalt der Speicherzelle aufgerufen, deren Adresse durch den numerischen Ausdruck gegeben ist.

#### POKE

POKE numerischer Ausdruck 1, numerischer Ausdruck 2, numerischer Ausdruck 3,.....  
Beginnend bei der Speicheradresse, die durch den numerischen Ausdruck 1 gegeben ist, werden die folgenden numerischen Ausdrücke in den Speicher gegeben.

#### CALL

CALL numerischer Ausdruck  
Es wird das Maschinenprogramm gestartet, dessen Anfangsadresse durch den numerischen Ausdruck gegeben ist. Zur Rückkehr zum Basicinterpreter vom Maschinenprogramm muß dieses mit den Befehlen RC (Reset Carry, Code 9F) und RETF (Return Far, Code 07) abgeschlossen sein.

#### CSAVEM

CSAVE M Textausdruck, numerischer Ausdruck 1, numerischer Ausdruck 2, numerischer Ausdruck 3  
Der Textausdruck ist nicht notwendig. Das Maschinenprogramm, dessen Anfangsadresse durch den numerischen Ausdruck 1 und dessen Endadresse durch den numerischen Ausdruck 2 gegeben ist, wird unter dem durch den Textausdruck gegebenen Namen auf Cassette gespeichert. Wird der numerische Ausdruck 3 angegeben, so wird nach CLOADM das Maschinenprogramm bei dieser Adresse gestartet, falls der RUN-Mode eingestellt ist.

#### CLOADM

CLOAD M Textausdruck, numerischer Ausdruck  
Der Textausdruck ist nicht notwendig. D.h., es kann ein Programm, auch wenn es unter einem Namen auf Cassette abgespeichert wurde, ohne Angabe des Namens wieder in den Computer eingelesen werden. Wird jedoch ein anderer Name angegeben, so wird das Programm nicht eingelesen. Der numerische Ausdruck ist nicht unbedingt notwendig. Soll das Maschinenprogramm in den gleichen Speicherbereich eingelesen werden, von dem aus es auf Cassette geschrieben wurde, braucht der numerische Ausdruck nicht angegeben wer-

den. Nur wenn in einen anderen Speicherbereich eingelesen werden soll, muß dessen Anfangsadresse angegeben werden.

Beispiele:

**CSAVEM &BF000,&BF3FF,&BF000**  
sichert ein im Bereich &BF000 bis &BF3FF stehendes Maschinenprogramm auf Cassette. Die letzte Angabe von &BF000 sorgt dafür, daß nach dem Laden des Programmes mit CLOADM von Cassette es unmittelbar gestartet wird.

**CLOADM**  
lädt das mit dem vorangehenden Befehl gespeicherte Maschinenprogramm nach &BF000 bis &BF3FF und startet es bei der Adresse &BF000 (entspricht CALL &BF000), falls der RUN-Mode eingeschaltet ist.

**CSAVEM "PRO1",&BF800,&BF8FF**  
sichert ein im Bereich &BF800 bis &BF8FF stehendes Maschinenprogramm unter dem Namen "PRO1" auf Cassette.

**SAVEM"E:MA",&BFA00,&BFAFF**  
sichert den Speicherbereich von &BFA00 bis &BFAFF unter dem Namen "MA" auf der Ramdisk E:.

**LOADM"E:MA",&BF900**  
lädt das mit dem vorangehenden Befehl gesicherte Programm in den Speicherbereich von &BF900 bis &BF9FF.

### 13.2 Bereitstellung von Speicherraum für Maschinenprogramme

Will man Maschinenprogramme entwickeln, so benötigt man einen Speicherbereich, der nicht durch die sonstige Benutzung des Pocketcomputers, z.B. die Basicprogrammierung, gestört wird. Diese Reservierung von Speicherplatz wird im folgenden beschrieben.

Durch die folgende Eingabe im RUN-Modus (Abschluß mit der Return-Taste) wird ein vor &BFC00 liegender Speicherbereich frei gemacht:

**POKE &BFE03,&1A,&FD,&0B,Y,X,&00:CALL &FFFFD8**

Der Beginn BEG des freigemachten Speicherplatzes wird durch X und Y beim Einpoken bestimmt. Es ist:  
 $BEG = &BFC00 - (&100*X + Y)$

Auch wird der Beginn BEG dieses frei gemachten Speicherbereiches auf dem Systemplatz &BFD1A/B/C gehalten. Es ist also:  
 $BEG = PEEK &BFD1A + &100*PEEK &BFD1B + &10000*PEEK &BFD1C$

Beispiele:

**POKE &BFE03,&1A,&FD,&0B,&00,&08,&00:CALL &FFFFD8**  
macht den Bereich &BF400 bis &BFBFF frei.  
Es ist: PEEK&BFD1A + &100\*PEEK&BFD1B + &10000\*PEEK&BFD1C = &BF400

**POKE &BFE03,&1A,&FD,&0B,&80,&0B,&00:CALL &FFFFD8**  
macht den Bereich &BF080 bis &BFBFF frei.  
Es ist: PEEK&BFD1A + &100\*PEEK&BFD1B + &10000\*PEEK&BFD1C = &BF080

**POKE &BFE03,&1A,&FD,&0B,&00,&10,&00:CALL &FFFFD8**  
macht den Bereich &BEC00 bis &BFBFF frei.  
Es ist: PEEK&BFD1A + &100\*PEEK&BFD1B + &10000\*PEEK&BFD1C = &BEC00

Durch  
**POKE &BFE03,&1A,&FD,&0B,&00,&00,&00:CALL &FFFFD8**  
wird wieder der nach ON/RESET und löschen des Speichers gegebene Zustand hergestellt. In &BFD1A/B/C befindet sich in diesem Ausgangszustand die Adresse &BFC00.

