KLEINCOMPUTER

KC compact

SYSTEMHANDBUCH Teil 1 u. 2

KLEINCOMPUTER

KC compact

Systemhandbuch Teil1

veb mikroelektronik >wilhelm pieck<
mühlhausen
im veb kombinat mikroelektronik

veb mikroelektronik "wilhelm pieck" mühlhausen

Hinweise, die zur Verbesserung dieser Dokumentation führen, werden gern entgegengenommen.
Redaktionsschluß: Dezember 1989

G l i e d e r u n g

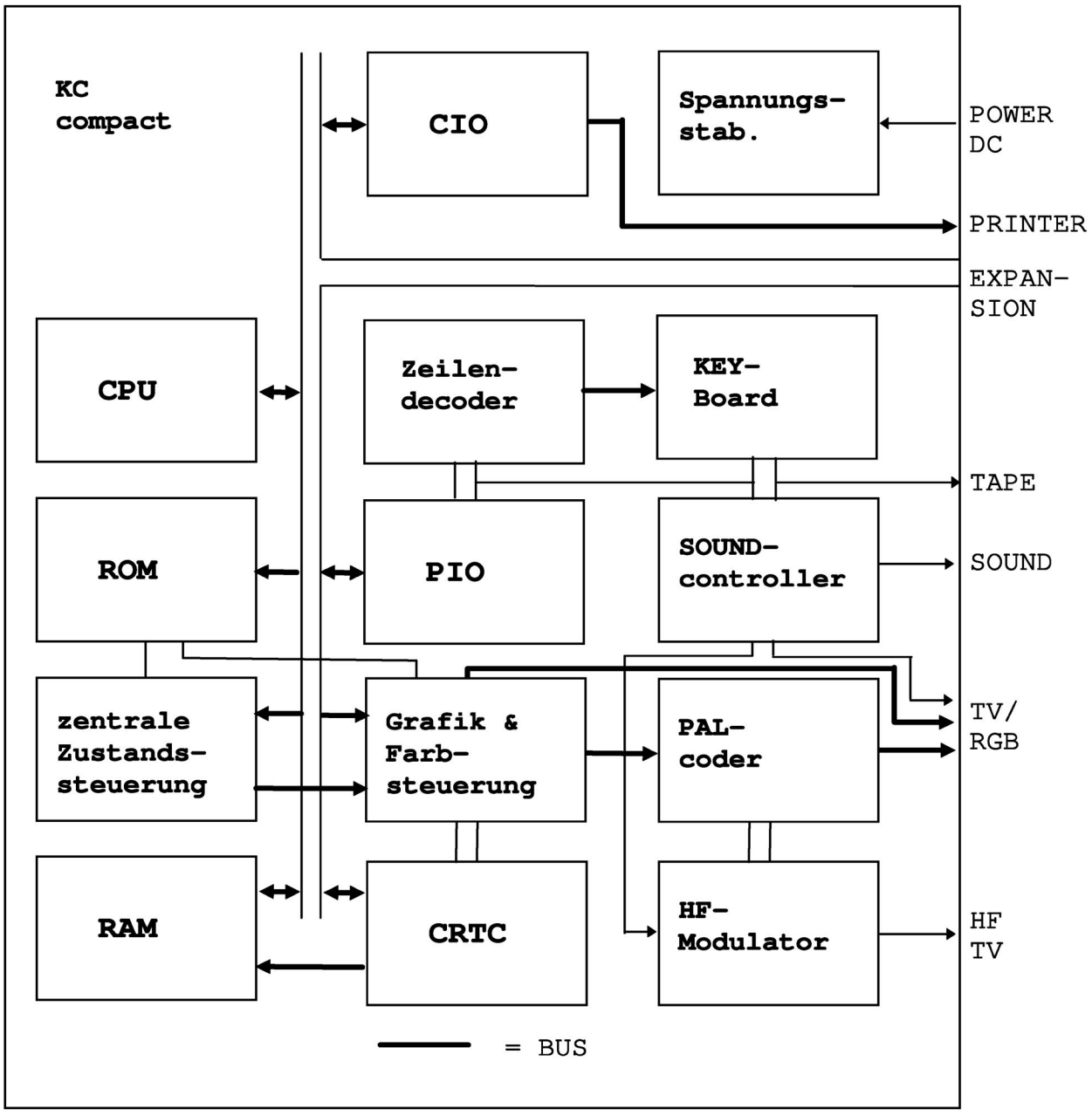
1. E I N L E I T U N G

Das vorliegende Systemhandbuch gibt einen kurzen Überblick über die Ressourcen der Hardware und des Betriebssystems, die durch den KC compact bereitgestellt werden. Die Hardware wird im Überblick vorgestellt. Alle I/O-Bausteine, ihre Adressierung, die für den Anwender nutzbaren Register und die Wirkung dieser Register werden beschrieben. Das Betriebssystem stellt mit seiner Interruptbehandlung Mittel zur Programmierung zeitabhängiger Vorgänge auch auf Hochsprachniveau bereit. Alle wichtigen Routinen des Betriebssystems und des BASIC sind durch eine Sprungleiste im zentralen RAM erreichbar. Die Wirkung dieser Routinen wird beschrieben und ihre Schnittstellen werden genau definiert. Die Einbindung eigener Routinen durch das Patchen der Betriebssystemroutinen (Änderung von Adreßzeigern) wird an einem Beispiel demonstriert. Die RAM-Zellen des Betriebssystems und des BASIC werden komplett aufgelistet. Die Speicherverwaltung wird vorgestellt, und es wird gezeigt,wie man die bestehenden BASIC-Befehle durch eigene Definitionen erweitern kann.
Damit richtet sich dieses Systemhandbuch vorwiegend an den fortgeschrittenen Computernutzer. Kenntnisse in Assemblerprogrammierung und in BASIC werden beim Leser vorausgesetzt.

2. H A R D W A R E

2.1. Das Blockschaltbild des KC compact

Im folgenden werden die wichtigsten Baugruppen vorgestellt und erläutert.



Legende:
CIO - Counter Input Output
CPU - Central Processing Unit
CRTC - Cathode Ray Tube Controler (Bildschirmsteuerung)
HF - High Frequency
PIO - Parallel Input Output
RAM - Random Access Memory (Schreib-/Lesespeicher)
ROM - Read Only Memory (Nur-Lese-Speicher)

Abb.2.1 Blockschaltbild des KC compact

Die CPU arbeitet auf einen Systembus, der auch als Expansionsinterface an der Geräterückseite herausgeführt wird. Ihre Signale werden nicht getrieben.
