

KLEINCOMPUTER

KC compact
Systemhandbuch Teil2

veb mikroelektronik >wilhelm pieck<
mühlhausen
im veb kombinat mikroelektronik

Das betrifft: IND KM TEST BREAK
Tastaturpuffer
Expansions-Puffer und -Strings
BREAKs werden ignoriert

BB06 KM WAIT CHAR Warten auf ein (erweitertes)
Zeichen von der Tastatur

PE: keine
PA: CY=1 und A=Zeichen
UR: BC,DE,HL,IX,IY

Es wird so lange gewartet, bis ein Zeichen von der Tastatur verfügbar ist. Erweiterungszeichen und an den KM zurückgegebene Zeichen werden ausgewertet.

BB09 KM READ CHAR Zeichen (auch erweitertes) von
der Tastatur holen

PE: keine
PA: wenn CY=1,dann A=Zeichen,sonst CY=0
UR: BC,DE,HL,IX,IY

Dieses UP übergibt ein Zeichen von der Tastatur, wenn eins vorhanden ist (CY=1). Die Routine wartet nicht. Erweiterungszeichen und zurückgegebene Zeichen werden ebenfalls ausgewertet.

BB0C KM CHAR RETURN Zeichen an den KM zurückgeben

PE: A=Zeichen
PA: keine
UR: AF,BC,DE,HL,IX,IY

Dem KM wird ein Zeichen zurückgegeben. Beim nächsten Versuch, ein Zeichen von der Tastatur abzuholen, wird dann dieses Zeichen ausgegeben. Es wird dabei nicht expandiert, auch wenn es ein Erweiterungszeichen ist. Das Zeichen FFH kann nicht zurückgegeben werden.

BB0F KM SET EXPAND einem Erweiterungszeichen eine
Zeichenkette zuordnen

PE: B=Erweiterungszeichen
C=Länge des Strings
HL=Adresse des Strings
PA: CY=1 -> kein Fehler
UR: IX,IY

Die Erweiterungszeichen 128-159 können mit Zeichenketten belegt werden. Dazu sind die Puffergröße und -adresse der Zeichenkette

anzugeben. Die Zeichenkette darf überall im RAM stehen (nicht im ROM). Ein Fehler (CY=0) tritt dann auf, wenn im Puffer kein Platz mehr ist oder B keine gültige Erweiterungszeichennummer enthält.

BB12 KM GET EXPAND Zeichen aus einer Erweiterungs-
Zeichenkette holen

PE: A=Erweiterungszeichen
L=Zeichennummer in der Erweiterungszeichenkette
PA: wenn CY=1, dann A=Zeichen
UR: BC,HL,IX,IY

Die Zeichen sind beginnend mit 0 durchnumeriert. Ein Fehler (CY=0) tritt auf, wenn A keine gültige Nummer enthält oder die Kette kürzer ist, als durch L verlangt wird.

BB15 KM EXP BUFFER Speicherbereich für Erweiterungs-
zeichenketten festlegen

PE: DE=Adresse für den Puffer
HL=Länge
PA: CY=1 -> ohne Fehler
UR: IX,IY

Der Puffer wird entsprechend DE und HL übernommen und mit den Standard-Expansion-Strings initialisiert. Ein Fehler (CY=0) tritt auf, wenn der Puffer dafür zu kurz ist. Dann wird der alte Puffer nicht freigegeben! Der neue Puffer muß deshalb mindestens 44 Zeichen lang sein. Der Puffer darf nur im zentralen RAM liegen (4000H-BFFFH).

BB18 KM WAIT KEY Warten auf ein nicht erweitertes
Zeichen von der Tastatur

PE: keine
PA: CY=1 und A=Zeichen
UR: BC,DE,HL,IX,IY

Die Routine wartet so lange, bis ein Zeichen von der Tastatur verfügbar ist. Erweiterungszeichen werden dabei nicht expandiert und zurückgegebene Zeichen nicht berücksichtigt.

BB1B KM READ KEY Holen eines nicht erweiterten
Zeichens von der Tastatur

PE: keine
PA: A=Zeichen, wenn CY=1
UR: BC,DE,HL,IX,IY

Wenn ein nicht erweitertes Zeichen verfügbar ist (CY=1), wird es abgeholt, sonst aber nicht gewartet. Erweiterungszeichen werden

nicht expandiert und zurückgegebene Zeichen nicht berücksichtigt.

BB1E KM TEST KEY Test auf eine bestimmte Taste

PE: A=Tastenummer

PA: Z=0 -> Taste ist gedrückt, sonst Z=1
CY=0 und C=Zustand von [SHIFT] und [CTRL]

UR: B,DE,IX,IY

Bit 7,C =1 -> [CTRL] ist gedrückt

Bit 5,C =1 -> [SHIFT] ist gedrückt

BB21 KM GET STATE Statusabfrage für [SHIFT]- und [CAPS LOCK]-Taste

PE: keine

PA: L=SHIFT LOCK-Status
H=CAPS LOCK-Status

UR: BC,DE,IX,IY

Wird in L bzw. H eine 0 zurückgegeben, ist das entsprechende LOCK nicht eingeschaltet.

BB24 KM GET JOYSTICK Joystickabfrage

PE: keine

PA: A und H=Status von Joystick 0
L =Status von Joystick 1

UR: BC,DE,IX,IY

gesetzte Bits im Status-Byte bedeuten:

0 -hoch

1 -runter

2 -links

3 -rechts

4 -Feuer 1

5 -Feuer 2

6 -nicht belegt

7 -immer 0

BB27 KM SET TRANSLATE Erstbelegung einer Taste festlegen

PE: A=Tastenummer
B=ASCII-Zeichen

PA: keine

UR: BC,DE,IX,IY

Es dürfen nur Tastennummern kleiner 80 verwendet werden. Folgende ASCII-Zeichen werden vom KM nicht weitergegeben, wenn die Tastatur abgefragt wird:

FDH --> CAPS LOCK-Schalter
FEH --> SHIFT LOCK-Schalter
FFH --> Ignorierzeichen

BB2A KM GET TRANSLATE Erstbelegung einer Taste abfragen

PE: A=Tastenummer
PA: A=ASCII-Zeichen
UR: BC,DE,IX,IY

KM GET TRANSLATE übergibt den ASCII-Code einer Taste in der Erstbelegung (ohne SHIFT bzw. CTRL). Die Zeichen 0FDH bis 0FFH werden nicht übergeben.

BB2D KM SET SHIFT Zweitbelegung einer Taste festlegen

Wie BB27 KM SET TRANSLATE, nur mit SHIFT.

BB30 KM GET SHIFT Zweitbelegung einer Taste abfragen

Wie BB2A KM GET TRANSLATE, nur mit SHIFT.

BB33 KM SET CONTROL Drittbelegung einer Taste festlegen

Wie BB27 KM SET TRANSLATE, nur mit CTRL.

BB36 KM GET CONTROL Drittbelegung einer Taste abfragen

Wie BB2A KM GET TRANSLATE, nur mit CTRL.

BB39 KM SET REPEAT Festlegen, ob eine Taste repetieren darf

PE: A=Tastenummer
 B=FFH --> Auto-Repeat ein, sonst B=0
PA: keine
UR: DE,IX,IY

Es sind nur Tastennummern kleiner 80 zugelassen.

BB3C KM GET REPEAT Erfragen, ob eine Taste repetiert

PE: A=Tastennummer

PA: Z=0 --> Taste repetiert, sonst Z=1

UR: BC,DE,IX,IY

BB3F KM SET DELAY Festlegen der Verzögerungszeit
beim Repetieren

PE: H=Zeit bis zum ersten Repeat
L=Zeit zwischen weiteren Repeats

PA: keine

UR: BC,DE,HL,IX,IY

Für H und L gilt 1/50 Sekunde als Zeiteinheit. Standardwerte sind 30 (0.6s) für H, und 2 (0.04s) für L. Werden die Zeichen nicht oder zu langsam aus dem Tastenpuffer abgeholt, wird das Repeaten unterbrochen. Sonst würde sich der Tastaturpuffer unbenutzt füllen.

BB42 KM GET DELAY Abfrage der Repeat-Verzögerungszeiten

PE: keine

PA: H=Zeit bis zum ersten Repeat
L=Zeit zwischen folgenden Repeats

UR: BC,DE,IX,IY

Siehe BB3F KM SET DELAY.

BB45 KM ARM BREAK Break-Routine initialisieren

PE: DE=Adresse der Break-Behandlungsroutine
C=ROM-Selekt-Byte (siehe Abschn. 3.7.)

PA: keine

UR: IX,IY

Das BREAK-Event ist immer synchron, express mit der Priorität 0 und mit far adress (siehe Abschn. 3.7.).

BB48 KM DISARM BREAK Break-Mechanismus abschalten

PE: keine

PA: keine

UR: BC,DE,IX,IY

BB4B KM BREAK EVENT Break-Event aufrufen, wenn der
Break-Mechanismus aktiv ist

PE: keine

PA: keine

UR: BC,DE,IX,IY

Bei aktiviertem Break-Mechanismus wird der Break-Eventblock in die Synchronous Pending Queue eingehängt und das BREAK-Event-Token (EFH=) in den Tastenpuffer eingefügt, wenn darin noch Platz ist. Der Break-Mechanismus wird danach wieder ausgeschaltet. Diese Routine ist dafür vorgesehen, vom Interrupt-Pfad aus aufgerufen zu werden.

BD3A KM SET LOCKS SHIFT- und CAPS LOCK-Status neu festlegen

PE: H=neuer CAPS LOCK-Status
L=neuer SHIFT LOCK-Status
PA: keine
UR: BC,DE,HL,IX,IY

Es gelten folgende Vereinbarungen:

00H --> Ausschalten des jeweiligen LOCK-Status
FFH --> Einschalten des jeweiligen LOCK-Status

BD3D KM FLUSH Tastaturpuffer löschen

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

Der Tastaturpuffer wird komplett gelöscht. Ein eventuell angefangenes Erweiterungszeichen oder ein zurückgegebenes Zeichen wird mit gelöscht.

3.9.1.2. Die Textausgabe-Routinen - TEXT VDU (TXT)

BB4E TXT INITIALISE Initialisierung der Text-VDU

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: - die Indirections der Text-VDU
 - die Control-Code-Funktionstabelle

Alle Streams werden wie folgt eingestellt:

- PAPER 0,PEN 1,
- Window=ganzer Screen
- Cursor ist erlaubt und ausgeschaltet
- Hintergrund-Modus ist deckend
- Schreiben auf Grafik-Cursor-Position ist ausgeschaltet
- der Cursor wird in der ersten Spalte der ersten Zeile positioniert
- es wird der Zeichensatz (alle 256 Zeichen aus dem ROM eingestellt
- Text-Stream 0 wird angewählt

BB51 TXT RESET

Rücksetzen der Text-VDU

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: - die Indirections der Text-VDU
- die Control-Code-Funktionstabelle

BB54 TXT VDU ENABLE

Zulassen, daß Zeichen im aktiven Fenster ausgegeben werden dürfen

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

Der Control-Code-Puffer wird geleert, der Cursor aktiviert und die Zeichenausgabe zugelassen.
Dieser Vektor wirkt auf BB5A TXT OUTPUT und BB5D TXT WR CHAR.

BB57 TXT VDU DISABLE

Zeichenausgabe im aktiven Fenster verbieten

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

Der Control-Code-Puffer wird geleert, der Cursor abgeschaltet und die Zeichenausgabe verboten.
Dieser Vektor wirkt auf BB5A TXT OUTPUT und BB5D TXT WR CHAR.

BB5A TXT OUTPUT

Zeichenausgabe auf Bildschirm, Control-Codes werden ausgeführt

PE: A=Zeichen,Control-Code oder dessen Parameter
PA: keine
UR: alle Register bleiben erhalten

Wartet ein vorher ausgegebener Control-Code noch auf einen Parameter, so wird A als Parameter behandelt.
Ist der Grafikmodus (TAG)eingeschaltet, werden alle Zeichen (0 bis 255) über BBFC GRA WR CHAR ausgedruckt.

BB5D TXT WR CHAR

Zeichenausgabe auf Bildschirm, Control-Codes werden als Sonderzeichen gedruckt

PE: A=Zeichen
PA: keine
UR: IX,IY

BB60 TXT RD CHAR Zeichen vom Bildschirm lesen

PE: keine
PA: CY=1 --> A=Zeichen, sonst CY=0 --> nicht identifizierbar
UR: BC,DE,HL,IX,IY

Es wird versucht, das Zeichen auf der aktuellen Cursor-Position zu lesen. Dazu wird der Bildschirminhalt mit der Zeichenmatrix verglichen.

BB63 TXT SET GRAPHIK Festlegen, ob der Text auf der Text- oder Grafik-Cursor-Position ausgegeben werden soll

PE: A=Schaltflag
PA: keine
UR: BC,DE,HL,IX,IY

Es gelten folgende Festlegungen:

A=0 Text auf der Text-Cursor-Position ausgegeben

A>0 Text auf der Grafik-Cursor-Position ausgegeben

Betroffen hiervon ist nur BB5A TXT OUTPUT. Wenn der Grafik-Schreibmodus aktiviert ist, werden Control-Codes nicht befolgt, sondern als Sonderzeichen gedruckt. Das Zeichenausgabeverbot ist hier nicht wirksam.

BB66 TXT WIN ENABLE Grenzen des aktuellen Textfensters festlegen

PE: H und D=linke und rechte Spalte
L und E=obere und untere Zeile des Textfensters
PA: keine
UR: IX,IY

Die angegebenen Werte werden automatisch nach ihrer Größe sortiert und entsprechend dem Bildschirmmodus auf die maximal möglichen Werte reduziert. Die linke, obere Ecke besitzt die Koordinaten (0,0).

BB69 TXT GET WINDOW Erfragen der Textfenster-Grenzen

PE: keine
PA: H und D = linke und rechte Spalte
L und E = obere und untere Zeile
UR: BC,IX,IY

Die linke, obere Ecke besitzt die Koordinaten (0,0).

BB6C TXT CLEAR WINDOW Löschen des aktuellen Textfensters

PE: keine
PA: keine
UR: IX,IY

Das aktuelle Textfenster wird mit dessen Paper-Tinte gelöscht. Der Cursor wird in die linke, obere Ecke gesetzt.

BB6F TXT SET COLUMN Cursor in angegebene Spalte bewegen

PE: A=Cursor-Spalte
PA: keine
UR: BC,DE,IX,IY

Die linke Spalte des aktuellen Textfensters hat die Spaltennummer 1. Es dürfen auch Werte außerhalb des Fensters angegeben werden. Vor einer Zeichenausgabe wird der Cursor in das Fenster zurückbewegt.

BB72 TXT SET ROW Cursor in die angegebene Zeile bewegen

PE: A=Cursor-Zeile
PA: keine
UR: BC,DE,IX,IY

Die oberste Zeile des aktuellen Textfensters hat die Zeilennummer 1. Sonst gilt das bei BB6F TXT SET COLUMN geschriebene.

BB75 TXT SET CURSOR neue Cursor-Position festlegen

PE: H=Spalte
L=Zeile des Cursors
PA: keine
UR: BC,DE,IX,IY

Die linke obere Ecke des aktuellen Textfensters hat die Koordinate (1,1). Sonst gilt das bei BB6F TXT SET COLUMN geschriebene.

BB78 TXT GET CURSOR Ermitteln der Cursor-Position

PE: keine
PA: H=Spalte
L=Zeile
A=Scroll-Zähler (Hardware-Scroll)
UR: BC,DE,IX,IY

Mit jedem Zeichenrollen nach oben wird der Scroll-Zähler decrementiert, mit jedem Zeichenrollen nach unten incrementiert. Sonst gilt das bei BB75 TXT SET CURSOR beschriebene.

BB7B TXT CUR ENABLE Einschalten des Cursors auf Be-
nutzerebene

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

Diese Routine ist für Anwenderprogramme vorgesehen. Bei Verbot der Zeichenausgabe wird auch der Cursor abgeschaltet.

BB7E TXT CUR DISABLE Ausschalten des Cursors auf Be-
nutzerebene

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

BB81 TXT CUR ON Einschalten des Cursors auf
Systemebene

PE:keine
PA:keine
UR:BC,DE,HL,AF,IX,IY

Dieses UP ist für das Einschalten des Cursors auf Betriebs-
system-Ebene vorgesehen.

BB84 TXT CUR OFF Ausschalten des Cursors auf
Systemebene

PE: keine
PA: keine
UR: BC,DE,HL,AF,IX,IY

Siehe BB81 TXT CUR ON.

BB87 TXT TXT VALIDATE Test, ob sich eine Position
innerhalb des aktuellen Textfen-
sters befindet

PE: H=Spalte der Position
L=Zeile der Position
PA: H=Spalte der ins Fenster zurückgezwungenen Position
L=Zeile der ins Fenster zurückgezwungenen Position
CY=1 --> Fenster wurde nicht gescrollt
CY=0 und B=0 --> Fenster mußte gescrollt werden
UR: BC,DE,IX,IY

Die linke, obere Ecke hat die Koordinatenangaben (1,1). Diese Routine dient der Sicherstellung, daß ein Zeichen bzw. der Cursor innerhalb des Fensters ausgegeben werden. Nötigenfalls wird das Fenster gescrollt.

BB8A TXT PLACE CURSOR Cursorfleck auf den Bildschirm bringen

PE: keine

PA: keine

UR: BC,DE,HL,IX,IY

Werden mehrere Cursor benötigt, können diese mit dieser Routine auf den Bildschirm gebracht werden. Der Copy-Cursor ist ein Beispiel dafür. Die zusätzlichen Cursor muß der Anwender selbst verwalten.

BB8D TXT REMOVE CURSOR Löschen eines Cursorflecks

PE: keine

PA: keine

UR: BC,DE,HL,IX,IY

Siehe BB8A TXT PLACE CURSOR.

BB90 TXT SET PEN Tinte für die Buchstaben festlegen

PE: A=Tintenummer

PA: keine

UR: BC,DE,IX,IY

Die Tintenummer wird abhängig vom Bildschirmmodus automatisch mit 1, 3 oder 15 maskiert. Der Cursor wird der neuen Tinte angepaßt.

BB93 TXT GET PEN Ermitteln der aktuellen Stift-Tinte

PE: keine

PA: A=Vordergrund-Tinte

UR: BC,DE,IX,IY

BB96 TXT SET PAPER Tinte für den Hintergrund der Buchstaben festlegen

PE: A=Hintergrund-Tinte

PA: keine

UR: BC,DE,IX,IY

Siehe BB90 TXT SET PEN. Außerdem wird das Fenster immer mit dieser Tinte gelöscht.

BB99 TXT GET PAPER Ermitteln der aktuellen Hintergrund-Tinte

PE: keine
PA: A=Hintergrund-Tinte
UR: BC,DE,HL,IX,IY

BB9C TXT INVERSE Austausch der Stift-und Hintergrund-Tinte im aktuellen Textfenster

PE: keine
PA: keine
UR: BC,DE,IX,IY

BB9F TXT SET BACK Festlegen des Hintergrundmodus

PE: A=Hintergrund-Flag
PA: keine
UR: BC,DE,IX,IY

Es gelten folgende Festlegungen:

A=0 --> deckend: Zeichen-Hintergrund wird mit Hintergrund-Tinte gelöscht,

A=1 --> transparent: Zeichen-Hintergrund bleibt erhalten, es werden nur die Pixel des Zeichens selber mit der Stift-Tinte gesetzt.

Diese Einstellung gilt nur für die Ausgabe auf der Textcursor-Position. Soll der Hintergrundmodus auch für die Ausgabe auf der Grafikcursor-Position eingestellt werden, muß BD46 GRA SET BACK verwendet werden.

BBA2 TXT GET BACK Abfrage, ob Transparentmodus eingeschaltet ist

PE: keine
PA: A=Hintergrund-Flag
UR: BC,IX,IY

Siehe BB9F TXT SET BACK.

BBA5 TXT GET MATRIX Ermitteln der Zeichenbildmatrix-adresse eines Zeichens

PE: A=Zeichencode
PA: HL=Adresse erstes Byte der Zeichenmatrix
CY=1 --> Matrix befindet sich im RAM
CY=0 --> Matrix befindet sich im ROM
UR: BC,DE,UX,IY

Die Matrix eines Zeichens ist 8*8 Pixel groß und damit in 8 Bytes gespeichert. Diese liegen hintereinander im Speicher. Das erste Byte stellt die oberste Pixelzeile, Bit 7 jeweils das Pixel ganz links dar usw.

BBA8 TXT SET MATRIX Festlegen einer neuen Matrix für ein Zeichen

PE: A =Zeichencode
HL=Anfangsadresse der neuen Matrix
PA: CY=1 --> kein Fehler
UR: IX,IY

Die zu verändernde Matrix muß im RAM liegen. Die Matrix besteht aus 8 Bytes (siehe BBA5 TXT GET MATRIX).

BBAB TXT SET M TABLE Festlegen eines neuen Speicherbereichs für die Zeichenmatrizen

PE: D =Flag
E =erstes Zeichen der neuen Zeichenbildtabelle
HL=Adresse der Zeichenmatrix
PA: CY=0 --> vorher keine Tabelle im RAM
CY=1 --> A=altes erstes Zeichen der Zeichenbildtabelle
HL=alte Adresse der Zeichenbildtabelle
UR: IX,IY

Es gelten folgende Festlegungen:

D>0 : Zeichentabelle wird im RAM vollständig gelöscht (alle Matrizen werden aus dem ROM verwendet)

D=0 : entsprechend E und HL wird im RAM eine neue Tabelle installiert.