Dieses Freimachen eines Speicherbereiches muß nicht nach einem ON/RESET durchgeführt werden. Die POKE/CALL-Eingabe ist auch bei vorhandenem Basicprogramm möglich. Dieses wird entsprechend nach vorn geschoben, falls noch genügend Speicherplatz vorhanden ist, was sich mit FRE0 feststellen läßt.

Die Größe des Speicherbereiches, den man freimachen kann, dürfte von der vorhandenen RAM-Karte (MEM\$="B") abhängen. Möglicherweise kann auch die letzte &00 im POKE-Ausdruck durch eine Zahl Z ersetzt werden. Es ist anzunehmen, daß um Z\*10000 verschoben wird. Zum Austesten stand nicht genügend RAM zur Verfügung.

Befehls-Tabelle, low/low

		&00	&10	&20	&30	&40	&50	&60	&70
&0	NOP	JP (n)		PRE (m), (BP+n)	ADD A,n	ADC A,n	CMP A,n	AND A,n	
&1	RETI	JP r3	PRE (BP+m), (BP+PY)	PRE (m), (BP+PY)	ADD (m),n	ADC (m),n	CMP (m),n	AND (m),n	
&2	JP mn	JR +n	PRE (BP+m), (n)	PRE (m), (n)	ADD A,(n)	ADC A,(n)	CMP [klm],n	AND [klm],n	
&3	JPF lmn	JR -n	PRE (BP+m), (PY+n)	PRE (m), (PY+n)	ADD (n),A	ADC (n),A	CMP (n),A	AND (n),A	
&4	CALL mn	JPZ mn	PRE (BP+PX) (BP+n)	PRE (PX+m), (BP+n)	ADD r2,r1 r2 r2	ADCL (m),(n)	TEST A,n	MV A,B	
&5	CALLF lmn	JPNZ mn	PRE (BP+PX) (BP+PY)	PRE (PX+m), (BP+PY)	ADD r3,r	ADCL (n),A	TEST (m),n	MV B,A	
&6	RET	JPC mn	PRE (BP+PX) (n)	PRE (PX+m), (n)	ADD r1,r1	MVL (m),[X+n]	TEST [klm],n	AND (m),(n)	
&7	RETF	JPNC mn	PRE (BP+PX) (PY+n)	PRE (PX+m), (PY+n)	PMDF (m),n	PMDF (n),A	TEST (n),A	AND A,(n)	

r = CPU-Register: A, IL, BA, I, X, Y, U, S  
 r<sub>1</sub> = Ein-Byte-Register: A, IL  
 r<sub>2</sub> = Zwei-Byte-Register: BA, I  
 r<sub>3</sub> = Drei-Byte-Register: X, Y, U, S  
 ( ) = Inhalt einer Zelle des internen RAM  
 [ ] = Inhalt einer Zelle des äußeren RAM  
 [X+n] = [r<sub>3</sub>+n],[r<sub>3</sub>-n]  
 (n) = (BP+n), wenn kein Pre-Byte

Befehls-Tabelle, high/low

		&00	&10	&20	&30	&40	&50	&60	&70
&8	MV A,n	JRZ +n	PUSHU A	POPUP A	SUB A,n	SBC A,n	XOR A,n	OR A,n	
&9	MV IL,n	JRZ -n	PUSHU IL	POPUP IL	SUB (m),n	SBC (m),n	XOR (m),n	OR (m),n	
&A	MV BA,mn	JRNZ +n	PUSHU BA	POPUP BA	SUB A,(n)	SBC A,(n)	XOR [klm],n	OR [klm],n	
&B	MV I,mn	JRNZ -n	PUSHU I	POPUP I	SUB (n),A	SBC (n),A	XOR (n),A	OR (n),A	
&C	MV X,lmn	JRC +n	PUSHU X	POPUP X	SUB r2,r1 r2 r2	SBCL (m),(n)	INC r	DEC r	
&D	MV Y,lmn	JRC -n	PUSHU Y	POPUP Y	SUB r3,r	SBCL (n),A	INC (n)	DEC (n)	
&E	MV U,lmn	JRNC +n	PUSHU F	POPUP F	SUB r1,r1	MVL [X+n], (m)	XOR (m),(n)	OR (m),(n)	
&F	MV S,lmn	JRNC -n	PUSHU IMR	POPUP IMR	PUSHF F	POPS F	XOR A,(n)	OR A,(n)	

r = CPU-Register: A, IL, BA, I, X, Y, U, S  
 r<sub>1</sub> = Ein-Byte-Register: A, IL  
 r<sub>2</sub> = Zwei-Byte-Register: BA, I  
 r<sub>3</sub> = Drei-Byte-Register: X, Y, U, S  
 ( ) = Inhalt einer Zelle des internen RAM  
 [ ] = Inhalt einer Zelle des äußeren RAM  
 [X+n] = [r<sub>3</sub>+n],[r<sub>3</sub>-n]  
 (n) = (BP+n), wenn kein Pre-Byte