Die zentrale Zustandssteuerung ist für das Ein- bzw. Ausblenden der RAM-und ROM-Bereiche, für den Grafikmodus und für die Interruptauslösung zuständig.
Im KC compact sind mehrere I/O-Bausteine enthalten. Die CIO ist ein moderner, hochintegrierter Schaltkreis, der drei parallele Ports und drei Zähler/Zeitgeber enthält. Sie wird weitgehend zur Realisierung von Hardware-und Systemfunktionen benutzt. Bis auf den Kanal A mit seinen vielfältigen Installierungsmöglichkeiten sind alle anderen Ressourcen dieses Schaltkreises für Systemfunktionen belegt, also für den Anwender tabu!Der Kanal A wird standardmäßig für das Druckerinterface (PRINTER) benutzt.
Ein weiterer I/O-Baustein ist eine PIO mit drei Ports. Sie wird benutzt zur Kasettenein- und -ausgabe, wählt über einen Dekoder die Zeilen der Tastatur an und bedient einen Soundcontroler.
Der Soundcontroler wird über die PIO programmiert und ausgelesen. Er wird im Gegensatz zu allen anderen Schaltkreisen mit einer Taktfrequenz von 1MHz betrieben. Er kann aus drei periodischen und einer rauschähnlichen Signalquelle drei Tonausgänge versorgen, die zu Stereo-und Monosignalen für die Ausgänge SOUND und TV/RGB sowie für den HF-Modulator gemischt werden. Über den Soundcontroler werden die Spalteninformationen der Tastatur eingelesen.
Die Ansteuerung des Bildschirms wird durch mehrere Baugruppen realisiert. Der CRTC generiert die Synchronimpulse für das angeschlossene Fernsehgerät bzw. den Monitor und die Adressen für den RAM, um die aktuell benötigte Bildinformation auslesen zu können. In der zentralen Zustandssteuerung werden die Bildinformationen entsprechend dem eingestellten Grafikmodus in Adressen für den Farbwertspeicher (Tintennummern) umgewandelt. Dort sind die Palettenfarbnummern abgelegt. In der Grafik- und Farbsteuerung werden die Palettenfarbnummern in die Farbinformationen rot, grün und blau umgewandelt, mit denen ein Farbmonitor angesteuert werden kann. In den Baugruppen PAL-coder und HF-Modulator werden die Signale für Schwarz/Weiß- und Farbfernsehgeräte aufbereitet. Die Baugruppe Spannungsstabilisierung erzeugt aus ca. 20 V Gleichspannung (Rohspannung) die im KC compact benötigten stabilisierten Spannungen von 5 V und 12 V.