Der benötigte Speicherplatz für die Zeichentabelle ergibt sich aus $(256-E)*8$ Bytes. Die Tabelle muß vollständig im zentralen RAM liegen.

In die neue Tabelle werden die bisher gültigen Zeichenmatrizen aus RAM oder/und ROM mit LDIR kopiert, was zu beachten ist, wenn sich eine alte Tabelle mit der neuen überschneidet. In dem Fall sollte die Anfangsadresse der neuen Tabelle unterhalb der der alten liegen.

BBAE TXT GET TABLE Ermitteln der Anfangsadresse einer selbstdefinierten Zeichenbildtabelle

PE: keine
PA: CY=0 --> keine Zeichenbildtabelle im RAM
CY=1 --> A=erstes Zeichen in der Tabelle
HL=Anfangsadresse der Tabelle
UR: BC,DE,IX,IY

Siehe BBA5 TXT GET MATRIX und BBA8 TXT SET TABLE.

BBB1 TXT GET CONTROLS Ermitteln der Anfangsadresse der
Control-Code-Tabelle

PE: keine
PA: HL=Anfangsadresse der Control-Code-Tabelle
UR: AF,BC,DE,IX,IY

Die Control-Code-Tabelle enthält zu jedem Control-Code (0 bis 31) in aufsteigender Reihenfolge in jeweils 3 Bytes folgende Informationen:

DEFB Anzahl benötigter Parameter (maximal 9)
DEFW Routinenadresse, die im zentralen RAM liegen muß.

Die Routinenadressen können durch Anfangsadressen eigener Routinen ersetzt werden. Den Control-Code-Routinen werden vom Betriebssystem bei deren Aufruf folgende Registerinhalte übergeben:

- A und C=letzter Parameter oder Control-Code (keine Parameter gebraucht)
- B=Anzahl Parameter + 1
- HL=Adresse vor dem ersten Parameter (Control-Code selbst)

Im Bit 7 des Parameteranzahlbytes wird festgelegt, ob der Control-Code trotz evtl. eingestelltem Zeichenausgabeverbotes im aktuellen Fenster ausgeführt werden soll oder nicht. Es gilt folgende Festlegung:

Bit 7 = 0 --> Control-Code wird in jedem Fall befolgt
Bit 7 = 1 --> Ausführung ist davon abhängig, ob eine Zeichenausgabe im aktuellen Fenster erlaubt ist oder nicht

BBB4 TXT STREAM SELECT Textfensterauswahl

PE: A=neue Textfensternummer
PA: A=alte Textfensternummer
UR: BC,DE,IX,IY

A wird mit 7 AND-verknüpft. Dadurch werden Fensternummern von 0 bis 7 erzeugt. Folgende Parameter werden für jedes Fenster verwaltet:

Vordergrund-Tinte
Hintergrund-Tinte
Cursor-Position (Spalte und Zeile)
Fenstergrenzen
Cursor-Status (ein/aus und erlaubt/verboten)
Zeichenausgabe erlaubt (ja/nein)
Hintergrundmodus (deckend/transparent)
Zeichenausgabe auf Grafikcursor-Position (ein/aus)

BBB7 TXT SWAP STREAMS Parameter zweier Fenster tauschen

PE: B und C =Nummern der Fenster
PA: keine
UR: IX,IY

Beide Nummern werden mit 7 AND-verknüpft, so daß nur Nummern von 0 bis 7 erzeugt werden.

BD40 TXT ASK STATE Ermitteln des Cursor- und VDU-
Status im aktuellen Fenster

PE: keine
PA: A=Status
UR: BC,DE,HL,IX,IY

Folgende Informationen werden in A zurückgegeben:

Bit 0 =0 Cursor erlaubt,=1 Cursor verboten (Anwender Ebene)
Bit 1 =0 Cursor erlaubt,=1 Cursor verboten (Systemebene)
Bit 7 =0 Zeichenausgabe erlaubt,=1 Zeichenausgabe verboten

3.9.1.3. Die Grafik-Routinen - GRAPHICS VDU (GRA)

BBBA GRA INITIALISE Initialisierung der Grafik-VDU

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: die Indirections der Grafik-VDU
Hintergrund-Tinte=0
Vordergrund-Tinte=1
Grafik-Fenster=ganzer Bildschirm
Koordinatenursprung=0,0 (ORIGIN)
Grafikcursor-Position=0,0
Hintergrundmodus=deckend
Linienmaske=FFH (durchgezogene Linie)
Erster-Punkt-Modus=Punkt wird immer gesetzt

BBBD GRA RESET Rücksetzen der Grafik-VDU

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: die Indirections der Grafik-VDU
Hintergrundmodus=deckend
Linienmaske=FFH (durchgezogene Linie)
Erster-Punkt-Modus=Punkt wird immer gesetzt

Die Koordinaten werden relativ zur linken, unteren Ecke mit den Koordinaten (0,0) und vorzeichenbehaftet (-32768 bis +32767) zurückgegeben.

BBCF GRA WIN WIDTH linke und rechte Grafikfenster-
grenze festlegen

PE: DE und HL=X-Koordinaten der linken und rechten Fenstergrenze
PA: keine
UR: IX,IY

Die Koordinatenangaben sind relativ zur linken unteren Ecke mit den Koordinaten (0,0) und vorzeichenbehaftet (-32768 bis +32767). DE und HL werden verglichen, und der kleinere Werte wird automatisch für den linken Rand verwendet. Weiterhin werden zu große Werte so verkleinert, daß das Fenster auf den Bildschirm paßt. Die Werte werden auf ein Vielfaches von 8 (ganzes Byte im Bildwiederholtspeicher) gerundet.

BBD2 GRA WIN HIGHT obere und untere Grafikfenster-
grenze festlegen

PE: DE und HL=Y-Koordinaten der oberen und unteren Fenstergrenze
PA: keine
UR: IX,IY

Es gelten die Aussagen von BBCF GRA WIN WIDTH, nur daß hier die obere und untere Grenze festgelegt wird. Die Werte werden auf ein Vielfaches von 2 gerundet.

BBD5 GRA GET W WIDTH Ermitteln der linken und rechten
Grafikfenstergrenze

PE: keine
PA: DE und HL=linke und rechte Fenstergrenze
UR: BC,IX,IY

Siehe BBCF GRA WIN WIDTH.

BBD8 GRA GET W HIGHT Ermitteln der oberen und unteren
Grafikfenstergrenze

PE: keine
PA: DE und HL= obere und untere Fenstergrenze
UR: BC,IX,IY

Siehe BBD2 GRA WIN HIGHT.

BBDB GRA CLEAR WINDOW Grafikfenster löschen

PE: keine
PA: keine
UR: IX,IY

Das Grafikfenster wird mit der Grafik-Hintergrund-Tinte gelöscht. Der Grafikcursor wird zum Koordinatenursprung bewegt.

BBDE GRA SET PEN Zeichenstift-Tinte festlegen

PE: A=Tintennummer
PA: keine
UR: BC,DE,HL,IX,IY

Die Zeichenstift-Tinte wird zum Punktsetzen, Linienzeichnen und zur Textausgabe auf der Grafikcursor-Position benutzt.

BBE1 GRA GET PEN Ermitteln der Zeichenstift-Tinte

PE: keine
PA: A=Tintennummer
UR: BC,DE,HL,IX,IY

BBE4 GRA SET PAPER Festlegen der Hintergrund-Tinte für Grafikausgaben

PE: A=Tintennummer
PA: keine
UR: BC,DE,HL,IX,IY

Die Hintergrund-Tinte wird zum Löschen des Grafikfensters, für den Hintergrund bei der Grafik-Textausgabe und beim Linienzeichnen mit Linienmaske (bei jedem nicht gesetztem Bit) verwendet.

BBE7 GRA GET PAPER Ermitteln der Hintergrund-Tinte

PE: keine
PA: A=Tintennummer
UR: BC,DE,HL,IX,IY

BBEA GRA PLOT ABSOLUTE Punktsetzen auf die angegebene Position

PE: DE=X-Koordinate
HL=Y-Koordinate
PA: keine
UR: IX,IY

Entsprechend dem Grafik-Vordergrundmodus wird die Grafikvordergrund-Tinte mit der alten Tinte des Punktes verknüpft und der Punkt entsprechend gesetzt. Liegt der Punkt außerhalb des Grafikfensters, wird er nicht gesetzt. Die Koordinaten sind vorzeichenbehaftet und relativ zum Ursprung anzugeben.

BBED GRA PLOT RELATIVE Punktsetzen relativ zur Grafikcursor-Position

PE: DE=X-Versatz
 HL=Y-Versatz
PA: keine
UR: IX,IY

Die Koordinaten sind vorzeichenbehaftet und relativ zur Grafikcursor-Position anzugeben. Sonst siehe BBEA GRA PLOT ABSOLUTE.

BBF0 GRA TEST ABSOLUTE Ermitteln der Tintennummer eines Punktes

PE: DE=X-Koordinate des Testpunktes
 HL=Y-Koordinate des Testpunktes
PA: A =Tintennummer
UR: IX,IY

Die Koordinaten sind relativ zum Ursprung und vorzeichenbehaftet anzugeben. Bei Punkten außerhalb des Grafikfensters wird die aktuelle Hintergrund-Tinte zurückgegeben.

BBF3 GRA TEST RELATIVE Ermitteln der Tintennummer eines Punktes relativ zum Grafikcursor

PE: DE=X-Versatz
 HL=Y-Versatz
PA: A =Tintennummer
UR: IX,IY

Die Koordinaten sind relativ zur Grafikcursor-Position und vorzeichenbehaftet anzugeben. Sonst siehe BBF0 GRA TEST ABSOLUTE.

BBF6 GRA LINE ABSOLUTE Linie von Grafikcursor-Position zur absoluten Position

PE: DE=X-Koordinate des Zielpunktes
 HL=Y-Koordinate des Zielpunktes
PA: keine
UR: IX,IY

Siehe BBEA GRA PLOT ABSOLUTE.

BBF9 GRA LINE RELATIVE

Linie von Grafikcursor-Position zu einer zu ihr relativen Position

PE: DE=X-Versatz
HL=Y-Versatz
PA: keine
UR: IX,IY

Siehe BBED GRA PLOT RELATIVE.

BBFC GRA WR CHAR

Buchstabe auf Grafikcursor-Position zeichnen

PE: A=Zeichencode
PA: keine
UR: IX,IY

Control-Codes werden nicht befolgt. Der Grafikcursor bildet immer die linke, obere Ecke der Zeichenmatrix. Nach dem Zeichnen wird der Grafikcursor um eine Buchstabenposition nach rechts bewegt.

BD43 GRA DEFAULT

Standardwerte für Grafikausgaben einstellen

PE: keine
PA: keine
UR: IX,IY

Betroffen sind:

Vordergrund-Modus -deckend
Hintergrund-Modus -deckend
Erster-Punkt-Modus -erster Punkt beim Linienzeichnen wird gesetzt
Linienmaske -FFH (durchgezogene Linie)

BD46 GRA SET BACK

Hintergrund-Modus festlegen

PE: A=Hintergrundmodus
PA: keine
UR: AF,BC,DE,HL,IX,IY

Es gelten folgende Festlegungen:

A=0 --> deckend
A>0 --> transparent

BD49 GRA SET FIRST

Erster-Punkt-Modus für die Darstellung von Linien festlegen

PE: A=Erster-Punkt-Modus
PA: keine
UR: AF,BC,DE,HL,IX,IY

Es gelten folgende Festlegungen:

A=0 --> ersten Punkt nicht zeichnen (sinnvoll, wenn Vordergrundmodus auf XOR eingestellt ist)

A>0 --> ersten Punkt zeichnen

BD4C GRA SET LINE MASK Linienmaske festlegen

PE: A=Linienmaske

PA: keine

UR: AF,BC,DE,HL,IX,IY

Mit dem Festlegen einer Linienmaske ist es möglich, unterbrochene (gestrichelte) Linien zu zeichnen. Gesetzte Bits in der Maske bedeuten Punktsetzen mit der Vordergrund-Tinte, nicht gesetzte in der Hintergrund-Tinte.

BD4F GRA FROM USER Umrechnen einer Koordinate relativ zum Ursprung in Koordinate relativ zur linken, unteren Bildschirmcke

PE: DE=X-Koordinate relativ zum Ursprung

HL=Y-Koordinate relativ zum Ursprung

PA: DE=X-Koordinate relativ zur Bildschirmcke

HL=Y-Koordinate relativ zur Bildschirmcke

UR: BC,IX,IY

BD52 GRA FILL Füllen einer beliebigen Fläche

PE: A =Fülltinte

HL=Pufferadresse

DE=Pufferlänge

PA: CY=1 --> vollständig gefüllt

CY=0 --> nicht oder nicht vollständig gefüllt

UR: IX,IY

Startpunkt ist immer die Grafikcursor-Position. Als Füllgrenze gelten:

- das Grafikfenster
- Punkte, die in der Fülltinte gesetzt sind
- Punkte, die in der Grafikvordergrund-Tinte gesetzt sind.

Ist der für die Füllfunktion zur Verfügung gestellte Puffer (Speichern von Verzweigungspunkten) zu klein, wird die Fläche nicht vollständig gefüllt (CY=0). Pro Verzweigung werden 7 Bytes benötigt. Für einfache Flächen reicht eine Puffergröße von etwa 100 Bytes aus.

3.9.1.4. Die Bildschirm-Routinen - SCREEN PACK (SCR)

BBFF SCR INITIALISE

Initialisierung des SCR

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: die Indirections des Screen-Packs
alle Tinten werden auf ihre Standardwerte gesetzt
die Blinkperioden werden auf ihre Standardwerte
gesetzt
der Bildschirmmodus wird auf MODE 1 eingestellt
der Bildwiederholpeicher wird auf C000H einge-
stellt
der Scroll-Offset wird auf 0 gesetzt
der Bildschirm wird mit Tinte 0 gelöscht
der Grafikmodus wird auf deckend eingestellt
das Event zum Farbenblinken wird initialisiert

BC02 SCR RESET

Rücksetzen des SCR

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: die Indirections des Screen-Packs
alle Tinten werden auf ihre Standardwerte gesetzt
die Blinkperioden werden auf ihre Standardwerte
gesetzt
der Grafikmodus wird auf deckend eingestellt

BC05 SCR SET OFFSET

Hardware-Scroll-Offset verändern

PE: HL=neuer Scroll-Offset
PA: keine
UR: BC,DE,IX,IY

Um einen erlaubten Wert sicherzustellen, wird der Scroll-Offset,
zuerst mit 07FEH maskiert.

BC08 SCR SET BASEBildwiederholpeicher in ein an-
deres Speicherviertel verlegen

PE: A=RAM-Viertel
PA: keine
UR: BC,DE,IX,IY

Zulässige Werte für A sind 0, 4, 8 und CH. Der neue Bildschirm
wird nicht gelöscht und der Scroll-Offset des alten Bildschirms
wird mit übernommen.

BC0B SCR GET LOCATION

Ermitteln des Bildwiederhol-
speicherviertels und des Scroll-
Offsets

PE: keine
PA: A =RAM-Viertel
HL=Scroll-Offset
PA: BC,DE,IX,IY

BC0E SCR SET MODE

Festlegen Bildschirmmodus mit
Bildschirmlöschen

PE: A=Bildschirm-Modus
PA: keine
UR: IX,IY

Die Routine wertet nur Bit 0 und 1 von A aus. Sind beide gesetzt, wird ohne Änderung zurückgekehrt, ansonsten wird der Bildschirm mit Tinte 0 gelöscht, alle Fenster werden auf Maximalgröße eingestellt, der Cursor wird in die linke obere Ecke, der Grafikkursor und der Ursprung (Origin) in die linke untere Ecke gestellt.

BC11 SCR GET MODE

Ermitteln des eingestellten Bild-
schirmmodus

PE: keine
PA: A=Bildschirmmodus
CY und Z siehe unten
UR: BC,DE,HL,IX,IY

MODE=0 --> CY=1, Z=0 und A=0
MODE=1 --> CY=0, Z=1 und A=1
MODE=2 --> CY=0, Z=0 und A=2

BC14 SCR CLEAR

Bildschirm mit Tinte 0 löschen

PE: keine
PA: keine
UR: IX,IY

BC17 SCR CHAR LIMITS

Ermitteln Spalten-und Zeilen-
anzahl des Bildschirms

PE: keine
PA: B=letzte Spalte
C=letzte Zeile
UR: DE,HL,IX,IY

Die linke, obere Ecke hat die Koordinaten (0,0). In Abhängigkeit vom eingestellten Bildschirmmodus sind für B die Werte 19, 39 oder 79 möglich. In C wird immer 24 zurückgegeben.

BC1A SCR CHAR POSITION Ermitteln der Bildwiederhol-
speicheradresse aus einer Zeichen-
Position

PE: H =Spalte
 L =Zeile
PA: HL=Adresse im Bildwiederholpeicher
 B =Bytes pro Buchstabenbreite
UR: C,DE,IX,IY

Die linke, obere Ecke hat die Koordinaten (0,0). Die errechnete Adresse (Byte)im Bildwiederholpeicher ist immer die linke, oberste Zeile der Zeichenmatrix.

BC1D SCR DOT POSITION Ermitteln der Bildwiederhol-
speicheradresse aus einer Grafik-
Position

PE: DE=X-Koordinate
 HL=Y-Koordinate
PA: HL=Byte-Adresse
 C =Bitmaske für den Punkt
 B =Anzahl Punkte pro Byte-1
UR: IX,IY

Die Koordinaten beziehen sich auf die linke untere Bildschirmecke (0,0). Der Wertebereich in X-Richtung ist abhängig vom Bildschirmmodus:

Modus 0: 0 - 159

Modus 1: 0 - 319

Modus 2: 0 - 639

Y kann Werte zwischen 0 und 199 einnehmen. Jedem realen Pixel ist im Unterschied zu BASIC eine Koordinate zugeordnet.

BC20 SCR NEXT BYTE Ermitteln der Adresse rechts
neben der angegebenen Adresse im
Bildwiederholpeicher

PE: HL=Bildwiederholpeicheradresse
PA: HL=Bildwiederholpeicheradresse rechts daneben
UR: BC,DE,IX,IY

BC23 SCR PREV BYTE Ermitteln der Adresse links neben
der angegebenen Adresse im Bild-
wiederholpeicher

PE: HL=Bildwiederholpeicheradresse
PA: HL=Bildwiederholpeicheradresse links daneben
UR: BC,DE,IX,IY

BC26 SCR NEXT LINE Ermitteln der Adresse unter der angegebenen Adresse im Bildwiederholtspeicher

PE: HL=Bildwiederholtspeicheradresse
PA: HL=Bildwiederholtspeicheradresse darunter
UR: BC,DE,IX,IY

BC29 SCR PREV LINE Ermitteln der Adresse über der angegebenen Adresse im Bildwiederholtspeicher

PE: HL=Bildwiederholtspeicheradresse
PA: HL=Bildwiederholtspeicheradresse darüber
UR: BC,DE,IX,IY

BC2C SCR INK ENCODE Konvertieren einer Tintennummer in das in den Bildschirmspeicher zu speichernde Byte, wenn alle Pixel mit der Tinte gesetzt werden sollen

PE: A=Tintennummer
PA: A=Farb-Byte
UR: BC,DE,HL,IX,IY

Das so erhaltene Byte ist vom Bildschirmmodus abhängig. Zusammen mit einer Punktmaske kann das Farb-Byte zum Punktsetzen verwendet werden.

BC2F SCR INK DECODE Ermitteln der Tintennummer des ersten Punktes von links in einem Bildspeicher-Byte

PE: A=Bildspeicher-Byte (Farb-Byte)
PA: A=Tintennummer
UR: BC,DE,HL,IX,IY

Die erhaltene Tintennummer ist abhängig vom Bildschirmmodus.

BC32 SCR SET INK Farben für eine Tinte festlegen

PE: A=Tintennummer
 B=Farbe der ersten Blinkperiode
 C=Farbe der zweiten Blinkperiode
PA: keine
UR: IX,IY

BC35 SCR GET INK Ermitteln der Farben einer Tinte

PE: A=Tintenummer
PA: B=Farbe der ersten Blinkperiode
 C=Farbe der zweiten Blinkperiode
UR: IX,IY

BC38 SCR SET BORDER Festlegen der Farben des Bildschirmrandes (BORDER)

PE: B=Farbe der ersten Blinkperiode
 C=Farbe der zweiten Blinkperiode
PA: keine
UR: IX,IY

BC3B SCR GET BORDER Ermitteln der Farben des Bildschirmrandes (BORDER)

PE: keine
PA: B=Farbe der ersten Blinkperiode
 C=Farbe der zweiten Blinkperiode
UR: IX,IY

BC3E SCR SET FLASHING Blinkperioden festlegen

PE: H=Länge der ersten Blinkperiode
 L=Länge der zweiten Blinkperiode
PA: keine
UR: BC,DE,IX,IY

Die Werte werden in 1/50 Sekunden angegeben.

BC41 SCR GET FLASHING Ermitteln der Blinkperioden

PE: keine
PA: H=Länge der ersten Blinkperiode
 L=Länge der zweiten Blinkperiode
UR: BC,DE,IX,IY

Siehe BC3E SCR SET FLASHING.