Befehls-Tabelle, low/high

	&80	&90	&AO	&B0	&CO	&DO	&EO	&FO
&0	MV A,(n)	MV A,[r3]	MV (n),A	MV [r3],A	EX (m),(n)	MV (k),[lmn]	MV (m),[r3]	MV ((n))
&1	MV IL,(n)	MV IL,[r3]	MV (n),IL	MV [r3],IL	EXW (m),(n)	MVW (k),[lmn]	MVW (m),[r3]	MVW ((n))
&2	MV BA,(n)	MV BA,[r3]	MV (n),BA	MV [r3],BA	EXP (m),(n)	MVP (k),[lmn]	MVP (m),[r3]	MVP ((n))
&3	MV I,(n)	MV I,[r3]	MV (n),I	MV [r3],I	EXL (m),(n)	MVL (k),[lmn]	MVL (m),[r3++]	MVL ((n))
&4	MV X,(n)	MV X,[r3]	MV (n),X	MV [r3],X	DADL (m),(n)	DSBL (m),(n)	ROR A	SHR A
&5	MV Y,(n)	MV Y,[r3]	MV (n),Y	MV [r3],Y	DADL (n),A	DSBL (n),A	ROR (n)	SHR (n)
&6	MV U,(n)	MV U,[r3]	MV (n),U	MV [r3],U	CMPW (m),(n)	CMPW (n),r2	ROL A	SHL A
&7	MV S,(n)	SG	MV (n),S	CMP (m),(n)	CMPP (m),(n)	CMPP (n),r3	ROL (n)	SHL (n)

r = CPU-Register: A, IL, BA, I, X, Y, U, S  
 r<sub>1</sub> = Ein-Byte-Register: A, IL  
 r<sub>2</sub> = Zwei-Byte-Register: BA, I  
 r<sub>3</sub> = Drei-Byte-Register: X, Y, U, S  
 ( ) = Inhalt einer Zelle des internen RAM  
 [ ] = Inhalt einer Zelle des äußeren RAM  
 [r<sub>3</sub>] = [r<sub>3</sub>],[r<sub>3</sub>++],[-r<sub>3</sub>],[r<sub>3</sub>+n],[r<sub>3</sub>-n]  
 [r<sub>3</sub>++] = [r<sub>3</sub>++],[-r<sub>3</sub>],[r<sub>3</sub>+n],[r<sub>3</sub>-n]  
 [(n)] = [(n)],[(m)+n],[(m)-n]  
 (n) = (BP+n), wenn kein Pre-Byte

Befehls-Tabelle, high/high

	&80	&90	&AO	&B0	&CO	&DO	&EO	&FO
&8	MV A,[lmn]	MV A,[(n)]	MV [lmn],A	MV [(n)],A	MV (m),(n)	MV [klm],(n)	MV [r3],(n)	MV [(m)],(n)
&9	MV IL,[lmn]	MV IL,[(n)]	MV [lmn],IL	MV [(n)],IL	MVW (m),(n)	MVW [klm],(n)	MVW [r3],(n)	MVW [(m)],(n)
&A	MV BA,[lmn]	MV BA,[(n)]	MV [lmn],BA	MV [(n)],BA	MVP (m),(n)	MVP [klm],(n)	MVP [r3],(n)	MVP [(m)],(n)
&B	MV I,[lmn]	MV I,[(n)]	MV [lmn],I	MV [(n)],I	MVL (m),(n)	MVL [klm],(n)	MVL [r3++],(n)	MVL [(m)],(n)
&C	MV X,[lmn]	MV X,[(n)]	MV [lmn],X	MV [(n)],X	MV (m),n	MVP (k),lmn	DSLL (n)	DSRL (n)
&D	MV Y,[lmn]	MV Y,[(n)]	MV [lmn],Y	MV [(n)],Y	MVW (1),mn	EX A,B	EX r2,r2 r3,r3	MV r2,r2 r3,r3
&E	MV U,[lmn]	MV U,[(n)]	MV [lmn],U	MV [(n)],U	TCL	HALT	SWAP A	INT
&F	MV S,[lmn]	RC	MV [lmn],S		MVLD (m),(n)	OFF	WAIT	RESET

r = CPU-Register: A, IL, BA, I, X, Y, U, S  
 r<sub>1</sub> = Ein-Byte-Register: A, IL  
 r<sub>2</sub> = Zwei-Byte-Register: BA, I  
 r<sub>3</sub> = Drei-Byte-Register: X, Y, U, S  
 ( ) = Inhalt einer Zelle des internen RAM  
 [ ] = Inhalt einer Zelle des äußeren RAM  
 [r<sub>3</sub>] = [r<sub>3</sub>],[r<sub>3</sub>++],[-r<sub>3</sub>],[r<sub>3</sub>+n],[r<sub>3</sub>-n]  
 [r<sub>3</sub>++] = [r<sub>3</sub>++],[-r<sub>3</sub>],[r<sub>3</sub>+n],[r<sub>3</sub>-n]  
 [(n)] = [(n)],[(m)+n],[(m)-n]  
 (n) = (BP+n), wenn kein Pre-Byte