2.1.1.Die Speicherkonfiguration

Der KC compact ist mit 32 KByte ROM (Festwertspeicher für Betriebssystem und BASIC)und 64 KByte RAM (Schreib/Lesespeicher) ausgestattet. Da die zentrale Recheneinheit nur einen 64 KByte großen Speicherbereich adressieren kann,werden einzelne Adreßbereiche mehrfach genutzt. Schreibzugriffe erfolgen dabei immer auf den RAM. Ob Lesezugriffe auf den ROM oder den RAM gelenkt werden, hängt davon ab, was letztmalig auf das Multifunktionsregister der zentralen Zustandssteuerung ausgegeben wurde (siehe auch Abschn. 2.1.4.). Im folgenden ist die Speicherkonfiguration dargestellt.

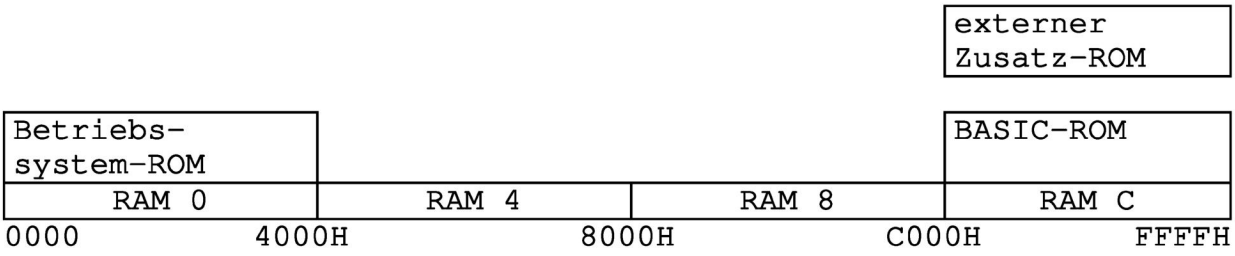


Abb. 2.2 Speicherkonfiguration im KC compact

2.1.2. Die CPU UA 880

Die CPU UA 880 wird mit 4 MHz Taktfrequenz betrieben. Die Taktperioden werden als T-Zustand bezeichnet. Jeder Maschinenzyklus (Grundfunktion der CPU) besteht aus drei bis sechs T-Zuständen. In der fallenden Flanke des zweiten Taktes eines jeden Maschinenzyklus wird der Zustand des WAIT-Eingangs abgetestet. Liegt der WAIT-Eingang auf Lowpotential, dann wird von der CPU im nächsten Takt ein WAIT-Zyklus eingeschoben. Im KC compact wird der WAIT-Eingang für drei Takte auf Lowpotential und für einen Takt auf Highpotential gehalten. Die CPU synchronisiert sich in ihrem Betrieb so, daß nach dem Highzustand des WAIT-Einganges ein T3-Zyklus folgt. Die Maschinenzyklen werden durch das automatische Einfügen von WAIT-Zyklen auf vier oder acht Takte Länge gebracht.Für die Berechnung der Laufzeit von Programmen auf dem KC compact wird deshalb der Begriff eines Pseudomaschinenzyklus eingeführt. Er ist immer vier Takte oder eine Mikrosekunde lang.

Im Anhang E sind alle Befehle der CPU UA 880 und deren Laufzeiten in Tabellenform dargestellt.