BC44 SCR FILL BOX Füllen einer rechteckigen Fläche mit einer Tinte (Grenzen in Zeichenpositionen)

PE: A=Farb-Byte (mit BC2C SCR INK ENCODE ermittelt)
 H=linke Zeichen-Spalte
 D=rechte Zeichen-Spalte
 L=oberste Zeichen-Zeile
 E=unterste Zeichen-Zeile

PA: keine
UR: IX,IY

BC47 SCR FLOOD BOX Füllen einer rechteckigen Fläche mit einer Tinte (Grenzen in Bytepositionen)

PE: C=Farb-Byte (mit BC2C SCR INK ENCODE ermittelt)
HL=Adresse des Bytes in der linken, oberen Ecke
D=Breite der Füllfläche in Bytes
E=Höhe der Füllfläche in Pixel-Zeilen

PA: keine
UR: IX,IY

BC4A SCR CHAR INVERT Invertieren einer Zeichenposition (Cursorfleck erzeugen)

PE: B und C=je ein Farb-Byte (mit BC2C SCR INK ENCODE erzeugt)
H=Zeichen-Spalte
L=Zeichen-Zeile

PA: keine
UR: IX,IY

Die Bytes der Zeichenposition werden wie folgt verknüpft:
neues Byte = altes Byte XOR B XOR C

Mit dieser Routine werden die Cursorflecken erzeugt. Durch nochmaliges Aufrufen mit den gleichen Parametern wird der ursprüngliche Zustand wieder hergestellt.

BC4D SCR HW ROLL Bildschirm hardwaremäßig um eine Zeile scrollen

PE: B=0 Bildschirm nach unten scrollen
B>0 Bildschirm nach oben scrollen
A=Farb-Byte (zum Füllen der entstehenden Leerzeile, mit BC2C SCR INK ENCODE erzeugt).

PA: keine
UR: IX,IY

BC50 SCR SW ROLL Bildschirmausschnitt (Fenster) um eine Zeile scrollen

PE: B=0 Bildschirmausschnitt nach unten scrollen
B>0 Bildschirmausschnitt nach oben scrollen
A=Farb-Byte (zum Füllen der entstehenden Leerzeile)
H=linke Spalte
D=rechte Spalte
L=oberste Zeile
E=unterste Zeile

PA: keine
UR: IX,IY

3.9.1.5. Die Kassetten-Routinen - CASSETTE MANAGER (CAS)

BC65 CAS INITIALISE Initialisierung der Kassetten-Routinen

PE: keine
PA: keine
UR: IX,IY

Betroffen sind: die Vektoren der Kassetten-Routinen
die Eingabe- und Ausgabedatei wird geschlossen
Einstellen der Schreibgeschwindigkeit auf 1000 Baud
die Meldungen der Kassetten-Routinen werden zugelassen
der Kassettenmotor wird ausgeschaltet

BC68 CAS SET SPEED Schreibgeschwindigkeit festlegen

PE: HL=halbe Periodenlänge eines 0-Bits
A =Vorkompensation
PA: keine
UR: BC,DE,IX,IY

siehe Abschnitt 3.3.

Für HL sind Werte zwischen 130 und 480 zulässig.

Standardwerte: 1000 Baud: HL=333 und A=25
2000 Baud: HL=167 und A=50

BC6B CAS NOISY Bildschirmmeldungen ein/aus

PE: A=0 Meldungen werden auf Bildschirm ausgegeben
A>0 Meldungen werden unterdrückt
PA: keine
UR: BC,DE,HL,IX,IY

Betroffen sind: Press PLAY then any key:
 Press REC and PLAY then any key:
 Found dateiname> block nummer>
 Loading dateiname> block nummer>
 Saving dateiname> block nummer>

nicht betroffen sind: Read error code>
 Write error code>
 Rewind tape

BC6E CAS START MOTOR Start des Kassettenrecordermotors

PE: keine
PA: A =alter Schaltzustand

CY=1 [ESC] wurde nicht betätigt
CY=0 [ESC] wurde betätigt (BREAK)
UR: BC,DE,HL,IX,IY

Der Motor wird immer gestartet. Wenn er vorher nicht lief, wird nach dem Start ca. 2 Sekunden gewartet. Wird in der Zwischenzeit [ESC] gedrückt, kehrt die Routine sofort zurück (CY=0).

BC71 CAS STOP MOTOR Stopp des Kassettenrecordermotors

PE: keine
PA: A =alter Schaltzustand
CY=1 [ESC] wurde nicht betätigt
CY=0 [ESC] wurde betätigt (BREAK)
UR: BC,DE,HL,IX,IY

BC74 CAS RESTORE MOTOR Start oder Stopp des Kassettenrecorders (Herstellen des ursprünglichen Schaltzustands)

PE: A =alter Schaltzustand
PA: CY=1 [ESC] wurde nicht betätigt
CY=0 [ESC] wurde betätigt (BREAK)
UR: BC,DE,HL,IX,IY

Entsprechend A wird der Motor gestoppt oder gestartet. Nach einem Start wird ca. 2 Sekunden gewartet.

BC77 CAS IN OPEN Eingabedatei eröffnen

PE: B =Länge des Dateinamens
HL=Adresse des Dateinamens
DE=Adresse des 2KByte-Eingabe-Puffers
PA: CY=0 Fehler (siehe Abschn. 3.3.)
CY=1 kein Fehler
A=Dateityp
BC=logische Dateilänge
DE=Einsprungadresse (bei Maschinencode)
HL=Adresse des 64-Byte-Puffers mit dem File-Header
UR: IY

Der Eingabe-Puffer und der Name der Datei können im gesamten RAM liegen. Die Kleinbuchstaben des Namens werden in Großbuchstaben gewandelt. Die Routine liest sofort die ersten 2 KByte der Datei. Der Datei-Header steht also sofort zur Verfügung. Es kann immer nur eine Eingabedatei eröffnet werden.

BC7A CAS IN CLOSE Eingabedatei schließen

PE: keine

PA: CY=0 Fehler (siehe Abschn 3.3.)
CY=1 kein Fehler

UR: IX,IY

Der Eingabepuffer wird wieder zur anderweitigen Nutzung freigegeben.

BC7D CAS IN ABANDON

Eingabedatei vergessen

PE: keine

PA: keine

UR: IX,IY

Diese Routine schließt die Eingabedatei und ist für den Fehlerfall gedacht.

BC80 CAS IN CHAR

ein Zeichen aus der Eingabedatei holen

PE: keine

PA: CY=0 Fehler (siehe Abschn.3.3.)
CY=1 kein Fehler,dann A=Zeichen

UR: BC,DE,HL,IY

Es ist nur möglich, eine Datei zeichenweise oder blockweise zu lesen. Wechselseitig zeichen- und blockweises Lesen sind nicht möglich.

BC83 CAS IN DIRECT

Eingabedatei mit einem Mal lesen

PE: HL=Adresse, ab der die Datei abgelegt werden soll

PA: CY=0 Fehler (siehe Abschn. 3.3.)
CY=1 kein Fehler, dann HL=Einsprungadresse

UR: IY

Nach der Eingabedateieröffnung (BC77 CAS IN OPEN)durfte noch kein Zeichen mit BC80 CAS IN CHAR gelesen worden sein.

BC86 CAS RETURN

Zurückgeben des letzten gelesenen Zeichens an die Eingabedatei

PE: keine

PA: keine

UR: AF,BC,DE,HL,IX,IY

Bedingung ist, daß bereits mindestens ein Zeichen gelesen wurde. Es kann nur maximal ein Zeichen zurückgegeben werden.

BCA1 CAS READ Speicherblock von Kassette einlesen

PE: HL=Zieladresse
DE=maximale Anzahl Bytes, die gelesen werden sollen
A =Synchronisationszeichen (siehe BC9E WRITE)
PA: CY=0 Fehler oder [ESC], A enthält Fehlercode (siehe Abschn. 3.3.)
CY=0 kein Fehler
UR: IY

BCA4 CAS CHECK Vergleich der Daten auf Kassette mit einem Speicherbereich

PE: HL=Anfangsadresse der Vergleichsdaten
DE=Länge des Blocks
A =Synchronisationszeichen (siehe BC9E CAS WRITE)
PA: CY=0 Fehler, A=Fehlercode (siehe Abschn. 3.3.)
CY=1 kein Fehler
UR: IY

3.9.1.6. Die Soundausgabe-Routinen - SOUND MANAGER (SOUND)

BCA7 SOUND RESET Rücksetzen des SOUND MANAGER

PE: keine
PA: keine
UR: IX,IY

Aktivitäten: -Stoppen der Tonausgabe
- Leeren der Ton-Warteschlangen
- Deaktivieren des SOUND QUEUE-EVENT

Die Hüllkurven bleiben erhalten.

BCAA SOUND QUEUE Ton zum SOUND MANAGER senden

PE: HL=Anfangsadresse Parameterblock
PA: CY=1 Ton wurde in die Warteschlange aufgenommen,
CY=0 Ton wurde nicht aufgenommen
UR: IY, wenn CY=0, dann auch HL

Der Parameterblock hat folgenden Aufbau:

DEFB Kanalstatus:

Bit 0=1: Ton für Kanal A
Bit 1=1: Ton für Kanal B
Bit 2=1: Ton für Kanal C
Bit 3=1: Rendezvous mit Kanal A
Bit 4=1: Rendezvous mit Kanal B
Bit 5=1: Rendezvous mit Kanal C

Bit 6=1: Hold-Status (Ton wird, wenn er an der Reihe ist nicht gestartet, er muß mit einer extra Routine gestartet werden)
Bit 7=1: Flush (Ton wird sofort ausgeführt, außer bei Bit 6=1)

DEFB Nr. Volumen-Hüllkurve (0=keine Änderung)
DEFB Nr. Frequenz-Hüllkurve (0=keine Änderung)
DEFW Periodenlänge (Frequenz)
DEFB Rauschperiode (0=kein Rauschen)
DEFB Startamplitude
DEFW Dauer (in 1/100 s) oder Wiederholfaktor

Für eine Pause ist eine Periodenlänge von 0 einzutragen. Soll die Länge der Volumenhüllkurve als Tonlänge gelten, muß eine negative Dauer angegeben werden. Der Betrag der Dauer bestimmt in diesem Fall, wie oft die Hüllkurve abgearbeitet werden soll.

BCAD SOUND CHECK

Anfrage, ob in der Ton-Warteschlange Platz für einen weiteren Ton ist

PE: A=zu testender Kanal
PA: A=Kanalstatus
UR: IX,IY

In A wird in den Bits 0, 1 und 2 (Kanal A, B und C) die Nummer des zu testenden Kanals übergeben. Sind mehrere Bits gesetzt wird der Kanal mit höchster Priorität getestet (A vor B und B vor C). zurückgegebener Status:

Bits 0,1,2 = Anzahl freier Plätze in Warteschlange

Bits 3,4,5 = Rendezvousbits (gesetzt, wenn der erste Ton in der Warteschlange ein Rendezvous mit einem Ton in einer oder beiden anderen Warteschlangen hat)

Bit 6 =1 --> erster Ton befindet sich im Haltezustand

Bit 7 =1 --> erster Ton ist gerade aktiv

Das SOUND QUEUE-EVENT des getesteten Kanals wird deaktiviert.

BCB0 SOUND ARM EVENT

Festlegen einer Routine, die gerufen wird, wenn ein Platz in einer Warteschlange frei wurde

PE: A =Kanal (siehe BCAD SOUND CHECK)
HL=Adresse des Event-Blocks
PA: keine
UR: IX,IY

Der Event-Block muß vollständig initialisiert sein. Deaktiviert wird der Event-Block, wenn

- er gekickt wurde,
- BCAA SOUND QUEUE oder
- BCAD SOUND CHECK aufgerufen wird.

BCB3 SOUND RELEASE Tonausgabe freigeben

PE: A=Kanal (siehe BCAD SOUND CHECK, es werden aber auch mehrere Kanäle freigegeben)

PA: keine

UR: IY

Töne, die mit BCB6 SOUND HOLD eingefroren wurden, oder Töne, bei denen Bit 6 im Statusbyte gesetzt wurde, werden freigegeben.

BCB6 SOUND HOLD Tonausgabe einfrieren

PE: keine

PA: CY=1 Töne waren gerade aktiv
CY=0 kein Ton war aktiv

UR: DE,IX,IY

Die Tonausgabe wird sofort unterbrochen. Fortgesetzt wird die Tonausgabe mit BCAA SOUND QUEUE, BCB3 SOUND RELEASE oder BCB9 SOUND CONTINUE.

BCB9 SOUND CONTINUE Tonausgabe fortsetzen

PE: keine

PA: keine

UR: HL,IY

BCBC SOUND AMPL ENVELOPE Volumenhüllkurve festlegen

PE: A =Hüllkurvennummer (1 bis 15)
HL=Anfangsadresse Parameterblock

PA: CY=1 kein Fehler
CY=0 ungültige Hüllkurvennummer

UR: IX,IY, wenn CY=0, dann auch HL, BC und A

Parameterblock:

DEFB Anzahl Hüllkurvenabschnitte (bei 0: zwei Sekunden konstanter Ton)

danach entsprechend oft:

DEFB Schrittzahl (0...127, bei 0 --> absolute Volumeneinstellung)

DEFB Schritthöhe (0...255 modulo 16)

DEFB Schrittlänge (0...255, 0=256)

Maximal können fünf Abschnitte angegeben werden. Der Parameterblock darf nur im zentralen RAM liegen.

Aufbau eines Hardware-Hüllkurven-Abschnitts:

DEFB 80H + Wert für Register 13 (Hüllkurvenform siehe Tab. 2.12)

DEFW Periodenlänge für den Hüllkurvengenerator (Reg. 11,12 siehe Abschn.2.1.10.)

BCBF SOUND TONE ENVELOPE Frequenzhüllkurve festlegen

PE: A =Hüllkurvennummer
 HL=Anfangsadresse Parameterblock
 PA: CY=1 kein Fehler
 CY=0 ungültige Hüllkurvennummer
 UR: IX,IY,wenn CY=0,dann auch A,BC,HL

Parameterblock:

DEFB Anzahl Hüllkurvenabschnitte (+80H, wenn die Hüllkurve über die volle Tonlänge wiederholt werden soll)

danach entsprechend oft:

DEFB Anzahl Schritte (0...239)

DEFB Schritthöhe (-128...+127)

DEFB Schrittlänge (0...255, 0=256)

Es können auch absolute Tonperiodenlängen eingestellt werden. Ein solcher Abschnitt ist folgendermaßen aufgebaut:

DEFB Periodenlänge/256 OR F0H ;Low-Teil und High-Teil der Perio-

DEFB Periodenlänge AND FFH ;denlänge werden vertauscht

DEFB Schrittlänge

BCC2 SOUND A ADDRESS Ermittle die Adresse einer
Volumenhüllkurve

PE: A =Hüllkurvennummer
 PA: CY=1 kein Fehler: HL=Adresse der Hüllkurve und
 BC=Länge der Hüllkurve (immer 16)
 CY=0 unzulässige Hüllkurvennummer
 UR: DE,IX,IY, wenn CY=0, dann auch BC

BCC5 SOUND T ADDRESS Ermittle die Adresse einer
Frequenzhüllkurve

PE: A =Hüllkurvennummer
 PA: CY=1 kein Fehler: HL=Adresse der Hüllkurve und
 BC=Länge der Hüllkurve (immer 16)
 CY=0 unzulässige Hüllkurvennummer
 UR: DE,IX,IY, wenn CY=0, dann auch BC

3.9.1.7. Die Zentrale - KERNEL (KL)**BCC8 KL CHOKE OFF** Rücksetzen des Kernels

PE: keine
 PA: B =ROM-Select-Adresse des laufenden Vordergrund-Programms
 DE=Kaltstartadresse des laufenden Vordergrund-Programms
 C =FFH --> ROM-Vordergrundprogramm
 =0 --> RAM-Vordergrundprogramm
 UR: IX,IY

Betroffen sind: - Löschen der Synchronous Pending Queue
- Löschen aller Ticker Chains (außer Sound-Events und Tastaturabfrage)

Diese Routine wird hauptsächlich von BD13 MC BOOT PROGRAM benutzt. Treten bei einem Ladevorgang Fehler auf, kann an Hand der durch BCC8 KL CHOKE OFF zurückgegebenen Parameter zum lauffähigen Vordergrundprogramm zurückgekehrt werden. Erfolgte aber der Aufruf BD13 MC BOOT PROGRAM von einem RAM-Vordergrundprogramm, wird bei einem Fehlerfall immer zum BASIC (Kaltstart des Systems) zurückgekehrt.

BCCB KL ROM WALK Initialisierung aller Hintergrund-ROMs

PE: DE=Adresse des 1.Byte des freien Speicherbereichs
HL=Adresse des letzten Byte des freien Speicherbereichs
PA: DE=Adresse des 1.Byte des freien Speicherbereichs
HL=Adresse des letzten Byte des freien Speicherbereichs
UR: IX,IY

Es werden alle Hintergrund-ROMs von 0 bis 15 berücksichtigt (BASIC=0). Ein laufendes ROM-Vordergrundprogramm wird nicht mit initialisiert. Jeder Hintergrund-ROM kann sich einen bestimmten Speicherbereich reservieren (z.B. Bereich der Arbeitszellen von BASIC).

Dazu werden jeweils Unter-und Obergrenze des freien RAM-Bereichs von ROM zu ROM weitergegeben, und jeder ROM kann sich benötigte Bereiche reservieren. Der RAM wird also dynamisch zugeteilt. Alle ROMs besitzen deshalb einen Header, der wie folgt aufgebaut ist:

C000H - DEFB ROMTYP - 00H= Vordergrund-ROM
80H= eingebauter Vordergrund-ROM (BASIC)
01H= Hintergrund-ROM
02H= Erweiterungs-ROM (wenn ein Vordergrund-ROM aus mehreren 16KByte-ROMs besteht, werden dessen Erweiterungen mit 02 gekennzeichnet)

C001H - DEFB MARKNR : ROM-Nummer
C002H - DEFB VERSION: Versionsnummer
C003H - DEFB MODIFI : Änderungsnummer
C004H - ff : Tabelle externer Kommandos (erster Eintrag muß immer die Initialisierungsroutine sein, Aufbau der Tabelle siehe BCD1 KL LOG EXT)

BCCE KL INIT BACK Initialisierung eines Hintergrund-ROMs

PE: C =ROM-Select-Adresse des ROM
DE=Adresse des ersten Bytes des freien Speicherbereichs
HL=Adresse des letzten Bytes

UR: BC,IX,IY

Siehe Abschnitt 3.7.

BCDD KL DEL FRAME FLY Entfernen eines Frame-Flyback-Datenblocks aus der Chain

PE: HL=Anfangsadresse des Frame-Flyback-Blocks

PA: keine

UR: BC,IX,IY

Handelt es sich um ein synchrones Event, kann es sein, daß noch einige Interrupts auf die Abarbeitung warten. Soll das unterbleiben, muß BCF8 KL DEL SYNCHRONOUS aufgerufen werden.

BCE0 KL NEW FAST TICKER Initialisierung eines Datenblocks für die Fast-Ticker-Chain

PE: HL=Anfangsadresse des Fast-Ticker-Blocks

B =Event-Klasse

C =ROM-Selekt-Byte für die Event-Routine

DE=Adresse der Event-Routine

PA: keine

UR: BC,IX,IY

Siehe BCD7 KL NEW FRAME FLY.

BCE3 KL ADD FAST TICKER Einfügen eines Fast-Ticker-Datenblocks in die Fast-Ticker-Chain

PE: HL=Anfangsadresse des Fast-Ticker-Blocks

PA: keine

UR: BC,IX,IY

Siehe Abschnitt 3.7.

BCE6 KL DEL FAST TICKER Entfernen eines Fast-Ticker-Datenblocks aus der Chain

PE: HL=Anfangsadresse des Fast-Ticker-Blocks

PA: keine

UR: BC,IX,IY

Siehe BCDD KL DEL FRAME FLY.

BCE9 KL ADD TICKER Einfügen eines Ticker-Datenblocks in die Ticker-Chain

PE: HL=Anfangsadresse des Ticker-Blocks

DE=Startverzögerung (Count Down)

BC=Wiederholverzögerung (Reload Count)

BD04 KL EVENT DISABLE Verbietet die Ausführung aller normalen synchronen Events

PE: keine

PA: keine

UR: AF,BC,DE,IX,IY

Siehe Abschnitt 3.7.

BD07 KL EVENT ENABLE Zulassung der Ausführung aller normalen synchronen Events

PE: keine

PA: keine

UR: AF,BC,DE,IX,IY

Siehe Abschnitt 3.7.

BDOA KL DISARM EVENT Verbietet die Abarbeitung eines Event-Blocks

PE: HL=Anfangsadresse des Event-Blocks

PA: keine

UR: BC,DE,HL,IX,IY

Der Kick Counter wird auf einen negativen Wert gesetzt. Noch ausstehende Event-Abarbeitungen gehen verloren (außer bei synchronen Events). Neu eintreffende Kicks werden ignoriert. Diese Routine ist nur für asynchrone Event-Blocks zu verwenden.

BD0D KL TIME PLEASE Ermitteln des Interruptzählerstandes

PE: keine

PA: DEHL=Zeit in 1/300 Sekunden

UR: AF,BC,IX,IY

Bei jedem Hardware-Interrupt wird der Zähler inkrementiert. Bei der Kassettenarbeit wird der Interrupt oft verboten, so daß in dem Fall nicht korrekt gezählt wird, sonst ist der Zähler zur Zeitmessung geeignet.