**S Stichwortverzeichnis**

Abkürzungen	A-1
Abkürzungen für CPU Register	12.4-1
Additionsbefehle	12.1-1, 12.4-12
Addressierung	12.3-1
Addressierung, direkte	12.3-1
Addressierung, indirekte	12.3-1
Addressierungsarten im äußeren RAM	12.3-1, 12.4-1
Addressierungsarten im inneren RAM	12.0-1, 12.2-1
Adressraum	E-1, 8.0-1, 8.2-1, 9.0-1
AER-Modus	6.0-1
AER-Funktion	12.0-1, 12.1-1, 12.2-2, 12.3-1
äußeres RAM	12.0-1, 12.0-2
Akkumulator	12.1-1
Anzeige-Maschinenroutinen	12.0-1
Architektur der CPU	12.0-1
Arithmetische Funktionen	1.0-1
Audio cassette	6.1-1
Aufbau des Basicprogramm	13.0-1
Aufbau von Maschinenprogrammen	13.0-1
Aufruf von Maschinenprogrammen	13.0-1
äußeres RAM	12.0-1, 12.1-1, 12.2-1, 12.3-1
ASCII-Code	2.0-2
AUTO	10.0-1
Basicbefehl	6.1-1
Basicbefehle zur Maschinenprogrammierung	13.1-1
Basic-Modus	6.1-1
Basicprogramm	E-1, 6.0-1, 6.1-1, 9.0-1
Basicprogramm, Organisation im RAM	6.7-1
BATT	10.0-2
BCD	2.0-2
Befehlssatz	12.1-1
Befehlstabelle, hex	14.0-1
Beginn des Basicprogrammes	6.1-1
binäre Blockaddition	12.4-13
binäre Blocksubtraktion	12.4-13
binary coded decimals	2.0-2
Bit	2.0-2
Bit-Shift Befehle	12.4-16
Bit-Test Befehle	12.1-1, 12.4-15
Blockaddition, binär	12.4-13
Blockaddition, dezimal	12.4-13
Blocksubtraktion, binär	12.4-13
Blocksubtraktion, dezimal	12.4-13
Blocktransfer	12.1-1, 12.4-7
Blocktransport	12.4-3
Blocktransport, dekrementierend	12.4-3
Blocktransport, inkrementierend	12.4-3
borrow	12.0-1
Breakpointmonitor	12.1-1
Byte	2.0-2
CALL	6.6-1, 13.0-1, 13.1-1
CAPS	10.0-2
Carry Flag	12.0-1, 13.0-1
Carry Flag Control Befehle	12.1-1, 12.4-15
Cassette	13.0-1

Cassetteninterface	1.0-1
Cassettenrekorder	1.0-1
CE-124	1.0-1
CE-126P	1.0-1
CE-130T	1.0-2
CE-140F	1.0-1
CE-212M	1.0-1
CE-2H16M	1.0-1
CE-2H32M	1.0-1
CE-2H64M	1.0-1
CE-515P	1.0-1
CLOAD M	13.0-1, 13.1-1
CMOS	12.0-1
Codierung, doppelt genaue Zahlen	5.1-1
Codierung, einfache genaue Zahlen	5.1-1
Codierung von Zahlen im Basicprogramm	6.2-1
Compare	12.1-1, 12.4-15
CONSOLE	10.0-2
Counter	12.0-1
CPU	E-1, 12.0-1
CPU Control Befehle	12.1-1, 12.4-17
CPU-Register	12.0-1
CPU-Register, Abkürzungen	12.4-1
CSAVE M	13.0-1, 13.1-1
Daten	7.1-7
Datenbereiche im RAM	8.0-1
DEG	10.0-1
Dekrement Befehle	12.1-1, 12.4-14
dekrementierender Blocktransport	12.4-3
dezimale Blockaddition	12.4-13
dezimale Blocksubtraktion	12.4-13
Dezimalsystem	2.0-1
Digit-Shift Befehle	12.4-16
Directory	7.1-6
direkte Addressierung	12.3-1
Disassembler	12.1-1
Diskettenlaufwerk	1.0-1
Display Bereich	12.2-1
Drucker	1.0-1
Dualsystem	2.0-1
Dumpprogramm	E-1, 3.0-1
doppelte Genauigkeit	5.0-1
doppelt genaue Zahlen	5.1-1
Drucker-Maschinenroutinen	12.1-1
einfache Genauigkeit	5.0-1
einfache Variable	5.0-1
einfach genaue Zahlen	5.1-1, 5.2-1
eingebautes RAM	1.0-1, 4.0-1, 6.0-1
Ende des Basicprogrammes	6.1-1, 6.7-1
ENG-Modus	E-1, 6.0-1, 8.0-1, 8.3-1, 9.0-1
Erweiterungen	1.0-1
EVAL	6.6-1
Exponent	5.1-1
far	12.3-1
Farbplotter	1.0-1
FAT	7.1-3
Felder	5.0-1