2.1.3. Die I/O-Adressen im KC compact

Bei der Ausführung des Befehls OUT (C),r oder IN r,(C) durch die CPU UA 880 wird der Inhalt des Doppelregisters BC auf den Adreßbus gelegt. Zur Selektion der I/O-Baugruppen wird beim KC compact der Inhalt der oberen acht Bit des Adreßbusses mit benutzt. Aus diesem Grund sind die repetierenden I/O Befehle, bei denen der Inhalt des B-Registers verändert wird, nicht anwendbar. Die Adressierung aller im I/O-Bereich liegenden Register durch die CPU UA 880 wird in Tab.2.1 gezeigt.

Tab.2.1 Die I/O Adressen des KC compact

ADRESSBUS B	ADRESSBUS C	DATENBUS Datenreg	HEX B	HEX C	HEX Reg.	FUNKTION	R/W
011111XX	XXXXXXXX	11XXXXXX	7F	XX	XX	reserviert!	W
011111XX	XXXXXXXX	10XXXXXX	7F	XX	XX	MF-Register	W
011111XX	XXXXXXXX	01XXXXXX	7F	XX	XX	FW-Register	W
011111XX	XXXXXXXX	00XXXXXX	7F	XX	XX	FN-Register	W
10111111	XXXXXXXX	XXXXXXXX	BF	XX	XX	CRTC Lesen Controlreg.	R
10111101	XXXXXXXX	XXXXXXXX	BD	XX	XX	CRTC Schr.Controlreg.	W
10111100	XXXXXXXX	Reg.adr.	BC	XX	reg	CRTC Schr.Reg.select	W
110111XX	XXXXXXXX	ROM-Nr.	DF	XX	Nr	ROM select	W
11101111	XXXXXXXX	XXXXXXXX	EF	XX	XX	Centronics (CIO A)	(R)W
11101110	XXXXXXXX	XXXXXXXX	EE	XX	XX	CIO Steueradr.	R/W
11101101	XXXXXXXX	XXXXXXXX	ED	XX	XX	CIO Port C	R/W
11101100	XXXXXXXX	XXXXXXXX	EC	XX	XX	CIO Port B	R/W
11110111	XXXXXXXX	XXXXXXXX	F7	XX	XX	PIO Controlreg.	R/W
11110110	XXXXXXXX	XXXXXXXX	F6	XX	XX	PIO Port C	R/W
11110101	XXXXXXXX	XXXXXXXX	F5	XX	XX	PIO Port B	R/W
11110100	XXXXXXXX	XXXXXXXX	F4	XX	XX	PIO Port A	R/W
111110XX	111110XX	XXXXXXXX	FX	FX	XX	frei für Anwender	R/W

2.1.4. Die zentrale Zustandssteuerung

Die zentrale Zustandssteuerung wird durch drei Register auf der I/O Adresse 7FXXH realisiert. Die Auswahl der Register wird durch die Bits 6 und 7 des Datenwortes bestimmt.