BD10 KL TIME SET Festlegen eines neuen Interruptzählerstandes

PE: DEHL=Zeit in 1/300 Sekunden

PA: keine

UR: BC,DE,HL,IX,IY

3.9.1.8. Die maschinennahen Routinen - MACHINE PACK (MC)

BD13 MC BOOT PROGRAM Laden und Starten eines Maschinencode-Programms

PE: HL=Adresse der Laderoutine
PA: -/-
UR: -/-

Aktivitäten: - Zurücksetzen von Peripheriegeräten
 - Löschen aller Software-Interrupts
 - Initialisierung aller Vektoren
 - Aufruf des Ladeprogramms

Die Laderoutine muß folgendes zurückgeben:

CY=0 Ladefehler (in der Regel Rückkehr ins BASIC)
CY=1 kein Fehler, HL=Einsprungadresse ins Maschinencode-Programm

BD16 MC START PROGRAM Start eines Vordergrund-Programms

PE: HL=Einsprungadresse
 C =ROM-Select-Byte
PA: -/-
UR: -/-

Aktivitäten: - Zurücksetzen des Betriebssystems
 - selektieren des ROMs
 - Vordergrundprogramm anspringen

BD19 MC WAIT FLYBACK Wartet auf den nächsten Strahlrücklauf des Bildschirms

PE: keine
PA: keine
UR: AF,BC,DE,HL,IX,IY

Diese Routine ist nicht interruptgesteuert. Sie wartet in einer Schleife auf den nächsten Strahlrücklauf.

BD1C MC SET MODE Festlegen des Bildschirmmodus

PE: A=Bildschirmmodus (0,1 oder 2)
PA: keine
UR: BC,DE,HL,IX,IY

Der Screen Pack wird nicht über den neuen Modus informiert und der Bildschirm nicht gelöscht. Alle Ausgaben werden so gemacht, als wäre kein neuer Modus eingestellt.

BD1F MC SCREEN OFFSET Festlegen des Hardware-Scroll-Offsets

PE: HL=Scroll-Offset
A =RAM-Viertel für den Bildwiederholtspeicher (0, 4, 8, 0CH)
PA: keine
UR: BC,DE,HL,IX,IY

Das Screen Pack wird nicht informiert.

BD22 MC CLEAR INKS alle Tinten auf eine Farbe setzen

PE: DE=Anfangsadresse einer Farbtabelle
PA: keine
UR: BC,DE,HL,IX,IY

Aufbau Farbtabelle:

Byte 1=Paletten-Farbnummer für den Bildschirmrand (BORDER)
Byte 2=Paletten-Farbnummer für alle Tinten

BD25 MC SET INKS Festlegen aller Tintenfarben

PE: DE=Anfangsadresse einer Farbtabelle
PA: keine
UR: BC,DE,HL,IX,IY

Aufbau Farbtabelle:

Byte 1=Paletten-Farbnummer für den Bildschirmrand (BORDER)
Byte 2=Paletten-Farbnummer für Tinte 0
Byte 3=Paletten-Farbnummer für Tinte 1

.
.
.

Byte 17=Paletten-Farbnummer für Tinte 15

BD28 MC RESET PRINTER Drucker-Vektoren zurücksetzen

PE: keine
PA: keine
UR: IX,IY

Die Druckerübersetzungstabelle wird mit ihren Standardwerten gefüllt.

BD2B MC PRINT CHAR Zeichen zum Drucker senden

PE: A =Zeichen
PA: CY=1 Zeichen wurde gedruckt
CY=0 Zeichen wurde nicht gedruckt
UR: BC,DE,HL,IX,IY

Konnte das Zeichen nicht innerhalb von 0,4 Sekunden gedruckt werden, kehrt die Routine mit CY=0 zurück.

BD2E MC BUSY PRINTER Feststellen, ob der Drucker bereit ist

PE: keine

PA: CY=1 Drucker ist nicht bereit oder nicht angeschlossen
CY=0 Drucker ist bereit

UR: A,BC,DE,HL,IX,IY

BD31 MC SEND PRINT Zeichen zum Drucker senden

PE: A=Zeichen

PA: CY=1

UR: BC,DE,HL,IX,IY

Die Routine überprüft nicht die Bereitschaft des Druckers!

BD34 MC SOUND REGISTER Soundcontroller-Register laden

PE: A =Registernummer

C =Datenbyte für das Register

PA: keine

UR: DE,HL,IX,IY

BD58 MC PRINT TRANSLATION Festlegen einer neuen Zeichen-Übersetzungstabelle für den Drucker

PE: HL=Anfangsadresse der neuen Übersetzungstabelle

PA: CY=1 kein Fehler

CY=0 Tabelle zu lang

UR: IX,IY

Aufbau der Übersetzungstabelle:

1. Byte - Anzahl der Einträge (max.20)

Aufbau eines Eintrags:

1. Byte - zu übersetzender Zeichencode

2. Byte - zugeordneter Zeichencode

Die Tabelle wird in die Übersetzungstabelle des Betriebssystems kopiert. Der Tabellenbereich kann deshalb danach anderweitig verwendet werden.

Standardbelegung der Übersetzungstabelle:

alt	neu	alt	neu
A0H	5EH	ABH	7CH
A1H	5CH	ACH	7DH
A2H	7BH	ADH	7EH
A3H	23H	AEH	5DH
A6H	40H	AFH	5BH

3.9.1.9. Routine für die Sprungleiste - JUMPER (JUMP)

BD37 JUMP RESTORE Herstellen der Original-Sprungleiste

PE: keine
PA: keine
UR: IX,IY

Alle Vektoren von 0BB00H bis 0BD5EH werden in den RESET-Zustand des KC compact zurückgesetzt.

3.9.1.10. Die Indirections der Betriebssystem-Packs (IND)

BDCD IND TXT DRAW CURSOR Cursor-Fleck zeichnen, falls dieser auf System- und Anwender-Ebenen eingeschaltet ist

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

BDD0 IND TXT UNDRAW CURSOR Entfernen des Cursor-Flecks, falls dieser auf System- und Anwender-Ebene eingeschaltet ist

PE: keine
PA: keine
UR: BC,DE,HL,IX,IY

BDD3 IND TXT WRITE CHAR Zeichenausgabe auf Bildschirm

PE: A=Zeichencode
 H=Spalte
 L=Zeile
PA: keine
UR: IX,IY

Die linke, untere Ecke (0,0) bildet den Bezugspunkt für die Koordinatenangaben. Mit dieser Routine werden keine Control-Codes ausgeführt, keine Ausgaben auf dem Grafikcursor realisiert, die Koordinaten werden nicht geprüft und es wird ohne Cursor gearbeitet.

BDD6 IND TXT UNWRITE Zeichen vom Bildschirm lesen

PE: H=Spalte
 L=Zeile
PA: CY=1 A=Zeichencode
 CY=0 kein Zeichen erkannt
UR: IX,IY

Die linke, untere Ecke (0,0) ist Bezugspunkt für die Koordinatenangaben. Soll die Cursor-Position geprüft werden, darf der Cursor nicht dargestellt sein. Es wird in zwei Durchgängen versucht, das Zeichen zu erkennen. Zuerst wird angenommen, daß das Zeichen mit der aktuellen Stift-Tinte geschrieben wurde. Wurde kein Zeichen erkannt, wird im zweiten Durchgang angenommen, daß der Hintergrund des Zeichens mit der aktuellen Hintergrund-Tinte geschrieben wurde.

BDD9 IND TXT OUT ACTION Zeichen auf Bildschirm ausgeben
oder Control-Code ausführen

PE: A=Zeichen-oder Control-Code
PA: keine
UR: IX,IY

BDDC IND GRA PLOT Punkt zeichnen

PE: DE=X-Koordinate
HL=Y-Koordinate
PA: keine
UR: IX,IY

Bezugspunkt für die Koordinatenangaben ist der Ursprung (Origin).

BDDF IND GRA TEST Ermitteln der Tinte eines Punkte

PE: DE=X-Koordinate
HL=Y-Koordinate
PA: A=Tintenummer des Punktes
UR: IX,IY

Bezugspunkt für die Koordinatenangaben ist der Ursprung (Origin).
Der Cursor wird auf die neue Position gesetzt.

BDE2 IND GRA LINE Linie zeichnen

PE: DE=X-Koordinate
HL=Y-Koordinate
PA: keine
UR: IX,IY

Bezugspunkt für die Koordinatenangaben ist der Ursprung (Origin).
Es wird eine Linie vom alten Grafikcursor zu den angegebenen Koordinaten gezogen. Der Cursor wird auf die neue Position gesetzt. Mit der Routine werden nur Punkte innerhalb des Grafikfensters gesetzt. Der eingestellte Vordergrund-und Erster-Punkt-Modus werden beachtet.

BDF4 IND KM SCAN KEYS

Tastaturabfrage

PE: keine
PA: keine
UR: IX,IY

Sind Tasten gedrückt, werden diese in den Tastaturpuffer eingetragen. Für diese Routine muß der Interrupt gesperrt sein. Wurde [ESC] gedrückt, wird automatisch BDEE IND TEST BREAK aufgerufen.

3.9.2. Die obere Sprungleiste des Kernel - HIGH KERNEL JUMBLOCK (HI KL)

B900 HI KL U ROM ENABLEEinblenden des selektierten ROM
im Bereich C000H-FFFFH

PE: keine
PA: A=alter ROM-Status
UR: BC,DE,HL,IX,IY

B903 HI KL U ROM DISABLEROM im Bereich C000H-FFFFH
ausblenden

PE: keine
PA: A=alter ROM-Status
UR: BC,DE,HL,IX,IY

B906 HI KL L ROM ENABLE

Betriebssystem-ROM einblenden

PE: keine
PA: A=alter ROM-Status
UR: BC,DE,HL,IX,IY

B909 HI KL L ROM DISABLE

Betriebssystem-ROM ausblenden

PE: keine
PA: A=alter ROM-Status
UR: BC,DE,HL,IX,IY

B90C HI KL ROM RESTOREROM-Konfiguration wieder herstel-
len

PE: A=alter ROM-Status
PA: keine
UR: BC,DE,HL,IX,IY

- B90F HI KL ROM SELECT** ROM auswählen und einblenden im Bereich C000H-FFFFH
- PE: C=ROM-Select-Byte
PA: C=alte ROM-Selection
B=alter ROM-Status
UR: DE,HL,IX,IY
- B912 HI KL CURR SELECTION** Ermitteln,welches ROM im Bereich C000H-FFFFH selektiert ist
- PE: keine
PA: A=ROM-Select-Byte
UR: F,BC,DE,HL,IX,IY
- B915 HI KL PROBE ROM** Klasse und Version eines ROM ermitteln
- PE: C=ROM-Select-Byte
PA: A=ROM-Klasse
L=ROM-Nummer
H=Versionsnummer
UR: C,DE,IX,IY
- Siehe auch BCCB KL ROM WALK.
- B918 HI KL ROM DESELECTION** frühere ROM-Selection wieder herstellen
- PE: C=alte ROM-Selection
B=alter ROM-Status
PA: C=zuletzt eingestelltes ROM
UR: AF,DE,HL,IX,IY
- B91B HI KL LDIR** LDIR
- PE: HL=Adresse 1. Quellbyte
DE=Adresse 1. Zielbyte
BC=Länge des zu verschiebenden Bereichs
PA: wie nach LDIR
UR: wie nach LDIR
- B91E HI KL LDDR** LDDR
- PE: HL=Adresse 1. Quellbyte
DE=Adresse 1. Zielbyte
BC=Länge des zu verschiebenden Bereichs
PA: wie nach LDDR
UR: wie nach LDDR

B921 HI KL POLL SYNCHRONOUS Test, ob ein synchrones Event auf seine Abarbeitung wartet

PE: keine

PA: CY=0 Warteschlange ist leer

CY=1 in der Warteschlange befindet sich ein abzuarbeitendes synchrones Event

UR: BC,DE,HL,IX,IY

Diese Routine entfernt den Event-Block nicht aus der Pending Queue. Sie dient dem schnelleren Polling und kann auch aus einer Event-Behandlungsroutine aufgerufen werden. Gemeldet werden dann nur Event-Blöcke deren Priorität höher ist, als die der sich gerade in der Abarbeitung befindlichen Event-Routine bzw. des aufrufenden Vordergrundprogramms.

B92A HI KL SCAN NEEDED Festlegen, daß beim nächsten Interrupt die Tastatur abgefragt werden soll

PE: keine

PA: keine

UR: AF,BC,DE,IX,IY

3.9.3. Die untere Sprungleiste des Kernel - LOW KERNEL JUMBLOCK (LOW)

0000 LOW RESET ENTRY Kaltstart
RST 0

PE: keine

PA: -/-

UR: -/-

0008 LOW JUMP Sprung zu einer Routine im ROM
RST 8 oder RAM im Bereich 0000H-3FFFH

PE: 2-Byte-Inline-Adresse

PA: keine

UR: AF,BC,DE,HL,IX,IY

Dem Restartbefehl wird im Maschinencode die Adresse der gewünschten Routine nachgestellt. Da Bit 14 und 15 für diesen Adreßbereich nicht gebraucht werden, dienen sie der Speichereinstellung:

Bit 14=0: von 0 bis 3FFFH ROM einblenden

 =1: von 0 bis 3FFFH ROM ausblenden

Bit 15=0: von C000H bis FFFFH ROM einblenden

 =1: von C000H bis FFFFH ROM ausblenden

Beispiel: RST 8

```
DEFW adresse+C000H ;kein ROM ein
```

Die Routine gibt vorher gesperrte Interrupts wieder frei und die alte Speicherkonfiguration wird nach Abarbeitung der gerufenen Routine wieder hergestellt.

000B LOW KL LOW PCHL Sprung zu einer Routine im ROM
oder RAM im Bereich 0000H-3FFFH

PE: HL=ROM-Status und Routinenadresse

PA: keine

UR: AF,BC,DE,HL,IX,IY

Die Routine entspricht 0008 LOW JUMP nur daß Adresse und ROM-Status in HL übergeben werden. HL entfällt somit zur Parameterübergabe.

000E LOW PCBC INSTRUCTION Sprung zur Adresse in BC,JP (BC)

PE: BC=Routinenadresse

PA: keine

UR: AF,BC,DE,HL,IX,IY

Der Sprung wird mit PUSH BC und RET durchgeführt.

0010 LOW SIDE CALL Aufruf einer Routine in einem RST
10H benachbarten Erweiterungs-ROM
im Bereich C000H-FFFFH

PE: 2-Byte-Inline-Adresse

PA: keine

UR: AF,BC,DE,HL,IX

Für den Adreßbereich C000H bis FFFFH werden eigentlich nur 14 Adreßbits gebraucht (BIT 14 und 15 sind immer 1). Deshalb wird zur "Adreßbildung" C000H abgezogen. Bit 14 und 15 stehen somit für die Auswahl des Erweiterungs-ROM zur Verfügung.

Beispiel: RST 10H

```
DEFW adresse-C000H+2*4000H ;Routine im 2.  
;Erweiterungs-ROM
```

Die Routine gibt vorher gesperrte Interrupts wieder frei und die alte Speicherkonfiguration wird nach Abarbeitung der gerufenen Routine wieder hergestellt.

0013 LOW KL SIDE PCHL Aufruf einer Routine in einem
benachbarten Erweiterungs-ROM im
Bereich C000H-FFFFH

PE: HL=Routinenadresse +ROM-Offset
PA: keine
UR: AF,BC,DE,HL,IX

Diese Routine entspricht 0010 LOW SIDE CALL, nur daß Adresse und ROM-Offset in HL übergeben werden. HL entfällt dann zur Parameterübergabe.

0016 LOW PCDE INSTRUCTION Sprung zur Adresse in DE,JP (DE)

PE: DE=Routinenadresse
PA: keine
UR: AF,BC,DE,HL,IX,IY

Der Sprung wird durch PUSH DE und RET durchgeführt.

0018 LOW FAR CALL Aufruf einer Routine im RAM oder
RST 18H ROM im ganzen Adreßbereich

PE: 2-Byte-Inline-Adresse der 3-Byte-far-Adresse
PA: keine
UR: AF,BC,DE,HL,IX

Beispiel: RST 18H
DEFW FADDRESS ;Zeiger far address
...
FADDRESS:DEFW adresse ;Routinenadresse
DEFB konfig ;benötigte ROM-Konfiguration

Mit diesem Restart können alle Adressen in jedem RAM oder ROM angesprungen werden. Das Konfigurationsbyte in der far address wird wie folgt gewertet:

0-FBH: ROM mit dieser Nummer selektieren und einblenden, der Betriebssystem-ROM wird ausgeblendet
FCH : oben wird kein neuer ROM selektiert, oben und unten wird ROM eingeblendet
FDH : oben wird kein neuer ROM selektiert, oben wird ROM und unten wird RAM eingeblendet
FEH : oben wird kein neuer ROM selektiert, oben wird RAM und unten wird ROM eingeblendet
FFH : oben wird kein neuer ROM selektiert, oben und unten wird RAM eingeblendet

Die Routine gibt vorher gesperrte Interrupts wieder frei und die alte Speicherkonfiguration wird nach Abarbeitung der gerufenen Routine wieder hergestellt.

001B LOW KL FAR PCHL Aufruf einer Routine im RAM oder
ROM im ganzen Adreßbereich

PE: HL=Routinenadresse
C =ROM-Konfiguration

PA: keine
UR: AF,BC,DE,HL,IX

Diese Routine entspricht dem RST 18H, nur wird die Adresse in HL und die gewünschte ROM-Konfiguration in C übergeben. Diese Register entfallen somit zur Parameterübergabe.

001E LOW PCHL INSTRUCTION Sprung zur Adresse in HL, JP (HL)

PE: HL=Routinenadresse
PA: keine
UR: AF,BC,DE,HL,IX,IY

Auf dieser Adresse steht der Befehl JP (HL).

0020 LOW RAM LAM Akku mit Inhalt der durch HL
RST 20H adressierten Adresse laden

PE: HL=RAM-Adresse
PA: A =Inhalt dieser Speicherzelle
UR: F,BC,DE,HL,IX,IY

Dieser Restart dient dazu, aus einem laufendem ROM-Programm heraus RAM-Zellen auszulesen. Da Schreibbefehle immer zum RAM gehen, ist ein Vektor für das RAM-Beschreiben nicht erforderlich.

0023 LOW KL FAR ICALL Aufruf einer Routine im RAM oder
ROM im ganzen Adreßbereich

PE: HL=Anfangsadresse der 3-Byte-far-adress
PA: keine
UR: AF,BC,DE,HL,IX

Diese Routine entspricht RST 18H, nur wird hier die far adress in HL übergeben. HL entfällt somit zur Parameterübergabe.

0028 LOW FIRM JUMP Sprung zu einer Routine im
RST 28H Betriebssystem-ROM

PE: Inline-angegebener Vektor zur Routine
PA: keine
UR: AF,BC,DE,HL,IX,IY

Die Adresse der gewünschten Routine wird im Maschinencode unmittelbar nach dem RST 28H angegeben.

Beispiel: RST 28H
DEFW adresse

Unabhängig davon, ob der Betriebssystem-ROM vorher ein- oder ausgeschaltet war, wird er nach der Abarbeitung der gewünschten Routine immer ausgeschaltet. Vorher verbotene Interrupts werden durch diesen Restart wieder freigegeben.

0030 LOW USER RESTART Speichern des ROM-Status in 2BH,
RST 30H Einblenden des unteren RAM und
 Sprung zum RST 20H im RAM

PE: -/- (je nach Anwenderprogramm)

PA: -/- (je nach Anwenderprogramm)

UR: -/- (je nach Anwenderprogramm)

Dieser Restart kann vom Anwender belegt werden. Der ROM-Status sollte nach Abarbeitung der Routine wieder hergestellt werden (2BH).

0038 LOW INTERRUPT ENTRY Einsprungsadresse für alle Hard-
RST 38H ware-Interrupts (Interruptmode 1)

PE: keine

PA: keine

UR: AF,BC,DE,HL,IX,IY

Siehe Abschnitt 3.7.

003B LOW EXT INTERRUPT Einsprungsadresse für externe
 Interrupts

PE: keine

PA: keine

UR: IX,IY

3.9.4. Die BASIC-Vektoren

Bei den nun folgend beschriebenen Unterprogrammen sind einige Routinen dabei, die keine Vektoren im RAM besitzen. Da diese jedoch für den Anwender interessant sein können, wurden sie mit ihren BASIC-ROM-Originaladressen aufgenommen. Diese Routinen müssen dann über die entsprechenden Restarts aufgerufen werden.