Felder für Zeichenvariable	5.8-1
Feldvariable	6.0-1
Feldvar., numerische, einf. Genauigkeit	5.6-1
Feldvar., numerische, dopp. Genauigkeit	5.7-1
File Allocation Table	7.1-3
Flag	12.0-1
Flagregister	12.0-2
FRE0	6.0-1, 6.6-1, 9.0-1
Funktionstasten	E-1, 6.0-1, 8.1-1, 9.0-1
GRAD	10.0-1
Hauptregister	12.0-2
Hexadezimalsystem	2.0-1
Hexzahlen	A-1
Hilfsakkumulator	12.0-1, 12.0-2
Immediate	12.1-1
Immediate Befehle	12.4-1
indirekte Adressierung	12.3-1
Inhaltsverzeichnis	7.1-6
INIT	7.0-1, 7.1-1
Inkrement Befehle	12.1-1, 12.4-14
inkrementierender Blocktransport	12.4-3
inneres RAM	12.0-1, 12.0-2, 12.1-1, 12.2-1, 12.3-1
Interrupt Befehl	12.1-1, 12.4-16
Interrupt Vektor	12.2-1
I-Register	12.0-2
Kennbyte	6.2-1
KILL	7.1-9
Kopf der Ramdisk	7.1-1
Kopf des AER-Bereiches	8.2-1
Kopf des Basicprogrammes	6.1-1, 6.7-1
Kopf des ENG-Bereiches	8.3-1
Kopf des Funktionstastenbereiches	8.1-1
Kopf des Variablenbereiches	5.9-1, 6.1-1
Logische Befehle	12.1-1, 12.4-14
Logische Funktionen	12.0-1
Maschinenprogramme, Aufbau	13.0-1
Maschinenprogramme, Aufruf	13.0-1
Maschinenprogramme, Speicher für	13.0-1, 13.2-1
Maschinenprogrammierung, Basicbefehle zur	13.1-1
Maschinensprache	E-1
Maschinensprachehandbuch	12.1-1
Maschinenspracheprogramm	9.0-1
Matrix-Modus	8.0-1, 8.4-1
Mehrbytebefehle	12.1-1
MEM\$	1.0-1, 4.0-1, 7.0-1, 9.0-1
MEM\$ = "B"	1.0-1, 4.0-1, 7.0-1, 9.0-2
MEM\$ = "S1"	1.0-1, 4.0-1, 7.0-1, 7.1-1, 9.0-1
MEM\$ = "S2"	1.0-1, 7.0-1, 9.0-2
Mikroprozessor	12.0-1
NAME	6.6-1
near	12.3-1
numerische Feldvar., einf. Genauigkeit	5.6-1
numerische Feldvar., dopp. Genauigkeit	5.7-1
numerische Standardvariable	5.2-1
numerische Variable	5.0-1
numerische Variable, einf. Genauigkeit	5.3-1
numerische Variable, dopp. Genauigkeit	5.4-1

Oktalsystem	2.0-1
Organisation der Variablen im RAM	5.9-1
Organisation der Basicprogramm im RAM	6.7-1
Passwort	6.7-1
Pagesegment Register	12.0-2
PEEK	6.6-1, 13.0-1, 13.1-1
Pegelkonverter	1.0-2
Pointer	12.0-1, 12.0-2
Pointer Additions Befehle	12.1-1, 12.4-13
Pointer für inneres RAM	12.0-2
POKE	6.6-1, 13.0-1, 13.1-1
PRINT	10.0-2
Programmcounter	12.0-1, 12.0-2
RAD	10.0-1
RAM	2.0-2
RAM, äußeres	12.0-1, 12.1-1, 12.2-1, 12.3-1
RAM-Diskette	E-1, 1.0-1, 7.0-1
RAM-Diskette E	1.0-1, 7.2-1, 9.0-1, 9.0-2
RAM-Diskette F	1.0-1, 7.1-1, 9.0-2
RAM-Diskette G	7.3-1
RAM, inneres	12.0-1, 12.0-2, 12.1-1, 12.2-1, 12.3-1
RAM-Karte	1.0-1, 4.0-1, 7.0-1
Register	12.0-1
Register, 1-Bit	12.0-1
Register, 8-Bit	12.0-1
Register, 16 Bit	12.0-1
Register, 20 Bit	12.0-1
Remote-Steuerung	1.0-1
Reset Befehl	12.1-1, 12.4-16
Reset Vektor	12.2-1
ROM	2.0-2
Rotate Befehle	12.1-1, 12.4-16
SAVE	7.1-9
Schaltbild	E-1
Schnittstelle, serielle	1.0-1
Segmentregister	12.0-1, 12.0-2
serielle Schnittstelle	1.0-1
Service Manual	E-1
Shift Befehle	12.1-1, 12.4-16
short	12.3-1
Sonderanzeigeelemente	10.0-1, 10.0-2
Speichereinteilung	E-1, 4.0-1
Speicher für Maschinenprogramme	13.0-1, 13.2-1
Speicherkonfiguration	1.0-1
Sprungbefehle	12.1-1, 12.4-10
S-Register	12.0-2
Stack	12.0-1, 12.4-9
Stackpointer	12.0-1, 12.0-2
Stackpointer, Daten	12.0-1
Stackpointer, CALL	12.0-1
Standardvariable	5.0-1, 5.2-1
Standardvariable, numerische	5.2-1
STAT-Modus	8.0-1, 8.4-1
Subtraktionsbefehle	12.1-1, 12.4-12
Swap Befehl	12.1-1, 12.4-16
Systemadressen	E-1, 9.0-1, 10.0-1
Systembereich	E-1, 9.0-1, 10.0-1