Tab. 2.2 Die Auswahl der Register der zentralen Zustandssteuerung durch das Datenbyte

Bit 7	Bit 6	Register
1	1	reserviert, nicht benutzen
1	0	Multifunktionsregister (MF-Register)
0	1	Farbwertregister (FW-Register)
0	0	Farbnummernregister (FN-Register)

Multifunktionsregister

Bit	7	6	5	4	3,2	1,0
Inhalt	0	0	X	ICR	ROMSEL	MODE

Bit 4: ICR (Interruptcounterreset)

Alle 52 Bildschirmzeilen wird ein Interrupt ausgelöst. Der Zähler dazu wird durch die fallende Flanke des Horizontalsynchronimpulses aus dem CRTC getaktet. Die Lage der Interrupts im Ablauf eines Bildes wird so eingestellt, daß zur fallenden Flanke des zweiten Horizontalsynchronimpulses im Vertikalsynchronimpuls

ein Interrupt erscheint. Durch die Ausgabe einer 1 auf des ICR wird der Zeilenzähler zurückgesetzt und beginnt erneut, 52 Zeilen abzuzählen. Folgendes BASIC-Programm soll das demonstrieren:

```
10 OUT &7FFF,&X10010101:GOTO 10
```

Nach dem Start des Programms mit RUN ist der Rechner blockiert. Es treten keine Interrupts mehr auf und die Tastatur wird nicht abgefragt. Wird erst innerhalb der zweiten 26 Zeilen nach einer Interruptanmeldung eine Interruptquittierung durch die CPU gesendet, dann wird das höchstwertige Bit des Zeilenzählers automatisch rückgesetzt. Das entspricht einer Verminderung der bereits gezählten Zeilen um 26, bzw. einer Verschiebung der nächsten Interruptanmeldung um 26 Zeilen. Damit wird verhindert, daß ein neuer Interrupt unmittelbar nach einer Interruptquittierung auftreten kann.

Bit 3: 0=selektierten ROM im Adreßbereich von 0C000H bis 0FFFFH einschalten.
1=ROM ausschalten.

Bei eingeschaltetem ROM erfolgen Lesezugriffe der CPU auf den entsprechenden Adressen im ROM (Festwertspeicher mit Betriebssystem, BASIC u.a.), ansonsten zum RAM (Schreib-Lesespeicher). Schreibzugriffe erfolgen grundsätzlich zum RAM.

Bit 2: 0=Betriebssystem-ROM im Adreßbereich von 0000H bis 03FFFH einschalten.
1=ROM ausschalten.

Bit 1,Bit 0: Auswahl des Bildschirmmodus entsprechend Tabelle 2.3

Tab. 2.3 Die Auswahl des Bildschirmmodus durch Bit 0 und 1 des Multifunktionsregisters

Bit 1	Bit 0	Wirkung
1	1	reserviert,nicht benutzen
1	0	MODE 2, 80 Zeichen/Zeile, 2 Farben
0	1	MODE 1, 40 Zeichen/Zeile, 4 Farben
0	0	MODE 0, 20 Zeichen/Zeile, 16 Farben

Im Mode 0 werden in einem Byte zwei Pixel (0 und 1) beschrieben. Für jedes Pixel stehen 4 Bit für eine Tintennummer zur Verfügung, d.h., es können 16 verschiedene Farben gleichzeitig auf dem Bildschim gezeigt werden. Im Mode 1 stehen nur noch 2 Bit und im Mode 2 nur 1 Bit für die Tintennummer zur Verfügung. Die Zuordnung der einzelnen Bits in einem Byte zu einem Pixel und die Rolle der einzelnen Bits in der Tintennummer wird für die drei Bildschirmmodi in Kapitel 3.4.2. dargestellt.

Farbwertregister

Bit	7	6	5	4,3,2,1,0
Inhalt	0	1	X	Palettenfarbwert

Über die eine I/O-Adresse für das Farbwertregister sind insgesamt 17 Register erreichbar, in denen ein 5 Bit breiter Farbwert abgelegt werden kann (Bit 0 bis Bit 4, Bit 5 hat keine Bedeutung). Welches dieser 17 Register in Benutzung ist, wird durch die Ausgabe einer Farbnummer im Farbnummernregister bestimmt und wird nachfolgend beschrieben. Die Zuordnung der Palettenfarbwerte zu den einzelnen Farben wird in der folgenden Tabelle gezeigt. Zusätzlich wurden die Farbwerte des BASIC mit in die Tabelle aufgenommen. Sie entsprechen nicht den Werten, die in das FW-Register geschrieben werden!