3.9.4.1. Der Editor

BD5E EDIT Ändern bzw. Neueingabe einer Zei-
 chenkette

PE: HL=Anfangsadresse des Zeichenketten-Puffers

PA: CY=1 Editieren wurde mit [RETURN] abgeschlossen

 CY=0 Editieren wurde mit [ESC] abgebrochen (Fehlerfall)

UR: BC,DE,HL,IX,IY

Die Länge der zu editierenden Zeichenkette kann nicht größer als 255 Zeichen sein. Die Zeichenkette muß immer mit einem Null-Byte abgeschlossen werden. Aus den beiden Gründen muß die Puffergröße immer 256 Bytes betragen. Der Puffer muß vollständig im zentralen RAM (4000H-BFFFH) liegen. Bei Aufruf des Editors werden alle

Zeichen des Puffers bis zum Null-Byte automatisch im aktuellen Bildschirmfenster ausgegeben. Steuerzeichen werden als Pseudografikzeichen dargestellt. Ausgenommen davon sind die Codes 0,13 und 16, die der Editor als Steuerzeichen interpretiert. Da der Editor von BASIC aus verwendet wird, wird der Cursor, wenn am Pufferanfang eine Zahl gefunden wurde, immer nach der Zahl positioniert, ansonsten immer am Zeichenkettenanfang. Während des Editierens bzw. der Eingabe ist die Anwendung des Copy-Cursors im aktuellen Textfenster möglich. Paßt der Text nicht vollständig in das Textfenster, wird dieses selbständig gescrollt.

3.9.4.2. Die Fließkomma-Routinen

Bei der Nutzung der Fließkomma-Routinen gelten folgende allgemeine Bedingungen:

- Fließkommazahlen werden an das Unterprogramm übergeben, indem Zeiger, die auf sie zeigen, in HL und bei Bedarf in DE übergeben werden. Als Abkürzung für eine Fließkommazahl werden im folgenden, je nach verwendetem Doppelregister, FLO(HL) bzw. FLO(DE) verwendet.
- Ergebniswerte werden mit einem Zeiger auf die Ergebniszahl in HL zurückgegeben. HL wird in den Routinen nicht geändert. In einigen Fällen wird das Ergebnis in A zurückgegeben.
- Im Fehlerfall (CY=0) werden in folgenden Flags zusätzliche Informationen zurückgegeben:
 - Z=1 -Division durch Null
 - S=1 -ungültiger Parameter
 - P=1 -Überlauf
- Die Fließkommazahlen dürfen nicht im Bereich 0-3FFFH liegen.

Zufallszahlenberechnung

BDBB FLO RANDOMIZE 0 89656C07H Standardstartwert zur Zufallszahlenberechnung

PE: keine
PA keine
UR: AF,BC,DE,IX,IY

Zufallszahlen werden immer aus einer vorgegebenen Zahl, einem Doppelwort (4 Bytes), berechnet. Diese Zahl ist beim Start des KC compact 89656C07H. BDBB FLO RANDOMIZE 0 setzt die Zufallszahlenroutine in den Ausgangszustand zurück.

BDBE FLO RANDOMIZE Festlegen eines neuen Doppelworts für die Zufallszahlenberechnung

PE: FLO(HL)=Fließkommazahl
PA: keine
UR: C, IY, FLO(HL)
Das Doppelwort wird mit der Verknüpfung
FLO(HL) XOR 89656C07H
gebildet.

BD7F FLO RND Berechnen einer Zufallszahl

PE: FLO(HL) = Speicher für die Zufallszahl
PA: FLO(HL) = errechnete Zufallszahl
UR: HL,IY

BD8B FLO LAST RND letzte Zufallszahl wiederholen

PE: FLO(HL)= Speicher für die Zufallszahl
PA: FLO(HL)= letzte Zufallszahl
UR: HL,IY

Operationen

BD7C FLO ADD Addition zweier Zahlen

PE: FLO(HL), FLO(DE)
PA: FLO(HL)
CY=0 Überlauffehler
UR: HL,FLO(DE)

Funktion: $FLO(HL) = FLO(HL) + FLO(DE)$

BD82 FLO SUB* Subtraktion zweier Zahlen

PE: FLO(HL), FLO(DE)
PA: FLO(HL)
CY=0 Überlauffehler
UR: HL,FLO(DE)

Funktion: $FLO(HL) = FLO(DE) - FLO(HL)$

349A FLO SUB Subtraktion zweier Zahlen

PE: FLO(HL), FLO(DE)
PA: FLO(HL)
CY=0 Überlauffehler
UR: HL,FLO(DE)

Funktion: $FLO(HL) = FLO(HL) - FLO(DE)$

BD85 FLO MULT Multiplikation zweier Zahlen

PE: FLO(HL), FLO(DE)
PA: FLO(HL)
CY=0 Überlauffehler
UR: HL,FLO(DE)

Funktion: $FLO(HL) = FLO(HL) * FLO(DE)$

BD88 FLO DIV Division zweier Zahlen

PE: FLO(HL), FLO(DE)
PA: FLO(HL)
CY=0, Z=0 Überlauferfehler
CY=0, Z=1 Division durch Null
UR: HL, FLO(DE)

Funktion: $FLO(HL) = FLO(HL) / FLO(DE)$

BDA0 FLO POT Potenzrechnung

PE: FLO(HL), FLO(DE)
PA: FLO(HL)
CY=0, S=1 ungültiger Parameter
CY=0, P=1 Überlauferfehler
UR: HL, FLO(DE)

Funktion: $FLO(HL) = FLO(HL) FLO(DE)$

BD8E FLO VGL Vergleichsfunktion

PE: FLO(HL), FLO(DE)
PA: A=-1, CY=1 wenn $FLO(HL) - FLO(DE) = \text{negativ}$
A=0, Z=1 wenn $FLO(HL) - FLO(DE) = \text{null}$
A=1 wenn $FLO(HL) - FLO(DE) = \text{positiv}$
UR: BC, DE, HL, FLO(HL), FLO(DE)

Funktionen**BD91 FLO VZW** Vorzeichenwechsel

PE: FLO(HL)
PA: FLO(HL)
UR: BC, DE, HL, IY

BD9D FLO SQR Wurzelfunktion

PE: FLO(HL)
PA: FLO(HL)
CY=0 Zahl war negativ
UR: HL

Funktion: $FLO(HL) = SQR(FLO(HL))$

BDA3 FLO LOG NAT natürlicher Logarithmus

PE: FLO(HL)
PA: FLO(HL)
CY=0 Argument war kleiner als Null
UR: HL

Funktion: $FLO(HL) = LOG (FLO(HL))$

BDA6 FLO LOG DEC

dekadischer Logarithmus

PE: FLO(HL)

PA: FLO(HL)

CY=0 Überlauferfehler

UR: HL

Funktion: $FLO(HL) = LOG_{10} (FLO(HL))$

BDAC FLO SIN

Sinusfunktion

PE: FLO(HL)

PA: FLO(HL)

CY=0 Argument war zu groß

UR: HL

Funktion: $FLO(HL) = SIN (FLO(HL))$ (siehe auch BD97 FLO DEG/RAD)

BDAF FLO COS

Cosinusfunktion

PE: FLO(HL)

PA: FLO(HL)

CY=0 Argument war zu groß

UR: HL

Funktion: $FLO(HL) = COS (FLO(HL))$ (siehe auch BD97 FLO DEG/RAD)

BDB2 FLO TAN

Tangensfunktion

PE: FLO(HL)

PA: FLO(HL)

CY=0, Z=1 Division durch Null

CY=0, S=1 Argument war zu groß

UR: HL

Funktion: $FLO(HL) = TAN (FLO(HL))$ (siehe auch BD97 FLO DEG/RAD)

BDB5 FLO ARC TAN

Arkustangensfunktion

PE: FLO(HL)

PA: FLO(HL)

UR: HL

Funktion: $FLO(HL) = ARCTAN (FLO(HL))$ (siehe auch BD97 DEG/RAD)

BD79 FLO 10 A Exponentialfunktion zur Basis 10

PE: FLO(HL),A

PA: FLO(HL)

CY=0 Überlauffehler
UR: HL

Funktion: $FLO(HL) = FLO(HL) * 10^A$

BD94 FLO SGN Signumfunktion

PE: FLO(HL)
PA: A=-1, CY=1 wenn FLO(HL)=negativ
A=0, Z=1 wenn FL(HL)=null
A=1 wenn FLO(HL)=positiv
UR: BC,DE,HL,IY,FLO(HL)

Funktion: $A = SGN (FLO(HL))$

BD61 FLO MOVE Verschiebefunktion

PE: FLO(DE), FLO(HL)
PA: FLO(HL), A=Exponentbyte von FLO(DE), CY=1
UR: BC,DE,HL,IX,IY,FLO(DE)

Funktion: $FLO(HL) = FLO(DE)$

BD9A FLO PI Laden der Zahl PI

PE: FLO(HL)
PA: FLO(HL),CY=1
UR: BC,HL,IX,IY

Funktion: $FLO(HL) = PI$

BD97 FLO DEG/RAD Umstellung Winkelmaß -Bogenmaß

PE: A=0 Bogenmaß (RAD)
A>0 Winkelmaß (DEG)
PA: keine
UR: AF,BC,DE,HL,IX,IY

3.9.4.3. Die Integer-Routinen

Die Parameterübergabe erfolgt standardmäßig in den Doppelregistern HL und DE.

Berechnungen mit vorzeichenbehafteten Integer-Zahlen

DD4A INT ADD VZ Addition zweier Zahlen

PE: HL, DE
PA: HL
CY=0 Überlauffehler
UR: BC,DE,IX,IY

Funktion: $HL = HL + DE$ (muß über RST aufgerufen werden)

DD52 INT SUB*VZ Subtraktion zweier Zahlen

PE: HL, DE
PA: HL
DE=alter Wert von HL
CY=0 Überlauferfehler
UR: BC,IX,IY

Funktion: $HL = DE - HL$ (muß über RST aufgerufen werden)

DD53 INT SUB VZ Subtraktion zweier Zahlen

PE: HL, DE
PA: HL
CY=0 Überlauferfehler
UR: BC,DE,IX,IY

Funktion: $HL = HL - DE$ (muß über RST aufgerufen werden)

DD5B INT MULT VZ Multiplikation zweier Zahlen

PE: HL,DE
PA: HL
CY=0 Überlauferfehler
UR: DE,IX,IY

Funktion: $HL = HL * DE$
(muß über RST aufgerufen werden)

DD9C INT DIV VZ Division zweier Zahlen

PE: HL,DE
PA: HL,DE
CY=0 Überlauferfehler
UR: IX,IY

Funktion: $HL = HL / DE$
 $DE = HL \text{ MOD } DE$
(muß über RST aufgerufen werden)

DDA3 INT MOD VZ Division zweier Zahlen

PE: HL,DE
PA: HL,DE
CY=0 Überlauferfehler
UR: IX,IY

Funktion: $HL = HL \text{ MOD } DE$
 $DE = HL / DE$
(muß über RST aufgerufen werden)

DE02 INT VGL Vergleich zweier Zahlen

PE: HL,DE
PA: A=-1, CY=1 wenn HL - DE = negativ
A=0, Z=1 wenn HL - DE = null
A=1 wenn HL -DE = positiv
UR: BC,DE,HL,IX,IY
(muß über RST aufgerufen werden)

DDED INT VZW Vorzeichenwechsel einer Zahl

PE: HL
PA: HL
CY=0 Überlauferfehler
UR: BC,DE,IX,IY
(muß über RST aufgerufen werden)

DDF9 INT SGN Signumfunktion einer Zahl

PE: HL
PA: A=-1, CY=1 wenn HL = negativ
A=0, Z=1 wenn HL = null
A=1 wenn HL = positiv
UR: BC,DE,IX,IY
(muß über RST aufgerufen werden)
Berechnungen mit vorzeichenlosen Integer-Zahlen

DD72 INT MULT Multiplikation zweier Zahlen

PE: HL,DE
PA: HL
CY=1 Überlauferfehler
UR: BC,DE,IX,IY

Funktion: HL = HL * DE
(muß über RST aufgerufen werden)

DDAB INT DIV Division zweier Zahlen

PE: HL,DE
PA: HL,DE
CY=0,Z=0 Überlauferfehler
CY=0,Z=1 Division durch Null
UR: BC,IX,IY

Funktion: HL = HL / DE
DE = HL MOD DE
(muß über RST aufgerufen werden)

3.9.4.4. Konvertierungs-Routinen

Diese Routinen dienen der Konvertierung zwischen Integer- und Fließkommazahlen. Als Abkürzung für den Zeiger auf ein Doppelwort (Zeiger in HL) wird LW(HL) verwendet. Wobei LW für long word (Doppelwort) steht.

BD6A ROUND FLO TO HLA Fließkommazahl --> Integerzahl

PE: FLO(HL)
PA: HL=Absolutwert
 Bit 7,A = Vorzeichen
 CY=0 Überlauferfehler
UR: BC,DE,IY

BD64 KONV HLA TO FLO Integerzahl --> Fließkommazahl

PE: HL=Absolutwert
 BIT 7,A = Vorzeichen
 FLO(DE) (für Ergebnis)
PA: FLO(HL) (HL=alter Wert von DE)
UR: BC,IX,IY

BD67 KONV LW TO FLO Doppelwort --> Fließkommazahl

JPE: LW(HL),BIT 7,A =Vorzeichen
PA: FLO(HL)
UR: BC,DE,HL,IY

BD6D ROUND FLO TO LW Fließkommazahl --> Doppelwort
mit Rundung der Zahl

PE: FLO(HL)
PA: LW(HL) = Absolutwert
 Bit 7,B = Vorzeichen
 CY=0 Überlauferfehler
UR: DE,HL,IY

Siehe BASIC-Kommando ROUND.

BD70 FIX FLO TO LW Fließkommazahl --> Doppelwort
ohne Rundung der Zahl

PE: FLO(HL)
PA: LW(HL) = Absolutwert
 Bit 7,B = Vorzeichen
UR: DE,HL,IY

Siehe BASIC-Kommando FIX.

BD73 INT FLO TO LW Fließkommazahl --> Doppelwort

PE: FLO(HL)
 PA: LW(HL) = Absolutwert
 Bit 7, B = Vorzeichen
 CY=0 Überlauferfehler
 UR: DE, HL, IY

Siehe BASIC-Kommando INT.

BDB8 KONV LW+C TO FLO "5-Byte-Wort" --> Fließkommazahl

PE: LW(HL), C
 PA: FLO(HL) = LW(HL)*256 +C
 UR: DE, HL, IY

DD37 KONV HLB TO INT Komplementärbildung

PE: HL=Absolutwert
 Bit 7, B = Vorzeichen
 PA: HL = Zahl in Komplementärdarstellung
 CY=0 Überlauferfehler
 UR: BC, DE, IX, IY

Dezimalumwandlung

BD76 FLO PREPARE Fließkommazahl --> Dezimalzahl

PE: FLO(HL)
 PA: LW(HL) = normierte Mantisse
 B = Vorzeichen Mantisse
 D = Vorzeichen Exponent
 E = Exponent bzw. Kommazifferposition
 C = Zahl signifikanter Mantissen-Bytes
 UR: HL

DD2A INT PREPARE VZ Integerzahl --> Dezimalzahl `mit
 Vorzeichen

PE: HL =Zahl in Komplementärdarstellung
 PA: HL =Absolutwert
 Bit 7, B =Vorzeichen
 C =2
 E =0
 UR: IX, IY

DD39 INT PREPARE Integerzahl --> Dezimalzahl ohne
 Vorzeichen

PE: HL=positive Zahl
 PA: HL=Absolutwert

B =0
 C =2
 E =0
 UR: AF,D,IX,IY

3.10. Arbeitszellen

Alle Teile des Betriebssystems und der BASIC-Interpreter beanspruchen für ihre Nutzung Arbeitszellen. In den folgenden Tabellen sind die wichtigsten Zellen und Bereiche zusammengefaßt.

3.10.1. Betriebssystem-Arbeitszellen

KERNEL

Adresse/Bereich	Bedeutung
B82D,B82E	Anfang der Asynchronous Pending Queue
B82F,B830	letzter Block in der Pending Queue
B831 Flag	bei Queue-Bearbeitung
B832,B833	Zwischenspeicher für Stackpointer bei der Queue-Bearbeitung
B834-B8B3	Stackbereich für Queue-Bearbeitung
B8B4-B8B7	TIME (Doppelwort der internen Uhr)
B8B8	Sperrbyte bei Überlauf der inneren Uhr
B8B9,B8BA	Anfang der Frame Flyback Chain
B8BB,B8BC	Anfang der Fast Ticker Chain
B8BD,B8BE	Anfang der Ticker Chain
B8BF	Zähler für Fast-Ticker (jeder 6. --> Ticker)
B8C0,B8C1	Anfang der Synchronous Pending Queue
B8C2	aktuelle Synchronous Event-Priorität
B8C3-B8D2	RSX-Name bei BCD4 KL FIND COMMAND
B8D3,B8D4	Anfang der External Command Chain
B8D5	aktuelle RAM-Konfiguration
B8D6	aktuelle ROM-Konfiguration
B8D7,B8D8	Startadresse des laufenden Vordergrund-Programms
B8D9	ROM-Konfiguration des laufenden Vordergrund-Programms
B8DA-B8F9	IY-Bereich für die Hintergrund-ROMs (0-15)
B8FA-B8FF	unbenutzt

MACHINE PACK

Adresse/Bereich	Bedeutung
B804-B82C	Drucker-Übersetzungstabelle

SCREEN PACK

Adresse/Bereich	Bedeutung
B7C3	Bildschirmmodus
B7C4, B7C5	Hardware-Scroll-Offset
B7C6	High-Teil der Bildwiederholungspeicher-Anfangs- adresse (00H, 40H, 80H, C0H)
B7C7-B7C9	Modus beim Punktesetzen (deckend, AND, XOR, OR)
B7CA-B7D1	unbenutzt
B7D2	Periode 1 für Farbblinken
B7D3	Periode 0 für Farbblinken
B7D4-B7E4	Paletten-Farbwerte aller Tinten und von Border der Blinkperiode 1
B7E5-B7F5	Paletten-Farbwerte aller Tinten und von Border der Blinkperiode 0
B7F6	aktueller Farbsatz
B7F7	Flag für neue Farben in der Tabelle
B7F8	Zähler für aktuelle Blinkperiode
B7F9-B801	Frame Flyback Block für Farbblinken
B802, B803	bei verschiedenen Grafik-Routinen verwendet

TEXT VDU

Adresse/Bereich	Bedeutung
B6B5	aktuelles Textfenster
B6B6-B6C3	Parameter Fenster 0
B6C4-B6D1	Parameter Fenster 1
B6D2-B6DF	Parameter Fenster 2
B6E0-B6ED	Parameter Fenster 3
B6EE-B6FB	Parameter Fenster 4
B6FC-B709	Parameter Fenster 5
B70A-B717	Parameter Fenster 6
B718-B725	Parameter Fenster 7
B726-B733	Parameter des aktuellen Textfensters
B726	Cursor-Zeile (ganz oben Zeile 0)
B727	Cursor-Spalte (ganz links Spalte 0)
B728	=0 bei Hardwarescroll, =FFH bei Softwarescroll
B729	Fenstergrenze oben
B72A	Fenstergrenze links
B72B	Fenstergrenze unten
B72C	Fenstergrenze rechts
B72D	Scroll-Zähler
B72E	Bit 0 =0 Cursor erlaubt, =1 Cursor verboten Bit 1 =0 Cursor ein, =1 Cursor aus Bit 7 =0 Textausgabe erlaubt, =1 verboten
B72F	Farbbyte der Vordergrund-Tinte
B730	Farbbyte der Hintergrund-Tinte
B731, B732	Routinenadresse entsprechend Hintergrundmodus
B733	=0 Textausgabe auf Textcursor, >0 Textausgabe

Adresse/Bereich	Bedeutung
	auf Grafikkursor
B734	Code (ASCII) der ersten Zeichenmatrix im RAM
B735	Zeichentabelle im RAM: =0 nein, =FFH ja
B736,B737	Anfangsadresse der Zeichentabelle im RAM
B738-B757	Puffer für expandierte Zeichenmatrix
B758	Anzahl Zeichen im Control-Code-Puffer
B759-B762	Control-Code-Puffer
B763-B7C2	Control-Code-Tabelle (Routinenadressen und Anzahl benötigter Parameter)

GRAPHICS VDU

Adresse/Bereich	Bedeutung
B693,B694	Ursprung-X-Koordinate (Origin)
B695,B696	Ursprung-Y-Koordinate (Origin)
B697,B698	Grafik-Cursor-X-Koordinate
B699,B69A	Grafik-Cursor-Y-Koordinate
B69B,B69C	Grafik-Fenstergrenze:links
B69D,B69E	Grafik-Fenstergrenze:rechts
B69F,B6A0	Grafik-Fenstergrenze:oben
B6A1,B6A2	Grafik-Fenstergrenze:unten
B6A3	Farbbyte der Vordergrund-Tinte
B6A4	Farbbyte der Hintergrund-Tinte
B6A5-B6B1	Bereich für verschiedene Zwecke (FILL, DRAW usw.)
B6B2	Erster-Punkt-Option
B6B3	Maske für Linienzeichnen

KEYBOARD MANAGER

Adresse/Bereich	Bedeutung
B6B4	Hintergrund-Modus: =0 deckend, =FFH transparent
B496-B4E5	Übersetzungstabelle für Tasten ohne [SHIFT] und [CTRL](ASCII-Codes)
B4E6-B535	Übersetzungstabelle für Tasten mit [SHIFT]
B536-B585	Übersetzungstabelle für Tasten mit [CTRL]
B586-B58F	Tasten-Repeat-Tabelle
B590-B627	Erweiterungszeichen-Puffer
B628	Zähler für Erweiterungszeichenkette
B629	Nummer des aktuellen Erweiterungszeichens
B62A	Puffer für zurückgegebene Zeichen
B62B,B62C	Zeiger auf Erweiterungszeichenketten-Puffer
B62D,B62E	Zeiger auf Ende des Erweiterungszeichenketten-Puffers
B62F,B630	Zeiger auf erstes freies Byte im Erweiterungszeichenketten-Puffer