Tastatur-Maschinenroutinen  
 Text-Modus  
 Textvariable  
 Thermodrucker  
 Token  
 Tokentabelle  
 Tokentabelle, alphabetisch  
 Tokentabelle, hex  
 Transfile  
 Transportbefehle  
 Überlauf  
 Unbekannte Basicbefehle  
 Unterprogrammbefehle  
 U-Register  
 Variable  
 Variable, numerische  
 Variable, numerische, einf. Genauigkeit  
 Variable, numerische, dopp. Genauigkeit  
 Variable, Organisation im RAM  
 Vergleich Befehle  
 Vierfarbendrucker  
 Vorcode  
 WAIT  
 X-Register  
 Y-Register  
 Zähler  
 Zahlensystem  
 Zeichensätze  
 Zeichensatz, 1.  
 Zeichensatz, 2.  
 Zeichenvariable  
 Zeilennummer  
 Zero Flag  
 Zusammenspiel der Datenbereiche  
 Zusatzgeräte  
 Zykluszahl  
 Zykluszeit

12.1-1  
 6.1-1  
 5.0-1, 5.5-1  
 1.0-1  
 6.1-1, 6.3-1  
 6.3-1  
 6.3-1  
 6.4-1, 6.5-1  
 1.0-1  
 12.1-1, 12.4-1  
 12.0-1  
 6.6-1  
 12.4-11  
 12.0-2  
 E-1, 1.0-1, 5.0-1, 6.0-1, 9.0-1  
 5.0-1  
 5.3-1  
 5.4-1  
 5.9-1  
 12.1-1, 12.4-15  
 1.0-1  
 12.4-1  
 10.0-1, 12.0-1  
 12.0-2  
 12.0-2  
 12.0-1  
 2.0-1  
 E-1, 10.0-1, 11.0-1  
 11.0-2  
 11.0-4  
 5.0-1, 5.5-1, 5.8-1  
 6.1-1  
 12.0-1  
 9.0-1  
 1.0-1  
 12.4-1  
 12.0-1

## Pocket + Laptop Computer

### Abonnement

Wenn es Ihnen Spaß gemacht hat, diese Ausgabe von "Pocket + Laptop Computer" zu lesen, und Sie sich auch in Zukunft durch unsere interessante Zeitschrift über alles Wissenswerte zum Thema Pocket und Laptop Computer informieren wollen, dann sollten Sie nicht länger zögern, "Pocket + Laptop Computer" jetzt im regelmäßigen Bezug per Post zu bestellen. Sicher Sie sich eine lückenlose Information und schicken Sie Ihre "Pocket + Laptop Computer" kommt dann regelmäßig jeden Monat ins Haus, ohne daß Ihnen zusätzliche Kosten entstehen.

Ja, ich möchte Abonnent und Club-Mitglied werden!

### Sichern Sie sich Ihr Gratis-Überraschungsbuch

Nutzen Sie jetzt Ihre Vorteile eines persönlichen Abonnements:

Ich bestelle folgende schon erschienene Exemplare von "Pocket + Laptop Computer", bzw. "Pocket Computer" (Stückpreis 5,- DM, Ausland 6,- DM):

Heft-Nr.:

Verkauf erfolgt solange Vorrat reicht!

Der Gesamtpreis von \_\_\_\_\_ DM  
 liegt bar bei \_\_\_\_\_

liegt als Vorauszahlungsscheck bei \_\_\_\_\_

(schlafe Einzelne)

wurde am \_\_\_\_\_ auf das Postgirokonto der Fischel GmbH, Kantonstr. 461533103 BLZ. 10010010, Postgarten Berlin überwiesen.

liegt (nur bei kleineren Beträgen) in Brüdermarken oder internationalen Antworten bei.

Alle Preise inkl. 7% Mwst.

Name, Vorname \_\_\_\_\_ Bitte einsenden an:

Strasse, Nr. \_\_\_\_\_ Fische GmbH

PLZ/Ort \_\_\_\_\_ Kaiser-Friedrich-Str. 54a

D-1000 Berlin 12

1. Datum, Unterschrift \_\_\_\_\_

2. Datum, Unterschrift \_\_\_\_\_

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestellstelle rückgängig machen kann. Zur Wahrung der Pünktlichkeit der Abrechnung Ich bestätige dies durch meine zweite Unterschrift.

### Sehr geehrter Software Lieferant!

Für eine Zusammenarbeit gibt es drei Möglichkeiten:  
 a) Lieferung, Fachberatung und Programme auf Druck oder Kassette,  
 b) Sendungen nur beschreibbar, wenn eine unterschriebene Bestellbestätigung benötigt,  
 c) Anfragen

### Zu a) Überlassung

1) Die Erwerbte erkennt, daß sie keine Berichtigung bei den Unbedenklichen Differenzen, 2) Programme und Lizenzen und auf Datenträger (Platte oder Kassette) entzerrt. Eine endgültige Beschreibung mit den 3) Durchdringlich Übermittlung und auf Datenträger (Platte oder Kassette) entzerrt. Eine endgültige Beschreibung mit den 4) Die Eisenkarte gilt als Zustimmung zum Abschluß. Eine gesonderten Zustimmung des Verkäufers besteht es nicht. 5) Mit der Annahme der Berichtigung wird der Vertrag vom Übereinkommen sämtliche Pläne, einschließlich der Vereinbarung zur Gewährung einer Gratifikation nach Erreichung des Vertrags. Der Contrahent kann Gratifikationsansprüche erheben. 6) Der Übereinkommen erhält mit seiner Unterschrift diese Bedingungen der FISCHEL GmbH, Kaiser-Friedrich Str. 54a, 1000 Berlin 12, a) Der Übereinkommen erhält mit seiner Unterschrift diese Bedingungen der FISCHEL GmbH, Kaiser-Friedrich Str. 54a, 1000 Berlin 12, a)