Tabelle 2.4 Die Zuordnung der Farben zu den Farbwerten

BASIC Farbwert	Palettenfarbwert dez.	hex.	grün	rot	blau	Farbe
0	84	54H	0	0	0	schwarz
1	68	44H	0	0	1/2	blau
2	85	55H	0	0	1	leuchtend blau
3	92	5CH	0	1/2	0	rot
4	88	58H	0	1/2	1/2	magenta
5	93	5DH	0	1/2	1	mauve
6	76	4CH	0	1	0	leuchtend rot
7	69	45H	0	1	1/2	purpur
8	77	4DH	0	1	1	leuchtend magenta
9	86	56H	1/2	0	0	grün
10	70	46H	1/2	0	1/2	blaugrün
11	87	57H	1/2	0	1	himmelblau
12	94	5EH	1/2	1/2	0	gelb
13	64	40H	1/2	1/2	1/2	weiß
14	95	5FH	1/2	1/2	1	pastellblau
15	78	4EH	1/2	1	0	orange
16	71	47H	1/2	1	1/2	rosa
17	79	4FH	1/2	1	1	pastellmagenta
18	82	52H	1	0	0	leuchtend grün
19	66	42H	1	0	1/2	seegrün
20	83	53H	1	0	1	leuchtend blaugrün
21	90	5AH	1	1/2	0	limonengrün
22	89	59H	1	1/2	1/2	pastellgrün
23	91	5BH	1	1/2	1	pastellblaugrün
24	74	4AH	1	1	0	leuchtend gelb
25	67	43H	1	1	1/2	pastellgelb
26	75	4DH	1	1	1	leuchtend weiß

Farbnummernregister

Bit	7	6	5	4	3,2,1,0
Inhalt	1	0	X	BOSEL	Tintennummer

Im Bildwiederholspeicher des KC compact sind für die einzelnen Pixel die Tintennummern gespeichert.
In der Grafik- und Farbsteuerung erfolgt die Umwandlung der Tintennummern in die Farbwerte bzw. die Farben. Die Farben werden

folgendermaßen den Tintennummern zugeordnet: Durch die Ausgabe einer Tintennummer auf das Farbnummernregister wird eines der 17 Farbwertregister ausgewählt, und kann durch die anschließende Ausgabe eines Palettenfarbwertes beschrieben werden. Ist Bit 4 der Tintennummer gesetzt, dann werden Bit 0 bis 3 der Farbnummer ignoriert. Der folgende Palettenfarbwert wird dann dem Border zugeordnet. Ist Bit 4 der Farbnummer rückgesetzt, dann bilden Bit 0 bis 3 eine Adresse des Farbwertspeichers und wählen eines von 16 Farbwertregistern an. Bit 5 hat keine Bedeutung. Der folgende Palettenfarbwert wird in dieses adressierte Register eingetragen.