Adresse/Bereich	Bedeutung
B631	Bit 7 =0 SHIFT LOCK aus, =1 SHIFT LOCK ein
B632	Bit 7 =0 CAPS LOCK aus, =1 CAPS LOCK ein
B633	Verzögerungszeit beim ersten Repeat
B634	Verzögerungszeit für die weiteren Repeats
B635-B652	Tabellen für aktuell gedrückte Tasten
B653	Zähler für Repeat
B654,B655	Informationen für aktuelle Taste
B656	=0 Break-Mechanismus aus, >0 ein
B657-B65D	Break-Event-Block
B65E-B685	Informationen für Tasten in der Warteschlange
B686-B68A	Parameter zur Verwaltung der Warteschlange
B68B,B68C	Zeiger auf Übersetzungstabelle ohne [SHIFT] oder [CTRL]
B68D,B68E	Zeiger auf Übersetzungstabelle mit [SHIFT]
B68F,B690	Zeiger auf Übersetzungstabelle mit [CTRL]
B691,B692	Zeiger auf Tasten-Repeat-Tabelle

SOUND MANAGER

Adresse/Bereich	Bedeutung
B1ED	alte Kanalaktivität (für SOUND CONTINUE)
B1EE	aktuelle Kanalaktivität
B1EF	1/3-Zähl-Byte für Sound-Chain
B1F0	Kanalarbeitungsflag
B1F1-B1F7	Event-Block Tonausgabe
B1F8-B236	Parameterblock Kanal A
B237-B275	Parameterblock Kanal B
B276-B2B4	Parameterblock Kanal C
B2B5	Kontroll-Register des Soundcontrollers
B2B6-B3A5	Volumen-Hüllkurven 1-15
B3A6-B495	Frequenz-Hüllkurven 1-15

CASSETTE MANAGER

Adresse/Bereich	Bedeutung
B118	Meldungen ausgeben (=0), unterdrücken (=FFH)
B119	Meldungen komplett (=0), zerteilt (=FFH) ausgeben
B11A	Status Eingabedatei
B11B,B11C	Anfangsadresse Eingabepuffer
B11D,B11E	Zeiger im Eingabepuffer
B11F-B15E	Header-Puffer Eingabedatei
B15F	Status Ausgabedatei
B160,B161	Anfangsadresse Ausgabepuffer
B162,B163	Zeiger im Ausgabepuffer
B164-B1A3	Header-Puffer Ausgabedatei
B1A4-B1E3	Puffer für neu gelesenen Header

Adresse/Bereich	Bedeutung
B1E4	Bit 0=0 Eingabe nicht aktiv, =1 aktiv Bit 1=0 Ausgabe nicht aktiv, =1 aktiv
B1E5	Synchronisationszeichen
B1E6-B1E8	Zwischenspeicher für unterschiedliche Aufgaben
B1E9	Kompensationswert beim Schreiben
B1EA	Ausgabegeschwindigkeit
B1EB, B1EC	Prüfwort

3.10.2. BASIC-Interpreter-Arbeitszellen

Adresse/Bereich	Bedeutung
AC00	Flag für Space-Unterdrückung bei der Umwandlung im Token
AC01	Flag für AUTO
AC02, AC03	aktuelle Zeilennummer für AUTO
AC04, AC05	Schrittweite für AUTO
AC06	aktueller Ausgabe-Stream
AC07	aktueller Eingabe-Stream
AC08	aktuelle X-Position auf dem Drucker
AC09	WIDTH
AC0A	aktuelle X-Position in der Ausgabedatei
AC0B	ON BREAK CONT-Flag (0=aktiv)
AC0C	NEXT-Behandlungs-Flag
AC0D-AC11	Speicher für Startwert in FOR-NEXT-Schleife
AC12, AC13	Zeiger hinter zugehöriges NEXT
AC14, AC15	Zeiger auf Zeile mit zugehörigem WEND
AC16	Synchron-Event-Flag
AC17-AC1B	ON BREAK GOSUB-Parameterblock
AC17	alte Priorität
AC18, AC19	BASIC-Rücksprungadresse (PC im BASIC-Programm)
AC1A, AC1B	BASIC-Unterprogrammadresse
AC1C, AC1D	Zeiger auf Routinenadresse im Break-Event-Block
AC1E-AC29	Bereich für ON SQ(1) GOSUB-Routine
AC1E-AC24	Event-Block
AC25-AC29	Parameterblock (wie bei ON BREAK GOSUB)
AC2A-AC35	Bereich für ON SQ(2) GOSUB-Routine
AC36-AC41	Bereich für ON SQ(4) GOSUB-Routine
AC42-AC53	Bereich für EVERY/AFTER GOSUB Priorität 0
AC42-AC4E	Ticker Chain Block
AC4F-AC53	Parameterblock (wie bei ON BREAK GOSUB)
AC54-AC65	Bereich für EVERY/AFTER GOSUB Priorität 1
AC66-AC77	Bereich für EVERY/AFTER GOSUB Priorität 2
AC78-AC89	Bereich für EVERY/AFTER GOSUB Priorität 3
AC8A-AD8B	Puffer für INPUT und LIST
AD8C, AD8D	Zeilenadresse des letzten Fehlers (ERL)
AD8E, AD8F	Statement-Adresse des letzten Fehlers
AD90	Nummer des letzten Fehlers (ERR)

Adresse/Bereich	Bedeutung
AD91	Fehlernummer (DERR)
AD92,AD93	Statement-Adresse nach BREAK für CONT
AD94,AD95	Zeilenadresse nach BREAK für CONT
AD96,AD97	Adresse des BASIC-Programms für ON ERROR GOTO
AD98	Flag für ON ERROR (=FFH gerade in Abarbeitung)
AD99-ADA1	SOUND-Parameter-Puffer
ADA2-ADB1	ENV-und ENT-Parameter-Puffer
ADB2-ADB6	Zwischenspeicher beim Potenzieren
ADB7-ADEA	Start-Pointer der Verkettungslisten normaler Variablen (26 je Variablentyp)
ADEB,ADEC	Start-Pointer der Verkettungsliste für DEF FN
ADED,ADEE	Start-Pointer der Verkettungsliste für Real-Variablenfelder
ADEF,ADF0	Start-Pointer der Verkettungsliste für Integer-Variablenfelder
ADF1,ADF2	Start-Pointer der Verkettungsliste für String-Variablenfelder
ADF3-AE0C	Standard-Variablentyp DEFINT, DEFREAL oder DEFSTR (für alle Anfangsbuchstaben =26 Stück)
AE0D	Flag für Dimensionierung von Feldern
AE0E-AE13	Zeiger beim Auswerten von Ausdrücken
AE14	Flag für CR/LF nach INPUT
AE15,AE16	aktuelle DATA-Zeile
AE17,AE18	DATA-Zeiger
AE19,AE1A	BASIC-Stackpointer zum Statement-Anfang
AE1B,AE1C	aktuelle Statementadresse
AE1D,AE1E	aktuelle BASIC-Zeilenadresse
AE1F	=0 TROFF, =FFH TRON
AE20	Flag für Tokenbildung
AE21	=0 keine Zeilenadressen im Programm
AE22-AE25	DELETE-Parameter
AE26,AE27	Startadresse beim Laden von Programmen
AE28	CHAIN/CHAIN MERGE-Flag
AE29	File-Typ-Speicher
AE2A,AE2B	File-Länge
AE2C	Flag für geschütztes BASIC-Programm
AE2D-AE51	Zahlenwandlungs-Puffer
AE52-AE54	Speicher für Zahlenwandlung
AE55-AE57	far adress für CALL-oder RSX-Aufruf
AE58,AE59	Speicher für BASIC-Programmzeiger bei CALL/RSX
AE5A,AE5B	Speicher für den SP der CPU bei CALL/RSX
AE5C	ZONE-Wert
AE5D	Ende-Flag des Format-Strings bei PRINT USING
AE5E,AE5F	HIMEM-Systemspeicher
AE60,AE61	BASIC-RAM-Ende
AE62,AE63	BASIC-RAM-Anfang
AE64,AE65	BASIC-Programm-Anfang
AE66,AE67	BASIC-Programm-Ende
AE68,AE69	Variablenbereich-Anfang
AE6A,AE6B	Variablenfelder-Anfang
AE6C,AE6D	Variablenfelder-Ende

Adresse/Bereich	Bedeutung
AE6E	Flag für geschützten Variablenbereich
AE6F-B06E	BASIC-Stack
B06F,B070	Stackpointer BASIC-Stack
B071,B072	Anfang Stringbereich
B073,B074	Ende Stringbereich
B075	Flag für den I/O-Puffer (Bit 0=1 Eingabe aktiv, Bit 1=1 Ausgabe aktiv, Bit 2=1 Puffer reserviert)
B076,B077	Zeiger auf I/O-Puffer
B078-B07B	Zwischenspeicher bei Änderungen von HIMEM
B07C,B97D	Stackpointer im String-Descriptor-Stack
B07E-B09B	String-Descriptor-Stack
B09C-B09E	String-Descriptor-Puffer
B09F	BASIC-Akku-Typ (Real, String,Integer)
B0A0-B0A4	Akku bei der Auswertung von Ausdrücken (Integer, Real, String-Descriptor-Zeiger)
B0A5-B0FF	unbenutzt

FLOATING POINT PACK

Adresse/Bereich	Bedeutung
B100-B103	Zufallszahl
B104-B112	Zwischenspeicher für drei Fließkommazahlen
B113	=0 Bogenmaß, >0 Winkelmaß

Der Editor

Adresse/Bereich	Bedeutung
B114	Copy-Cursor-Flag
B115	Insert-Flag
B116,B117	Copy-Cursor-Koordinaten

3.11. Patchen von Vektoren

Fast alle Routinen des Betriebssystems benutzen die Vektoren der Sprungliste. Dies sollte auch in allen Anwenderprogrammen ausgenutzt werden. Dadurch ist das Betriebssystem sehr leicht auf eigene Belange anpaßbar. Will man z.B. eine Druckeroutine benutzen, die alle acht Bit des CENTRONICS Interfaces ansteuert, dann braucht man nur dafür zu sorgen, daß ein Aufruf von BD2B MC PRINT CHAR auf die eigene Druckeroutine zugreift. Alle anderen Routinen des Betriebssystems und des BASIC benutzen dann die neue Druckeroutine. Das Ändern von Vektoren der Sprungliste wird auch Patchen genannt. Will man den ursprünglichen Zustand der Vektoren wieder herstellen, sind zwei Wege möglich. Zum einen kann man den ursprünglichen Inhalt eines Vektors in einer

Kopie ablegen und den Vektor mit Hilfe dieser Kopie wieder restaurieren. Zum anderen werden durch den Aufruf von BD37H JMP RESTORE alle Sprungvektoren initialisiert.

Das folgende Programm zeigt, wie die Vektoren BD31H MC SEND PRINT und BDF1H IND MC WAIT PRINTER gepatcht werden. Die Ein- und Ausgabeparameter der Drucker-Routinen werden durch die zusätzliche Bit-Schalt-Routine nicht verändert.

```
;
;Vektoren patchen
;
PATCH: LD    HL,(0BDF2H) ;IND MC WAIT PRINTER
        LD    (ALT1+1),HL ;alte Adresse hinter die neue Routine
        DI                                ;zur Sicherheit
        LD    HL,PRINT1  ;neue Adresse in
        LD    (0BDF2H),HL ;den Vektor eintragen
        LD    HL,(0BD32H) ;MC SEND PRINT
        LD    (ALT2+1),HL ;alte Adresse hinter die neue Routine
        LD    HL,PRINT2  ;neue Adresse in
        LD    (0BD32H),HL ;den Vektor eintragen
        LD    A,0C3H     ;Restart-Befehl durch
        LD    (0BD31H),A ;Jump-Befehl ersetzen
;
;BD2BH MC PRINT CHAR wandelt einige Codes größer 80H
;deshalb muß eine Null an den Anfang der Code-Wandel-Tabelle
;gespeichert werden (BD2BH MC PRINT CHAR verwendet BDF1 IND MC
;WAIT PRINTER)
;
        XOR   A
        LD   (0B804H),A
        EI
        RET
;
;neue Routine IND MC WAIT PRINTER
;
PRINT1: CALL  SCHALT
ALT1:   JP    0           ;hier wird alte Adresse eingetragen
;
;neue Routine MC SEND PRINT
;
PRINT2: CALL  SCHALT
ALT2:   RST   8
        DEFS 2           ;hier wird alte Adresse eingetragen
;
;Schalt-Routine für das 8. Bit
;
SCHALT: PUSH  BC
        LD   BC,0F70BH   ;PIO-Port C Bit 5 setzen
        BIT  7,A
        JR   NZ,SCH1
        LD   C,0AH       ;PIO-Port C Bit 5 rücksetzen
SCH1:   OUT  (C),C
        POP  BC
        RET
```

4. SOFTWARE - BASIC - INTERPRETER

4.1. Einleitung

Ein Interpreter arbeitet ein Programm ab, indem er Zeichen einer Datei liest und deutet (interpretiert). Die Datei in BASIC ist eine Aneinanderreihung von Befehlen, Zahlen, Variablen und Zeichenketten. Für jeden Befehl ist in der Textdatei ein Byte, das sogenannte Token (siehe Anhang D), eingetragen. Mit Hilfe dieser Token entscheidet der Interpreter, welche Aktionen ausgeführt werden sollen. Noch aufwendiger wird es, wenn mit Variablen und Sprungbefehlen gearbeitet wird. Findet der BASIC-Interpreter z.B. eine Variable im Quelltext, muß er diese erst im Variablenspeicher suchen, um deren Wert zu ermitteln. Ähnlich verhält es sich bei Sprungbefehlen und Unterprogrammaufrufen. Die Zeilennummer, die im Text angegeben ist, muß erst in eine Adresse umgerechnet werden. Dazu muß der Text nach der gewünschten Zeilennummer durchsucht werden. Das BASIC im KC compact hat gegenüber anderen BASIC-Interpretern den großen Vorteil, daß die Adressen von Variablen und Zeilen, zu denen gesprungen wird, nur einmal pro Durchlauf bestimmt werden. Die ermittelten Adressen werden anstelle der Zeilennummern bzw. zusätzlich bei den Variablen in den Quelltext eingetragen. Wird dann diese Programmsequenz noch einmal abgearbeitet, kann sofort auf diese zurückgegriffen werden.

4.2. Speicheraufteilung bei der Arbeit mit BASIC

In der folgenden Darstellung wird die prinzipielle Aufteilung des gesamten BASIC-RAM deutlich:

```
B100H -----  
Arbeitszellen, Zeiger und Flags des BASIC-Interpreters  
AC00H -----  
in diesem Speicherbereich können sich Hintergrund-ROMs  
Platz reserviert haben, die Länge des Bereichs ist nicht  
begrenzt und hängt vom Hintergrundprogramm ab  
-----  
in diesem Speicherbereich können verschiedene Programme,  
Puffer und Tabellen liegen (siehe Abschn. 4.5.)  
-----  
HIMEM -----  
Stringspeicherbereich  
-----  
freier Speicherbereich  
-----  
Bereich der Variablen-Felder  
-----  
Bereich der Variablen  
-----  
Quelltext des BASIC-Programms
```

```
016FH -----  
      Puffer zur Umwandlung in Token  
0040H -----
```

Abb. 4.1: Speichereinteilung in BASIC

Da die RAM-Verwaltung dynamisch ist, die Grenzen sich im Laufe der Zeit verschieben, können für die Grenzen zwischen den Bereichen keine Absolutwerte angegeben werden. In den BASIC-Arbeitszellen existieren deshalb eine Reihe von Zeigern, die auf diese Grenzen zeigen (siehe Abschn. 3.10.2.).

4.3. Der Aufbau eines BASIC-Programms

Ein BASIC-Programm besteht aus BASIC-Zeilen, die folgenden Aufbau haben:

```
Byte 1 und 2   : Länge der gesamten Zeile (n Bytes)  
Byte 3 und 4   : Zeilennummer  
Byte 5 bis n-1 : Quelltext der BASIC-Zeile (mit Token)  
Byte n        : Endemarkierung, ist immer Null
```

Die BASIC-Zeile "10 CLS" würde z.B. mit folgenden Bytes in einem Programm abgelegt:

```
06H 00H 0AH 00H 8AH 00H
```

Ein BASIC-Programm hat folgenden Aufbau:

```
ein Null-Byte  : Programmstartmarkierung  
  Zeile 1  
  Zeile 2  
  ...  
  Zeile n  
zwei Null-Bytes : Programmendemarkierung  
1. Byte des Variablenspeichers
```

4.4. Variablentypen

In BASIC sind drei Variablentypen zugelassen, die Integer-, die Real- (Fließkomma-) und die String-Variablen. Zum Abspeichern einer Integerzahl werden 2 und zum Abspeichern einer Realzahl 5 Bytes benötigt. Die Anzahl der für einen String verbrauchten Bytes hängt von dessen Länge ab. Zusätzlich zur eigentlichen Länge werden noch 3 Bytes für einen Descriptor benötigt. In ihm ist die Länge und die Anfangsadresse des Strings gespeichert. Die Variablen mit ihren Namen sind im Variablenspeicher abgelegt. Eine Ausnahme bilden die String-Variablen. Im Variablen-

speicher sind nur der Name und der Descriptor abgelegt, der String selber liegt im Stringspeicher. Ein Variablen-Eintrag hat folgenden allgemeinen Aufbau:

- 2 Bytes : Adresse der nächsten Variablen mit dem gleichen Anfangsbuchstaben relativ zur Anfangsadresse des Variablenspeichers
- 1-40 Bytes : Name der Variablen, im letzten Zeichen ist Bit 7 gesetzt
- 1 Byte : Variablentyp (Integer=1, Real=4, String=2)
- 2, 3 oder
- 5 Bytes : Zahlenwert bzw. String-Descriptor

Ebenfalls im Variablenspeicher werden die User-Funktionen (DEF FN) abgelegt. Gekennzeichnet werden sie durch das Setzen des Bit 6 im Variablentyp-Byte. Anstelle des Zahlenwertes wird die Anfangsadresse der Funktions-Definition eingetragen.

Variablenfelder werden in einem extra reservierten Bereich oberhalb des Speichers für einfache Variablen abgelegt. Ein Feld hat folgenden allgemeinen Aufbau:

- 2 Bytes : Adresse des nächsten Feldes vom gleichen Typ relativ zur Anfangsadresse des Variablenfeldbereichs
- 1-40 Bytes : Name des Feldes, im letzten Zeichen ist Bit 7 gesetzt
- 1 Byte : Variablentyp (Integer=1, Real=4, String =2)
- 2 Bytes : Feldlänge (ab dem nächsten Byte gerechnet)
- 1 Byte : Anzahl der Dimensionen
- 2 Bytes : maximaler Index der ersten Dimension
- ...
- 2 Bytes : maximaler Index der letzten Dimension
- ff. : Daten des Feldes

4.5. Der BASIC-Stack

Unabhängig vom Stack des Betriebssystems wird von BASIC ein eigener Stack verwaltet. Er wird für verschiedene Programmstrukturen, z.B. FOR-NEXT-Schleifen, Unterprogrammaufrufe usw. und bei der Auswertung von arithmetischen Ausdrücken benötigt. Der BASIC-Stack-Bereich ist 512 Bytes (AE6FH-B06EH) lang. Außerdem existiert ein Zeiger (B06FH/B070H) auf den aktuellen Stackwert. Im Gegensatz zum normalen Stack des Prozessors wächst der BASIC-Stack von unten nach oben.

4.6. Die Verwaltung des HIMEM

Über dem HIMEM (obere Grenze des Stringspeichers) gibt es verschiedene Bereiche, die bei Bedarf eingerichtet werden. Nach

dem Kaltstart des KC compact werden z.B.automatisch die Zeichenbildmatrizen der letzten 16 ASCII-Zeichen unterhalb von HIMEM kopiert und HIMEM anschließend darunter gesetzt. Der Anwender hat nun die Möglichkeit weitere Bereiche für eigene Belange zu reservieren. Das können z.B. Maschinencodeprogramme oder weitere Zeichenbildmatrizen sein. Da die Zeichenbildmatrizen immer zusammenhängend in einer Tabelle gespeichert werden müssen, muß darauf geachtet werden, daß Zeichenbildmatrizen nur angefügt bzw. gelöscht werden können, wenn HIMEM sich gerade unterhalb der Zeichenbildtabelle befindet. Anderenfalls erscheint die Fehlermeldung "Improper argument".

Durch das Heruntersetzen des HIMEM mit MEMORY besteht die Möglichkeit, Maschinenprogramme über HIMEM nachzuladen, um sie von BASIC aus zu nutzen.

Der HIMEM wird ebenfalls bei der Eröffnung von Ein- oder/und Ausgabe-Dateien auf Kassette heruntergesetzt. BASIC reserviert für den Ein/Ausgabe-Puffer (I/O-Puffer) immer 4 KByte. Nach dem Schließen der Ein- oder/und Ausgabe-Datei wird der HIMEM wieder um 4 KByte nach oben gesetzt, wenn unterhalb des I/O-Puffers keine Zeichenbildtabelle oder/und Maschinencodeprogramme angelegt wurden.