Unterschrift des Erreichens

### Zu b) Provision

1) Beleg für die Bezahlung von 2) Nach Abschluß der Berichtigung erhalten die Fischel GmbH die Bezahlung des Betrag, Entgelts, 3) Für die Erreichung von Bezahlungen wird die Wiedergabe erhalten die Fischel GmbH die Bezahlung des Betrag, Entgelts, 4) Die Abrechnung der Produkte erfolgt durch Rechnung, Prüfung des Bestellbelegs und der Rechnung, Prüfung des 5) Eine Vereinbarung kann zum Ende eines jeden Kalenderquartals mit einer Frist von einem Monat spätestens vorliegenden Quartals.

Unterschrift

### Zu c) Anzeigen in Pocket + Laptop Computer

1) Werden sie um eine unten befindlichen Anzeige mit ihrem Namen über den zu 2) Wenn sie eine Anzeige aufgeben wollen, senden sie uns die Kürzung nicht beschreiben. Wie behandeln wir die Fischel, ohne 3) Werden sie eine Anzeige aufgeben wollen, senden sie uns die Fischel, ohne 4) Alle Anzeigen werden dann natürlich umgehend zurückgesandt.

Unterschrift

Unterschrift des Anbieters

# Computer-Bücher • SUPER-BESTELLSCHEIN • Fischel GmbH

Herrn bestellt ich:

**Anzahl:** Buch:

**Atari**

PC Portfolio Anwendungshandbuch  
ISBN 3-89374-048-5, VK = 49,- DM  
Der PC Portfolio in Deiner Hand  
ISBN 3-89374-053-3, VK = 49,- DM  
PC Portfolio Systemhandbuch  
ISBN 3-89374-067-8, VK = 49,- DM