2.1.5. Der Videocontroler (CRTC)CM 607 /4/

Der CRTC ist ein sehr leistungsfähiger Baustein. Er stellt die Adressen für den Bildwiederholspeicher bereit und erzeugt die Synchronimpulse für die Ansteuerung eines Monitors bzw. eines Fernsehgerätes. Die Anzahl der Zeichen/Zeile, der Zeilen/Bild, die vertikale Ausdehnung der Punktmatrix für ein Zeichen und der Cursor (Blinken und HOME)sind programmierbar. Ein Speicherbereich von 16 KByte kann durch den CRTC adressiert werden, wobei dynamische RAMs während einer Bildausgabe automatisch aufgefrischt werden.
Die Ausgangssignale des CRTC werden in ihrer zeitlichen Ausdehnung aus einem Zeichentakt (caracter clock) mit der Frequenz von 1 MHz gebildet. In einer Taktperiode werden aber beim KC compact zwei Zeichen und nicht wie in Standardanwendungen des CRTC ein Zeichen dargestellt. Das muß beim Programmieren des CRTC beachtet werden. Bei der Erklärung der Bedeutung der einzelnen Register wird deshalb immer von Zeichentaktperioden (caracter clock) und nicht von der Anzahl der Zeichen gesprochen.
Der CRTC enthält ein Selectregister und 18 Controlregister. Durch die Ausgabe der entsprechenden Registernummer an das Selectregister (I/O-Adresse 0BCXXH) wird das gewünschte Controlregister ausgewählt und kann über die I/O-Adresse 0BDXXH beschrieben und über die I/O-Adresse 0BFXXH ausgelesen werden. Das Selectregister ist nur beschreibbar. Es folgt eine Zusammenstellung der Funktion der 18 Controlregister.

Tab. 2.5 Die Controlregister des CRTC

Reg Nr	R/W	Name	Anz Bit	Funktion
0	-/W	horizon-tal total	8	Anzahl Taktperioden/totale Zeile -1, d.h. Bild, Bildrand und Strahlrücklauf
1	-/W	horizon-tal displ	8	Anzahl Taktperioden/sichtbare Bildzeile, d.h. ohne Bildrand und Strahlrücklauf
2	-/w	horizon-tal sync	8	bestimmt die Lage des Horizontalsynchronimpulses innerhalb einer Zeile position in Taktperioden ab Zeilenanfang
3	-/W	sync widt	4	Länge des Horizontalsynchronimpulses

Reg Nr	R/W	Name	Anz Bit	Funktion
4	-/W	vertical total	7	Anzahl aller Zeichenzeilen/Bild -1, auch in Border und Strahlrücklauf
5	-/W	vertical total adj	5	Anzahl der Pixelzeilen,die zusätzlich zu den Zeichenzeilen generiert werden müssen,um eine normgerechte Bildwechselfrequenz zu erhalten
6	-/W	vertical displayed	7	Anzahl Zeichenzeilen/sichtbares Bild, d.h.ohne Bildrand und Strahlrücklauf
7	-/W	vertical sync pos	7	Lage des Vertikalsynchronimpulses vom Bildanfang an in Zeichenzeilen
8	-/W	interlace	2	Darstellung mit oder ohne Zeilensprung
9	-/W	max raster adress	5	Anzahl der Pixelzeilen/Zeichenzeile, d.h.pro darzustellendem Zeichen
10	-/W	cursor start raster	7	Bit 0 bis 4 bestimmen,auf welcher Pixelzeile des darzustellenden Zeichens der Cursor beginnt. Bit 6,5 Funktion --- 1 1 Cursor blinkt schnell 1 0 Cursor blinkt langsam 0 1 Cursor unsichtbar 0 0 Cursor nicht blinkend
11	-/W	cursor end adress	5	legt fest,auf welcher Pixelzeile in der Zeichenzeile der Cursor endet
12	R/W	start adress high	6	Highteil der Anfangsadresse des Bildwiederholspeichers im 16 KByte-Adreß-bereich des CRTC. Mitten im Bildwiederholspeicher kann ein Sprung vom Ende des 16 KByte-Bereiches auf den Anfang erfolgen.
13	R/W	start adress low	8	Lowteil der Anfangsadresse des Bildwiederholspeichers im 16 KByte-Adreß-bereich des CRTC
14	R/W	cursor adress high	6	Highteil der Adresse der momentanen Cursorposition im 16 KByte-Adreß-bereich des CRTC.
15	R/W	cursor adress low	8	Lowteil der Adresse der momentanen Cursorposition im 16 KByte-Adreß-bereich des CRTC.