Es sind eine Vielzahl von Kombinationen zur Einteilung des Bereiches über HIMEM möglich.

Beispiel: In einem BASIC-Programm sollen veränderte Zeichenmatrizen für die ASCII-Zeichen von 32 bis 255 benutzt werden. Weiterhin soll eine kleine Maschinenroutine genutzt werden und Datenfelder sollen auf Kassette ausgelagert werden. Folgende Programmzeilen stellen eine von sechs Möglichkeiten zur Einteilung des Bereiches über HIMEM dar:

```
10 SYMBOL AFTER 256 'Löschen der gesamten Zeichenbildtabelle im
    RAM
20 MEMORY HIMEM-100 '100 Bytes für Maschinenprogramm reservieren
30 LOAD "prog.bin",HIMEM+1 'Laden des Maschinenprogramms
40 OPENOUT "feld.dat" 'I/O-Puffer anlegen
50 SYMBOL AFTER 32 'Zeichenbildtabelle im RAM anlegen
```

4.7. BASIC und Maschinencode

Das BASIC im KC compact ist sehr komfortabel und leistungsfähig. Bei zeitkritischen Berechnungen jedoch ist eine Programmierung in Maschinensprache erforderlich. Für die meisten Probleme reicht eine Kombination von BASIC-Programm und Maschinencode aus. Es gibt mehrere Möglichkeiten, Maschinencode von BASIC aus zu nutzen. Eine sehr einfache Möglichkeit wurde bereits im Abschnitt 3.6. beschrieben. Die dort beschriebene RSX-Programmierung wird aber zumeist dann zur Anwendung kommen, wenn viele Unterprogramme in Maschinencode geschrieben sind. Soll aber nur ein oder sollen nur sehr wenige Unterprogramme genutzt werden, kann der Maschinencode in einen Bereich über HIMEM

geladen werden und direkt über CALL von BASIC aus aufgerufen werden. Die Übergabebedingungen entsprechen denen der RSX-Kommandos (siehe Abschn. 3.6.). Bei Aufruf über CALL muß die Einsprungadresse bekannt sein. Ein Nachteil beider Varianten ist es, daß die Programme meist an eine feste Adresse gebunden sind, da die Verwendung von Absolutadressen in längeren Programmen meist nicht vermieden werden kann. Die gleichzeitige Nutzung mehrerer solcher voneinander unabhängiger Programmpakete, deren Speicherbereiche sich überschneiden, ist deshalb nicht möglich. Ein Ausweg ist, die Programme auf den Adreßbereich anzupassen, auf den sie geladen werden. Dazu sind alle Absolutadressen umzurechnen. Das kann durch eine kleine Initialisierungsroutine (Relocater) im Programm selber realisiert werden. Diese Routine muß nach dem Laden des Maschinenprogramms als erstes aufgerufen werden. Geladen werden können die Maschinenprogramme entweder vom Massenspeicher (Kassette oder Diskette) oder sie sind in DATA-Zeilen abgelegt und werden vom BASIC-Programm ausgelesen und mit POKE in den Zieladreßbereich gespeichert. Letztere Methode kommt meist wegen des großen Speicherbedarfs und der langen Ladezeit (READ und POKE) nur für kurze Programme in Frage.

Eine dritte Möglichkeit sehr kurze Maschinenprogramme zu nutzen besteht darin, den Code in einem String abzulegen. Voraussetzung dafür ist, daß das Programm keine Absolutadressen enthält. Zwei Probleme sind bei dieser Möglichkeit zu lösen. Zum Einen muß das Programm in den String geladen werden, und zum Zweiten muß die Routine von BASIC aus aufgerufen werden.

Eine Möglichkeit zur Lösung des ersten Problems wäre:

- einen String mit Maschinencode-Länge definieren (enthaltene Zeichen spielen keine Rolle),
- aus dem Stringdescriptor die Anfangsadresse des Strings ermitteln (Descriptoradresse wird bereitgestellt, wenn vor dem Variablennamen ein angegeben wird) und
- Laden des Codes aus DATA-Zeilen in den Adreßbereich des Strings.

Zum Aufruf der Maschinencoderoutine muß dessen Anfangsadresse jedesmal neu aus dem Stringdescriptor ermittelt werden. Neben dem Vorteil, daß sich solche Maschinenroutinen nicht überschneiden können, können Routinen, die nicht mehr benötigt werden, jederzeit gelöscht werden.

Die letzte hier erwähnte Möglichkeit besteht darin, Maschinencode in nicht verwendeten BASIC-Zeilen (werden bei der Abarbeitung übersprungen) oder in Kommentar-Zeilen zu speichern. Dazu muß die genaue Lage der zu verändernden Zeilen bekannt sein.

Diese Methode hat den Vorteil, daß die Maschinencodeprogramme in abarbeitsfähigem Zustand mit dem BASIC-Programm geladen und gerettet werden können.

5. L i t e r a t u r h i n w e i s e

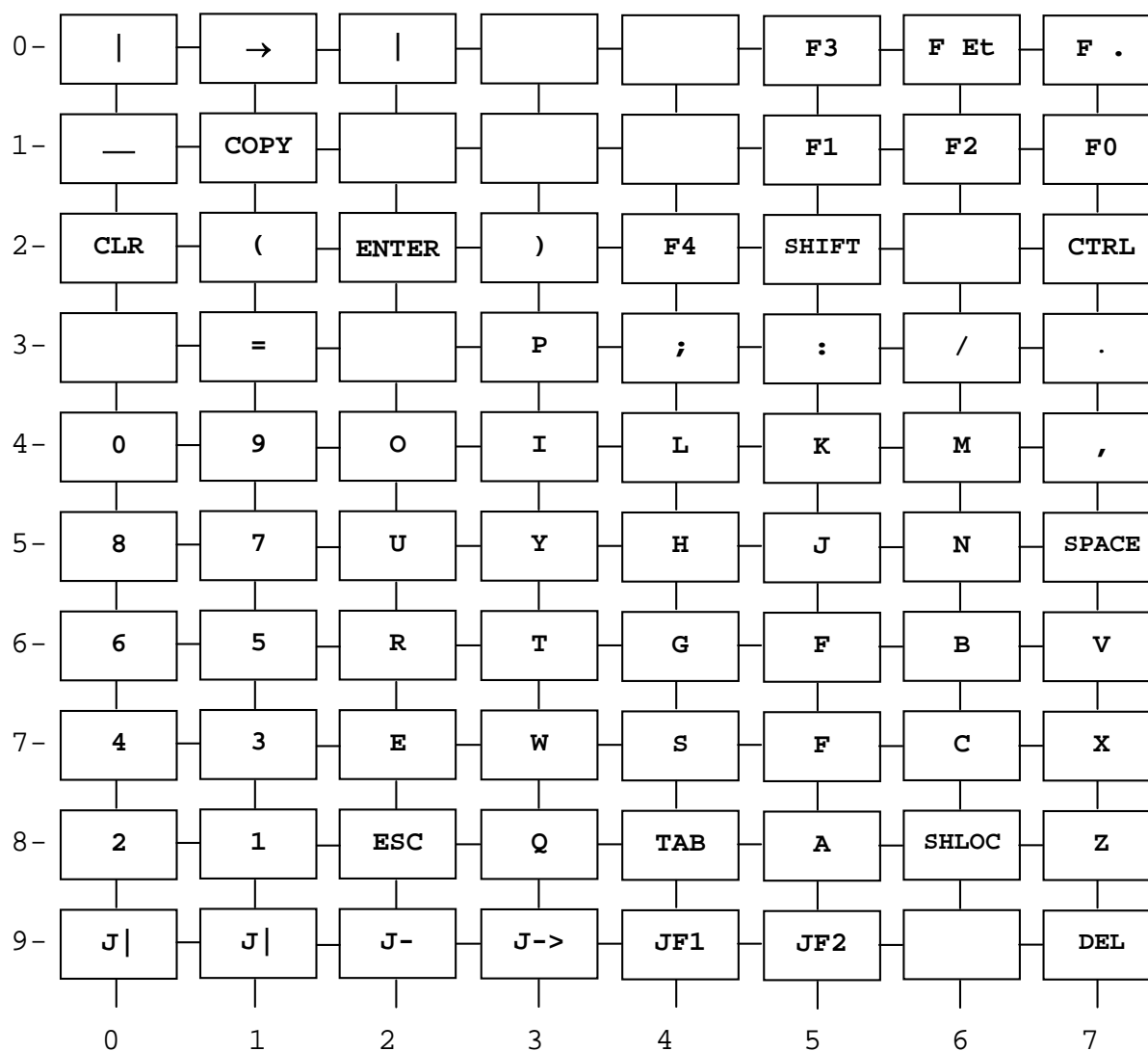
- /1/ H.Kieser, M.Meder; Mikroprozessortechnik, VEB Verlag Technik, Berlin 1985
- /2/ M.Roth; Mikroprozessoren, Wiss. Zeitschrift und KdT Hochschulsektion TH Ilmenau, DDR 1980
- /3/ M.Kramer; Praktische Mikrocomputertechnik, Militärverlag der Deutschen Demokratischen Republik, Berlin 1987
- /4/ MC 6845, CRT Controller, Advance Information, Motorola
- /5/ F.Claßen, H.Oeffler; Wissenspeicher Mikrorechenprogrammierung, VEB Verlag Technik Berlin, 1989
- /6/ AY 3-89-12A, Programmable Sound Generator, Datenblatt, General Instrument Corporation 1985
- /7/ BASIC-Handbuch KC compact, VEB Mikroelektronik "Wilhelm Pieck" Mühlhausen
- /8/ Gerätebeschreibung KC compact, VEB Mikroelektronik "Wilhelm Pieck" Mühlhausen

Anhang A Steuercodes

Code hex dez	Name	Para- meter	Beschreibung
0 0	NUL	0	Dummy-Zeichen
1 1	SOH	1	Bildschirmausgabe eines Zeichens, Steuer- codes werden als Grafikzeichen dargestellt
2 2	STX	0	Cursor ausschalten
3 3	ETX	0	Cursor einschalten
4 4	EOT	1	Bildschirmmodus einstellen
5 5	ENQ	1	Zeichen auf Grafikcursorposition ausgeben
6 6	ACK	0	Zeichenausgabe im Textfenster zulassen
7 7	BEL	0	Ausgabe Warnton
8 8	BS	0	Cursor eine Position zurück
9 9	TAB	0	Cursor eine Position vor
A 10	LF	0	Cursor eine Zeile runter
B 11	VT	0	Cursor eine Zeile hoch
C 12	FF	0	Bildschirm löschen und Cursor in linke, obere Ecke setzen
D 13	CR	0	Cursor in erste Spalte der aktuellen Zeile setzen
E 14	SO	1	Hintergrund-Tinte festlegen
F 15	SI	1	Vordergrund-Tinte festlegen
10 16	DLE	0	Zeichen auf Cursorposition löschen
11 17	DC1	0	Zeile von Anfang bis Cursorposition löschen
12 18	DC2	0	Zeile von Cursorposition bis Ende löschen
13 19	DC3	0	Textfenster von oben zeilenweise bis Cursorposition löschen
14 20	DC4	0	von Cursorposition bis Textfensterende zeilenweise löschen
15 21	NAK	0	Zeichenausgabe im aktuellen Textfenster verbieten
16 22	SYN	1	Transparentmodus ein/aus (1/0)
17 23	ETB	1	Grafikvordergrundmodus festlegen (0=deckend, 1=XOR, 2=AND, 3=OR)
18 24	CAN	0	Vorder- und Hintergrund-Tinte tauschen
19 25	EM	9	Zeichenmatrix für ein Zeichen festlegen 1.Parameter=Zeichen 2.-9.Parameter=Matrixbytes
1A 26	SUB	4	Textfenstergrenzen festlegen 1.Parameter=linker Rand 2.Parameter=rechter Rand 3.Parameter=oberer Rand 4.Parameter=unterer Rand
1B 27	ESC	0	wird ignoriert
1C 28	FS	3	Farben einer Tinte festlegen 1.Parameter=Tinten-Nr. 2./3.Parameter=Farbwerte
1D 29	GS	2	Farben für Border festlegen
1E 30	RS	0	Cursor in linke, obere Ecke setzen
1F 31	US	2	Cursorposition festlegen

Anhang B Tastaturmatrix

Ein 1-aus 10-Dekoder legt die angewählte Zeilenleitung auf Low-Potential und läßt alle anderen Leitungen im hochohmigen Zustand. Durch den Druck auf eine Taste werden die Spalten- und die Zeilenleitung miteinander verbunden, deren Kreuzungspunkt in der Tastaturmatrix durch die entsprechende Taste belegt ist. Alle Spaltenleitungen werden durch Zieh Widerstände auf High-Potential gehalten und nehmen nur dann Low-Potential an, wenn sie durch eine gedrückte Taste mit einer Zeilenleitung verbunden werden, die auf Low-Potential liegt. Will man eine bestimmte Taste direkt abfragen, muß die entsprechende Zeilennummer auf Bit 0 bis Bit 3 des Port C der PIO ausgegeben werden, die Spalteninformation der Tastaturmatrix über den Soundcontroller und PIO-Port A eingelesen und das eingelesene Byte mit dem Sollwert verglichen werden.



Anhang C Vektoradressen

(alle Adressen sind hexadezimal angegeben)

Adr.Bezeichnung	Adr.Bezeichnung
0000 LOW RESET ENTRY	BB30 KM GET SHIFT
0008 LOW LOW JUMP	BB33 KM SET CONTROL
000B LOW KL LOW PCHL	BB36 KM GET CONTROL
000E LOW PCBC INSTRUCTION JP (BC)	BB39 KM SET REPEAT
0010 LOW SIDE CALL	BB3C KM GET REPEAT
0013 LOW KL SIDE PCHL	BB3F KM SET DELAY
0016 LOW PCDE INSTRUCTION JP (DE)	BB42 KM GET DELEAY
0018 LOW FAR CALL	BB45 KM ARM BREAK
001B LOW KL FAR PCHL	BB48 KM DISARM BREAK
001E LOW PCHL INSTRUCTION JP (HL)	BB4B KM BREAK EVENT
0020 LOW RAM LAM	BB4E TXT INITIALISE
0023 LOW KL FAR ICALL	BB51 TXT RESET
0028 LOW FIRM JUMP	BB54 TXT VDU ENABLE
0030 LOW USER RESTART	BB57 TXT VDU DISABLE
0038 LOW INTERRUPT ENTRY	BB5A TXT OUTPUT
003B LOW EXT INTERRUPT	BB5D TXT WR CHAR
349A FLO SUB	BB60 TXT RD CHAR
B900 HI KL U ROM ENABLE	BB63 TXT SET GRAPHIK
B903 HI KL U ROM DISABLE	BB66 TXT WIN ENABLE
B906 HI KL L ROM ENABLE	BB69 TXT GET WINDOW
B909 HI KL L ROM DISABLE	BB6C TXT CLEAR WINDOW
B90C HI KL ROM RESTORE	BB6F TXT SET COLUMN
B90F HI KL ROM SELECT	BB72 TXT SET ROW
B912 HI KL CURR SELECTION	BB75 TXT SET CURSOR
B915 HI KL PROBE ROM	BB78 TXT GET CURSOR
B918 HI KL ROM DESELECTION	BB7B TXT CUR ENABLE
B91B HI KL LDIR	BB7E TXT CUR DISABLE
B91E HI KL LDDR	BB81 TXT CUR ON
B921 HI KL POLL SYNCHRONOUS	BB84 TXT CUR OFF
B92A HI KL SCAN NEEDED	BB87 TXT TXT VALIDATE
BB00 KM INITIALIZE	BB8A TXT PLACE CURSOR
BB03 KM RESET	BB8D TXT REMOVE CURSOR
BB06 KM WAIT	BB90 TXT SET PEN
BB09 KM READ CHAR	BB93 TXT GET PEN
BB0C KM CHAR RETURN	BB96 TXT SET PAPER
BB0F KM SET EXPAND	BB99 TXT GET PAPER
BB12 KM GET EXPAND	BB9C TXT INVERSE
BB15 KM EXP BUFFER	BB9F TXT SET BACK
BB18 KM WAIT KEY	BBA2 TXT GET BACK
BB1B KM READ KEY	BBA5 TXT GET MATRIX
	BBA8 TXT SET MATRIX
	BBAB TXT SET M TABLE
	BBAE TXT GET TABLE
	BBB1 TXT GET CONTROLS
	BBB4 TXT STREAM SELECT

Adr. Bezeichnung	Adr. Bezeichnung
BB1E KM TEST KEY	BBB7 TXT SWAP STREAMS
BB21 KM GET STATE	BBBA GRA INITIALISE
BB24 KM GET JOYSTICK	BBBD GRA RESET
BB27 KM SET TRANSLATE	BBC0 GRA MOVE ABSOLUTE
BB2A KM GET TRANSLATE	BBC3 GRA MOVE RELATIVE
BB2D KM SET SHIFT	BBC6 GRA ASK CURSOR
BBC9 GRA SET ORIGIN	BC6E CAS START MOTOR
BBC C GRA GET ORIGIN	BC71 CAS STOP MOTOR
BBCF GRA WIN WIDTH	BC74 CAS RESTORE MOTOR
BBD2 GRA WIN HIGHT	BC77 CAS IN OPEN
BBD5 GRA GET W WIDTH	BC7A CAS IN CLOSE
BBD8 GRA GET W HIGHT	BC7D CAS IN ABANDON
BBDB GRA CLEAR WINDOW	BC80 CAS IN CHAR
BBDE GRA SET PEN	BC83 CAS IN DIRECT
BBE1 GRA GET PEN	BC86 CAS RETURN
BBE4 GRA SET PAPER	BC89 CAS TEST EOF
BBE7 GRA GET PAPER	BC8C CAS OUT OPEN
BBEA GRA PLOT ABSOLUTE	BC8F CAS OUT CLOSE
BBED GRA PLOT RELATIVE	BC92 CAS OUT ABANDON
BBF0 GRA TEST ABSOLUTE	BC95 CAS OUT CHAR
BBF3 GRA TEST RELATIVE	BC98 CAS OUT DIRECT
BBF6 GRA LINE ABSOLUTE	BC9B CAS CATALOG
BBF9 GRA LINE RELATIVE	BC9E CAS WRITE
BBFC GRA WR CHAR	BCA1 CAS READ
BBFF SCR INITIALISE	BCA4 CAS CHECK
BC02 SCR RESET	BCA7 SOUND RESET
BC05 SCR SET OFFSET	BCAA SOUND QUEUE
BC08 SCR SET BASE	BCAD SOUND CHECK
BC0B SCR GET LOCATION	BCB0 SOUND ARM EVENT
BC0E SCR SET MODE	BCB3 SOUND RELEASE
BC11 SCR GET MODE	BCB6 SOUND HOLD
BC14 SCR CLEAR	BCB9 SOUND CONTINUE
BC17 SCR CHAR LIMITS	BCBC SOUND AMPL ENVELOPE
BC1A SCR CHAR POSITION	BCBF SOUND TONE ENVELOPE
BC1D SCR DOT POSITION	BCC2 SOUND A ADDRESS
BC20 SCR NEXT BYTE	BCC5 SOUND T ADDRESS
BC23 SCR PREV BYTE	BCC8 KL CHOKE OFF
BC26 SCR NEXT LINE	BCCB KL ROM WALK
BC29 SCR PREV LINE	BCCE KL INIT BACK
BC2C SCR INK ENCODE	BCD1 KL LOG EXT
BC2F SCR INK DECODE	BCD4 KL FIND COMAND
BC32 SCR SET INK	BCD7 KL NEW FRAME FLY
BC35 SCR GET INK	BCDA KL ADD FRAME FLY
BC38 SCR SET BORDER	BCDD KL DEL FRAME FLY
BC3B SCR GET BORDER	BCE0 KL NEW FAST TICKER
BC3E SCR SET FLASHING	BCE3 KL ADD FAST TICKER
BC41 SCR GET FLASHING	BCE6 KL DEL FAST TICKER
BC44 SCR FILL BOX	BCE9 KL ADD TICKER
BC47 SCR FLOOD BOX	BCEC KL DEL TICKER
BC4A SCR CHAR INVERT	BCEF KL INIT EVENT