**Casio FX-850P/880P/PLB-1000/2000**

FX-850P/880P Anwendungshandbuch  
ISBN 3-89374-007-5, VK = 49,- DM

Der FX-850P/880P in Deiner Hand  
ISBN 3-89374-022-1, VK = 49,- DM

Die besten Programme für den FX-850P/880P  
ISBN 3-89374-057-0, VK = 49,- DM

FB-1000 Tips und Tricks Programmhandbuch  
ISBN 3-89374-044-4, VK = 49,- DM

FB-1000 Anwendungshandbuch  
ISBN 3-89374-022-9, VK = 49,- DM

FB-1000 Intens. dort ROM-Listung  
ISBN 3-89374-029-7, VK = 59,- DM

FB-1000 Power-Software  
ISBN 3-89374-044-9, VK = 49,- DM

FB-1000 Mett-Power-Software, Band 2  
ISBN 3-89374-051-6, VK = 49,- DM

FB-1000 Systemhandbuch  
ISBN 3-89374-047-3, VK = 49,- DM

FB-1000 Assembler  
ISBN 3-89374-059-2, VK = 69,- DM

FB-2000 Anwendungshandbuch  
ISBN 3-89374-042-2, VK = 49,- DM

**Hewlett Packard**

HP-200CS Anwendungsprogramme  
ISBN 3-89374-029-5, VK = 49,- DM

HP-200CS Programmierung  
ISBN 3-89374-041-4, VK = 49,- DM

HP-200CS Programmierhandbuch  
ISBN 3-89374-079-1, VK = 49,- DM

HP-200CS Testhandbuch  
ISBN 3-89374-054-4, VK = 49,- DM

HP-200CS Statistikprogramme  
VK = 39,- DM

HP-200CS Betriebslehre  
ISBN 3-89374-070-8, VK = 49,- DM

HP-2040 Höhere Mathematik Programmierung  
ISBN 3-89374-078-3, VK = 49,- DM

HP-2050 Praktikum handbuch  
ISBN 3-89374-069-1, VK = 49,- DM

HP-2050 Tips und Tricks Programmhandbuch  
ISBN 3-89374-078-7, VK = 49,- DM

**PSION**

PSION Organizer II Anwendungshandbuch  
ISBN 3-89374-040-1, VK = 49,- DM

PSION Organizer Programmierung  
ISBN 3-89374-073-2, VK = 49,- DM

**Sharp PC-1500/A/PC-1600**

PC-1500/A/PC-1600 Hardw.-handbuch  
ISBN 3-89374-130-8, VK = 49,- DM

PC-1500A TIPS und TRICKS  
ISBN 3-89374-122-2, VK = 49,- DM

PC-1600 Systemhandbuch  
ISBN 3-89374-021-9, VK = 49,- DM

PC-1600 Anwendungshandbuch  
ISBN 3-89374-056-8, VK = 49,- DM

PC-1600 MaschinenSprachHandbuch  
ISBN 3-89374-001-5, VK = 49,- DM

PC-1600 Tips und Tricks Programmhandbuch  
ISBN 3-89374-058-8, VK = 49,- DM

Die besten Programme für den PC-1600  
ISBN 3-89374-040-8, VK = 49,- DM

**Sharp PC-1350/60/2500**

PC-1350 Anwendungshandbuch  
ISBN 3-89374-157-1, VK = 49,- DM

PC-1350 MaschinenSprachHandbuch  
ISBN 3-89374-020-8, VK = 49,- DM

PC-1350 Systemhandbuch  
ISBN 3-89374-014-9, VK = 49,- DM

Die besten Programme für den PC-1350  
ISBN 3-89374-036-4, VK = 49,- DM

PC-2000 Systemhandbuch  
ISBN 3-89374-203-3, VK = 49,- DM

PC-2500 Anwendungshandbuch  
ISBN 3-89374-050-0, VK = 49,- DM

PC-2500 TIPS und TRICKS  
ISBN 3-89374-050-8, VK = 49,- DM

Die besten Programme für den PC-1350  
ISBN 3-89374-040-6, VK = 49,- DM

**Sharp PC-1402/02/03/21/50/75**

PC-1100/50/1245/46S/48/60/61/80

PC-1100 Systemhandbuch  
ISBN 3-89374-58-4, VK = 39,- DM

PC-1403 Anwendungshandbuch  
ISBN 3-89374-03-3, VK = 49,- DM

PC-1403 MaschinenSprachHandbuch  
ISBN 3-89374-73-4, VK = 49,- DM

Die besten Programme für den PC-1403  
ISBN 3-89374-003-2, VK = 49,- DM

PC-1404 Anwendungshandbuch  
ISBN 3-89374-18-1, VK = 49,- DM

PC-1401/02/21/03 TIPS und TRICKS Programmhandbuch  
ISBN 3-89374-32-3, VK = 49,- DM

PC-1401/02/21/03 MaschinenSprachProgrammHandbuch  
ISBN 3-89374-03-5, VK = 49,- DM

PC-1401/02/21/03 Chemie  
ISBN 3-89374-09-9, VK = 15,- DM

PC-1401/02/21/03 Physik  
ISBN 3-89374-45-9, VK = 39,- DM

PC-1401/02/21/03 Astronomie  
ISBN 3-89374-45-9, VK = 39,- DM

PC-1150/1245/1246/1270 Anwendungshandbuch  
ISBN 3-89374-005-8, VK = 49,- DM

PC-1200/01/02/03/04/05/06/07/08 Anwendungshandbuch  
ISBN 3-89374-007-7, VK = 49,- DM

PC-1200/175 Anwendungshandbuch  
ISBN 3-89374-054-4, VK = 49,- DM

PC-1200/175 Systemhandbuch  
ISBN 3-89374-053-3, VK = 49,- DM

PC-1200/175 MaschinenSprachHandbuch  
ISBN 3-89374-008-2, VK = 49,- DM

PC-1200/175 TIPS und TRICKS Programmhandbuch  
ISBN 3-89374-037-8, VK = 49,- DM

Der Sharp in Deiner Hand  
VK = 49,- DM

**Sharp PC-E220/500**

PC-E220 Programmierhandbuch  
ISBN 3-89374-060-3, VK = 49,- DM

PC-E220 MaschinenSprachHandbuch  
ISBN 3-89374-061-4, VK = 49,- DM

PC-E220 Systemhandbuch  
ISBN 3-89374-062-9, VK = 49,- DM

PC-E220 TIPS und TRICKS  
ISBN 3-89374-051-1, VK = 49,- DM

PC-E220 MaschinenSprachHandbuch  
ISBN 3-89374-053-3, VK = 49,- DM

PC-E220 Systemhandbuch  
ISBN 3-89374-054-8, VK = 49,- DM

PC-E220 TIPS und TRICKS  
ISBN 3-89374-055-4, VK = 49,- DM

PC-E500 Anwendungshandbuch  
ISBN 3-89374-050-1, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-074-4, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-075-9, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-076-5, VK = 49,- DM

PC-E500 MaschinenSprachHandbuch  
ISBN 3-89374-077-3, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-078-8, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-079-4, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-080-9, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-085-6, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-087-1, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-088-7, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-089-5, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-090-1, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-091-6, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-092-2, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-093-7, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-094-3, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-095-9, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-096-5, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-097-0, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-098-6, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-099-3, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-100-9, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-101-4, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-102-0, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-103-5, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-104-1, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-105-6, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-106-2, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-107-7, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-108-3, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-109-8, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-110-4, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-111-9, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-112-5, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-113-0, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-114-6, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-115-1, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-116-7, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-117-2, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-118-8, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-119-3, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-120-9, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-121-4, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-122-0, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-123-5, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-124-1, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-125-6, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-126-2, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-127-7, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-128-3, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-129-8, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-130-4, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-131-9, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-132-5, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-133-0, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-134-6, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-135-1, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-136-7, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-137-2, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-138-8, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-139-3, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-140-9, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-141-4, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-142-0, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-143-5, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-144-1, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-145-6, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-146-2, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-147-7, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-148-3, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-149-8, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-150-4, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-151-9, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-152-5, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-153-0, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-154-6, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-155-1, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-156-7, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-157-2, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-158-8, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-159-3, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-160-9, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-161-4, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-162-0, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-163-5, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-164-1, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-165-6, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-166-2, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-167-7, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN 3-89374-168-3, VK = 49,- DM

PC-E500 Systemhandbuch  
ISBN 3-89374-169-8, VK = 49,- DM

PC-E500 TIPS und TRICKS  
ISBN