Adr. Bezeichnung	Adr. Bezeichnung
BC4D SCR HW ROLL	BCF2 KL EVENT
BC50 SCR SW ROLL	BCF5 KL SYNC RESET
BC53 SCR UNPACK	BCF8 KL DEL SYNCHRONOUS
BC56 SCR REPACK	BCFB KL NEXT SYNC
BC59 SCR ACCESS	BCFE KL DO SYNC
BC5C SCR PIXELS	BD01 KL DONE SYNC
BC5F SCR HORIZONTAL	BD04 KL EVENT DISABLE
BC62 SCR VERTICAL	BD07 KL EVENT ENABLE
BC65 CAS INITIALISE	BD0A KL DISARM EVENT
BC68 CAS SET SPEED	BD0D KL TIME PLEASE
BC6B CAS NOISY	BD10 KL TIME SET
BD13 MC BOOT PROGRAM	BDBB FLO RANDOMIZE 0
BD16 MC START PROGRAM	BDBE FLO RANDOMIZE
BD19 MC WAIT FLYBACK	BDCD IND TXT DRAW CURSOR
BD1C MC SET MODE	BDD0 IND TXT UNDRAW
BD1F MC SCREEN OFFSET	CURSOR
BD22 MC CLEAR INKS	BDD3 IND TXT WRITE CHAR
BD25 MC SET INKS	BDD6 IND TXT UNWRITE
BD28 MC RESET PRINTER	BDD9 IND TXT OUT ACTION
BD2B MC PRINT CHAR	BDDC IND GRA PLOT
BD2E MC BUSY PRINTER	BDDF IND GRA TEST
BD31 MC SEND PRINT	BDE2 IND GRA LINE
BD34 MC SOUND REGISTER	BDE5 IND SCR READ
BD37 JUMP RESTORE	BDE8 IND SCR WRITE
BD3A KM SET LOCKS	BDEB IND SCR MODE CLEAR
BD3D KM FLUSH	BDEE IND KM TEST BREAK
BD40 TXT ASK STATE	BDF1 IND MC WAIT PRINTER
BD43 GRA DEFAULT	BDF4 IND KM SCAN KEYS
BD46 GRA SET BACK	DD2A INT PREPARE VZ
BD49 GRA SET FIRST	DD37 KONV HLB TO INT
BD4C GRA SET LINE MASK	DD39 INT PREPARE
BD4F GRA FROM USER	DD4A INT ADD VZ
BD52 GRA FILL	DD52 INT SUB*VZ
BD55 SCR SET POSITION	DD53 INT SUB VZ
BD58 MC PRINT TRANSLATION	DD5B INT MULT VZ
BD5E EDIT	DD72 INT MULT
BD61 FLO MOVE	DD9C INT DIV VZ
BD64 KONV HLA TO FLO	DDA3 INT MOD VZ
BD67 KONV LW TO FLO	DDAB INT DIV
BD6A ROUND FLO TO HLA	DDED INT VZW
BD6D ROUND FLO TO LW	DDF9 INT SGN
BD70 FIX FLO TO LW	DE02 INT VGL
BD73 INT FLO TO LW	
BD76 FLO PREPARE	
BD79 FLO 10 A	
BD7C FLO ADD	
BD7F FLO RND	
BD82 FLO SUB*	
BD85 FLO MULT	
BD88 FLO DIV	

Adr. Bezeichnung	Adr. Bezeichnung
BD8B FLO LAST RND	
BD8E FLO VGL	
BD91 FLO VZW	
BD94 FLO SGN	
BD97 FLO DEG/RAD	
BD9A FLO PI	
BD9D FLO SQR	
BDA0 FLO POT	
BDA3 FLO LOG NAT	
BDA6 FLO LOG DEC	
BDAC FLO SIN	
BDAF FLO COS	
BDB2 FLO TAN	
BDB5 FLO ARC TAN	
BDB8 KONV LW+C TO FLO	

Anhang D BASIC-Token

(die Tokenwerte T sind hexadezimal angegeben)

T	Bedeutung	T	Bedeutung
00	Zeilenende	9B	ERASE
01	':', Ende eines Statements	9C	ERROR
02	es folgt Integervariable	9D	EVERY
03	es folgt Stringvariable	9E	FOR
04	es folgt Realvariable	9F	GOSUB
0D	es folgt Variable ohne Kennung	A0	GOTO
0E	Konstante 0	A1	IF
0F	Konstante 1	A2	INK
10	Konstante 2	A3	INPUT
11	Konstante 3	A4	KEY
12	Konstante 4	A5	LET
13	Konstante 5	A6	LINE
14	Konstante 6	A7	LIST
15	Konstante 7	A8	LOAD
16	Konstante 8	A9	LOCATE
17	Konstante 9	AA	MEMORY
19	Ein-Byte-Wert	AB	MERGE
1A	Zwei-Byte-Wert, dezimal	AC	MID\$
1B	Zwei-Byte-Wert, binär	AD	MODE
1C	Zwei-Byte-Wert, hexadez.	AE	MOVE
1D	Zeilenadresse	AF	MOVER
1E	Zeilennummer	B0	NEXT
1F	Fließkommawert	B1	NEW
80	AFTER	B2	ON
81	AUTO	B3	ON BREAK
82	BORDER	B4	ON ERROR GOTO 0
		B5	ON SQ

T	Bedeutung	T	Bedeutung
83	CALL	B6	OPENIN
84	CAT	B7	OPENOUT
85	CHAIN	B8	ORIGIN
86	CLEAR	B9	OUT
87	CLG	BA	PAPER
88	CLOSEIN	BB	PEN
89	CLOSEOUT	BC	PLOT
8A	CLS	BD	PLOTR
8B	CONT	BE	POKE
8C	DATA	BF	PRINT
8D	DEF	C0	'
8E	DEFINT	C1	RAD
8F	DEFREAL	C2	RANDOMIZE
90	DEFSTR	C3	READ
91	DEG	C4	RELEASE
92	DELETE	C5	REM
93	DIM	C6	RENUM
94	DRAW	C7	RESTORE
95	DRAWR	C8	RESUME
96	EDIT	C9	RETURN
97	ELSE	CA	RUN
98	END	CB	SAVE
99	ENT	CC	SOUND
9A	ENV	CD	SPEED
CE	STOP	E7	SWAP
CF	SYMBOL	E8	-
D0	TAG	E9	-
D1	TAGOFF	EA	TAB
D2	TRON	EB	THEN
D3	TROFF	EC	TO
D4	WAIT	ED	USING
D5	WEND	EE	>
D6	WHILE	EF	=
D7	WIDTH	F0	>=
D8	WINDOW	F1	
D9	ZONE	F2	>
DA	WRITE	F3	=
DB	DI	F4	+
DC	EI	F5	-
DD	FILL	F6	*
DE	GRAPHICS	F7	/
DF	MASK	F8	
E0	FRAME	F9	
E1	CURSOR	FA	AND
E2	-	FB	MOD
E3	ERL	FC	OR
E4	FN	FD	XOR
E5	SPC	FE	NOT
E6	STEP	FF	es folgt ein Funktions-Token

Funktions-Token (stehen nach einem FF):

T Bedeutung	T Bedeutung	T Bedeutung
00 ABS	18 SQR	79 RIGHT\$
01 ASC	19 STR\$	7A ROUND
02 ATN	1A TAN	7B STRING\$
03 CHR\$	1B UNT	7C TEST
04 CINT	1C UPPER\$	7D TESTR
05 COS	1D VAL	7E COPYCHR\$
06 CREAL	40 EOF	7F VPOS
07 EXP	41 ERR	
08 FIX	42 HIMEM	
09 FRE	43 INKEY\$	
0A INKEY	44 PI	
0B INP	45 RND	
0C INT	46 TIME	
0D JOY	47 XPOS	

T Bedeutung	T Bedeutung	T Bedeutung
0E LEN	48 YPOS	
0F LOG	49 DERR	
10 LOG10	71 BIN\$	
11 LOWER\$	72 DEC\$	
12 PEEK	73 HEX\$	
13 REMAIN	74 INSTR	
14 SGN	75 LEFT\$	
15 SIN	76 MAX	
16 SPACE\$	77 MIN	
17 SQ	78 POS	

Anhang E

UA 880-Befehle und Laufzeiten

Die Befehle der CPU UA 880 werden in Tabellenform dargestellt. In den Tabellenplätzen wird der Operationscode der Befehle in hexadezimaler Form angeführt. Damit können kurze Maschinenprogramme handassembliert und in BASIC Programmen z.B. in DATA-Zeilen eingebunden werden. Die Freiräume in den Tabellen bedeuten nichtvorhandene Befehle. In den Tabellen werden mit n Einbytevariablen, mit nn Zweibytevariablen (nl ist dabei der Low-Teil und nh der High-Teil), mit d ein Offset (-128 bis +127, negative Zahlen im Zweierkomplement) und mit e eine Sprungdistanz (-126 bis 129, Differenz von Ziel- und Startadresse minus 2, negative Zahlen ebenfalls im Zweierkomplement) angegeben.

Damit im KC compact die CPU und die Bildansteuerung auf den selben Speicher konfliktfrei zugreifen können, wird die CPU in

LD >, > A B C D E H L (HL) (BC) (DE) (IX+d) (IY+d) (nn) n I R

(DE)	12	
(IX+d)	DD DD DD DD DD DD DD DD 77 70 71 72 73 74 75 76 d d d d d d d d	DD 36 d n
(IY+d)	FD FD FD FD FD FD FD FD 77 70 71 72 73 74 75 76 d d d d d d d d	FD 36 d
(nn)	32	8 Bit Ladebefehle
	nl	Beispiel: LD A,(HL)
	nh	Opcode = 7EH
		Quelle = (HL)
I	ED	Ziel = A
	47	Alle Befehle außer LD A,I und LD A,R ändern die Flags nicht.
R	ED	Für diese beiden Befehle gilt: S Z H P/V CY
	4F	. * .

*Inhalt von Interrupt-Flipflop

Laufzeiten: LD r,r :1
 LD r,n , LD r,(rr) und LD (rr),r :2
 LD (HL), n ,LD A,i und LD i,A :3
 LD A,(nn) und LD (nn),A :4
 LD r,(ii+d) und LD (ii+d),R :6
 Quelle

POP >

LD >, >	AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
AF										F1
BC								01 nl nh	ED 4B nl nh	C1
DE								11 nl nh	ED 5B nl nh	D1
HL								21 nl nh	2A nl nh	E1
SP			F9		DD F9	FD F9		31 nl nh	ED 7B nl nh	
IX								DD 21 nl nh	DD 2A nl nh	DD E1

LD >, >	AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)

IY								FD	FD	FD
								21	2A	E1
								nl	nl	
								nh	nh	

(nn)		ED	ED	22	ED	DD	FD			
		43	53	nl	73	22	22			
		nl	nl	nh	nl	nl	nl			
		nh	nh		nh	nh	nh			

(SP)	F5	C5	D5	E5		DD	FD			
PUSH >						E5	E5			

16 Bit Ladebefehle

Das Flagregister wird nicht beeinflusst.

Laufzeiten: LD SP,HL :2

LD rr,nn, LD SP,ii, POP rr :3

LD ii,nn, PUSH rr, POP ii :4

LD HL,(nn), LD (nn),HL, PUSH ii :5

LD rr,(nn), LD (nn),rr, LD ii,(nn), LD (nn),ii :6

Befehl	Opcode	Wirkung	S	Z	H	P/V	N	CY
LDI	ED (DE) -> (HL)		.	.	0	*	0	.
	A0 DE -> DE+1, HL -> HL+1, BC -> BC-1							

LDIR	ED (DE) -> (HL)		.	.	0	0	0	.
	B0 DE -> DE+1, HL -> HL+1, BC -> BC-1 bis BC 0 wird							

LDD	ED (DE) -> (HL)		.	.	0	*	0	.
	A8 DE -> DE-1, HL -> HL-1, BC -> BC-1							

LDDR	ED (DE)-> (HL)		.	.	0	0	0	.
	B8 DE -DE-1,HL -HL-1,BC -BC-1 bis BC 0 wird							

CPI	ED Vergleich, ob A=(HL)?			#		*	1	.
	A1 HL -> HL+1, BC -> BC-1							

CPIR	ED Vergleich, ob A=(HL)?			#		*	1	.
	B1 HL -> HL+1, BC -> BC-1 bis BC 0 wird oder A=(HL)							

CPD	ED Vergleich, ob A=(HL)?			#		*	1	.
	A9 HL -> HL-1, BC -> BC-1							

CPDR	ED Vergleich, ob A=(HL)?			#		*		.
	B9 HL -> HL-1, BC -> BC-1 bis BC 0 wird oder A=(HL)							

Blocktransport- und -suchbefehle

Flags: * P/V Flag=0, wenn BC im Ausgang des Befehls zu 0 wird
 ansonsten 1

Z Flag=|, wenn a=(HL), ansonsten 0

Laufzeiten: LDI, LDD, CPI, CPD :5

LDIR, LDDR, CPIR, CPDR :5, wenn BC zu 0 wird,
 7 ansonsten

Befehl	Opcode	Wirkung	S	Z	H	P/V	N	CY
EX AF	08	AF -> AF'						
EXX	D9	BC -> BC', DE ->DE', HL ->HL'						
EX DE,HL	EB	DE -> HL						
EX (SP),HL	E3							
EX (SP),IX	DD E3							
EX (SP),IY	FD E3							

Austauschbefehle

Flags werden nicht beeinflusst

Laufzeiten: EX AF, EXX, EX DE, HL :1

EX (SP),HL :6

EX (SP),ii :7

	A	B	C	D	E	H	L (HL)	(IX+d)	(IY+d)	n	S	Z	H	P/V	N	CY	
ADD	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n				V	0	
ADC	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n				V	0	
SUB	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n				V	1	
SBC	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n				V	1	
AND	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n			1	P	0	0
XOR	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n			0	P	0	0

	A	B	C	D	E	H	L (HL)	(IX+d)	(IY+d)	n	S	Z	H	P/V	N	CY	
OR	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n			0	P	0	0
CMP	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n				V	1	
INC	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d					V	0	.
DEC	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d					V	1	.

8 Bit Arithmetikbefehle

Laufzeiten: ADD r,ADC r,SUB r,SBC r,AND r,OR r,XOR r,CMP r :1
 INC r,DEC r :1
 ADD n,ADC n,SUB n,SBC n,AND n,OR n,XOR n,CMP n :2
 ADD (HL),ADC (HL),SUB (HL),SBC (HL),AND (HL) :2
 OR (HL),XOR (HL),CMP (HL) :2
 INC (HL),DEC (HL):3
 ADD (ii+d),ADC (ii+d),SUB (ii+d),SBC (ii+d) :6
 AND (ii+d),OR (ii+d),XOR (ii+d),CMP (ii+d) :6
 INC (ii+d),DEC (ii+d):7

	A	B	C	D	E	H	L (HL)	(IX+d)	(IY+d)		S	Z	H	P/V	N	CY	
RLC	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d	FD CB d				0	P	0	
RLCA	07								06	06	CY	-	-	7..	-	..0	-
RRC	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d	FD CB d				0	P	0	
RRCA	0F								0E	0E	->	7..	->	..0	->	->	CY
RL	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d	FD CB d				0	P	0	
RLA	17								16	16	-CY	-	7..	-	..0	-	
RR	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d	FD CB d				0	P	0	
RRA	1F								1E	1E	->	7..	->	..0	->	CY	->
SLA	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d	FD CB d				0	P	0	
									26	26	CY	-	7..	-	..0	-	0

	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	S	Z	H	P/V	N	CY		
SRA	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E	--.			0	P	0		
												-7...->...0->CY						
SLL	CB 37	CB 30	CB 31	CB 32	CB 33	CB 34	CB 35	CB 36	DD CB d 36	FD CB d 36				0	P	0		
												CY -7...-..0 -1						
SRL	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E				0	P	0		
												0->7...->...0->CY						
RLD									ED 6F					0	P	0	.	
Bits sind in Register A und in (HL)									7..4 3..0		-		7..4		-		3..0	
RRD									ED 67					0	P	0	.	
Bits sind in Register A und in (HL)									7..4 3..0		->		7..4		->		3..0	

Verschiebefehle

Flags: bei RLCA, RRCA, RLA und RRA CY=|, N, H=0 und S,Z,P/V=.

Laufzeiten: RLCA,RRCA,RLA,RRA :1

RLC r,RRC r,RL r,RR r,SLA r,SRA r,SLL r,SRL r :2

									46	46	
BIT 1, >	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E	
BIT 2, >	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56	
BIT 3, >	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E	
BIT 4, >	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66	
BIT 5, >	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E	

BIT 6, >	CB CB CB CB CB CB CB	CB DD	FD
	77 70 71 72 73 74 75	76 CB	CB
		d d	
		76 76	

BIT 7, >	CB CB CB CB CB CB CB	CB DD	FD
	7F 78 79 7A 7B 7C 7D	7E CB	CB
		d d	
		7E 7E	

Bittestbefehle:

Flags: P=X, S=X, N=0, H=| und Z=Komplement des getesteten Bits.
 Laufzeiten: BIT b,r :2
 BIT b,(HL) :3
 BIT b,(ii) :6

Bitrücksetzbefehle: wie Bittestbefehle, letztes Byte des Operationscodes+40H
 Bitsetzbefehle: wie Bittestbefehle,letztes Byte des Operationscodes+80H

Flags: unbeeinflusst
 Laufzeiten: RES b,r,SET b,r :3
 RES b,(HL),SET b,(HL) :4
 RES b,(ii),SET b,(ii) :7

	BC DE HL SP IX IY S Z H P/V N CY		
ADD HL,	09 19 29 39	. . X . 0	IN A,n DB n
ADD IX,	DD DD DD DD 09 19 39 29	. . X . 0	IN A,(C) ED 78
ADD IY,	FD FD FD FD 09 19 39 29	. . X . 0	IN B,(C) ED 40
ADC HL,	ED ED ED ED 4A 5A 6A 7A	X V 0	IN C,(C) ED 48
SBC HL,	ED ED ED ED 42 52 62 72	X 1 0	IN D,(C) ED 50
INC / /	03 13 23 33 DD FD 23 23	IN E,(C) ED 58
DEC / /	0B 1B 2B 3B DD FD 2B 2B	IN H,(C) ED 60
			IN L,(C) ED

16 Bit Arithmetikbefehle

Laufzeiten: INC rr,DEC rr :2 68

ADD HL,rr :3

ADC HL,rr,SBC HL,rr :4

ADD ii,rr,ADD ii,ii :4

INF ED

Flags,(C) 70

OUT n,A D3

Befehl	Opcode	Wirkung	S	Z	H	P/V	N	CY		n
INI	ED A2	(HL) =(C) B -B-1,HL	X		X	X	1	.	OUT (C),A	ED 79
INIR	ED B2	(HL)=(C) B -B-1,HL	X	1	X	X	1	.	OUT (C),B	ED 41
IND	ED AA	(HL)=(C) B -B-1,HL	X		X	X	1	.	OUT (C),C	ED 49
INDR	ED BA	(HL)=(C) B -B-1,HL	X	1	X	X	1	.	OUT (C),D	ED 51
OUTI	ED A3	(C)=(HL) B -B-1,HL	X		X	X	1	.	OUT (C),E	ED 59
OUTIR	ED B3	(C)=(HL) B -B-1,HL	X	1	X	X	1	.	OUT (C),H	ED 61
OUTD	ED AB	(C)=(HL) B -B-1,HL	X		X	X	1	.	OUT (C),L	ED 69
OUTDR	ED BB	(C)=(HL) B -B-1,HL	X	1	X	X	1	.	OUTF (C),Flags	ED 71

Block-I/O-Befehle I/O Befehle

Flags: Z=1, wenn B null wird, sonst 0

Laufzeiten: 5, bei repet. :6 wenn B=0

Bef Wirkung unbed.

NZ Z NC C PO

Flags: konstant

Laufzeit :3

PE P M

RET > Rückk.	C9	C0	C8	D0	D8	E0	E8	F0	F8
JP >,nn Sprung	C3 n1 nh	C2 n1 nh	CA n1 nh	D2 n1 nh	DA n1 nh	E2 n1 nh	EA n1 nh	F2 n1 nh	FA n1 nh
CALL >,nn CD UP Ruf	C4 n1 nh	CC n1 nh	D4 n1 nh	DC n1 nh	E4 n1 nh	EC n1 nh	F4 n1 nh	FC n1 nh	
JR >,e+2 rel Sprung	18 e	20 e	28 e	30 e	38 e	DJNZ ,B -B-1		10	rel. Sprung wenn B=0

Sprung-, Unterprogrammruf- und -Rückkehrbefehle

Flags: nicht beeinflusst, DJNZ setzt Z in Abh. von B

Laufzeiten: RET cc und JR cc, wenn cc nicht erfüllt :2

RET,JP und JP cc wenn cc erfüllt :3

CALL cc,nn wenn cc nicht erfüllt, DJNZ (B -0) :3

JR, RET cc und JR cc wenn cc erfüllt :4

CALL nn, CALL cc,nn wenn cc erfüllt, DJNZ (B >0):5

Befehl	Opcode	Wirkung	S	Z	H	P/V	N	CY	Befehl	Opcode
DAA	27	Dezimalkorr.				P	.		RST 0	C7
CPL	2F	A -ßA)	.	.	1	.	1	.	RST 8	CF
SCF	37	CY -1	.	.	1	.	1	1	RST 10	D7
CCF	3F	CY -ßCY	.	.	X	.	1		RST 18	DF
NEG	ED 44	A -ßA+1			V	1			RST 20	E7
									RST 28	EF
									RST 30	F7
									RST 38	FF

Akkumulator Zusatzbefehle und Flagoperationen

Laufzeiten: DAA, CPL, SCF, CCF :1, NEG :2

Restartbefehle, Laufzeiten: 4 ==>

Befehl	Wirkung	Opcode	Zeit	Befehl	Oc	Zeit	Befehl	Oc	Zeit
JP (HL)	Sprung zum	E9	1	NOP	00	1	IM0	ED 46	2
JP (IX)	Inhalt eines Speicher-	DD E9	2	HALT	76	1	IM1	ED 56	2
JP (IY)	platzes	FD E9	2	DI	F3	1	IM2	ED 5E	2
RETI	Rückk. von Int.	ED 4D	4	Sonderbefehle, Flags: unbeeinflusst					
RETN	Rückk. von MNI	ED 45	4	Laufzeiten bei Interruptquittie- rung: NMI, IM1 :4 IM0 :abh.vom Befehl 4-6 IM2 :6					
Flags: unbeeinflusst									

Sachwortverzeichnis