Reg Nr	R/W	Name	Anz Bit	Funktion
16	R/-	lightpen adress high	6	Highteil der Adresse der momentanen Lichtstiftposition im 16 KByte-Adreß-bereich des CRTC. Wird durch einen low/high Impuls am LSTROBE-Eingang gesetzt.
17	R/-	lightpen adress low	8	Lowteil der Adresse der momentanen Lichtstiftposition im 16 KByte-Adreß-bereich des CRTC

Der CRTC und die CPU adressieren denselben Speicher. Dennoch sind die Adressen von CRTC und CPU nicht gleichzusetzen! Der Adresse A0 der CPU entspricht z.B. der Zeichentakt des CRTC. So werden während der CRTC eine Adresse an den RAM sendet, zwei aufeinanderfolgende Adressen generiert und 16 Bit aus dem RAM ausgelesen. Die Verknüpfung der anderen Adreßsignale wird in folgender Tabelle gezeigt.

Tab. 2.6 Die Zuordnung von CPU-und CRTC-Adressen

CPU	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15
CRTC	MA0	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9	RA0	RA1	RA2	MA13	MA14

Im Grundzustand des KC compact wird ein Bild folgendermaßen im Bildwiederholspeicher abgelegt:
Ab 0C000H liegt die erste Pixelzeile der ersten Zeichenzeile. Dann folgt die erste Pixelzeile der zweiten Zeichenzeile usw. bis zur ersten Pixelzeile der 24. Zeichenzeile. Ab 0C800H folgt die zweite Pixelzeile der ersten Zeichenzeile. Es folgen die zweiten Pixelzeilen der anderen Zeichenzeilen. So werden nacheinander alle acht Pixelzeilen aller Zeichenzeilen in je 2 KByte großen Bereichen abgespeichert.

2.1.6. Die CIO U82536 /5/

Der KC compact enthält eine CIO (Zähler/Zeitgeber und paralleler Ein/Ausgabebaustein). Dem Anwender stehen die 8 Bit des Ports A am parallelen Druckerinterface zur Verfügung. Wegen der leichten und universellen Programmierbarkeit der Eigenschaften dieser acht Leitungen, empfiehlt sich ihre Nutzung auch zu anderen Zwecken. Dabei muß allerdings beachtet werden, daß die CIO fest in die Hardware des KC compact eingebunden ist.
Die anderen beiden Ports und die drei Zähler/Zeitgeber sind für den Anwender tabu!
Die Steuerwortadresse der CIO ist 0EEXXH. Über diese eine Adresse sind 49 Register erreichbar. Damit das einfach geschehen kann wird in der CIO eine Zustandsmaschine realisiert,die die CIO zwischen den Modi 0 und 1 umschalten kann. Im Mode 0 erwartet die CIO die Ausgabe einer Registernummer auf der Steuerwortadresse. Nach einer Ausgabe auf die Steuerwortadresse wird der Mode 1 eingenommen. Im Mode 1 kann dann das angewählte

Register über die selbe I/O-Adresse beschrieben oder ausgelesen werden. Damit wird aber auch automatisch wieder der Mode 0 eingestellt. Beim Lesen im Mode 0 erhält man die Nummer des zuletzt angewählten Registers und der Mode 0 wird beibehalten. Im Betriebssystem des KC compact wird die CIO nach der Einschaltinitialisierung nur im Mode 0 betrieben.
Im folgenden werden nur die Register und Eigenschaften der CIO beschrieben, die auch vom Anwender genutzt werden können.

Tab.2.7 Vom Anwender nutzbare Register des CIO

Reg.Nr.(Hex)	Bezeichnung,Wirkung
1	Master Configuration Control Register
22	Port A Data Path Polarity Register
23	Port A Data Direction Register
24	Port A Special I/O Control Register

Master-Configuration-Control-Register

Mit Hilfe dieses Registers kann der Port A der CIO während der Umprogrammierung inaktiv geschaltet werden.In der vorliegenden Anwendung sind nur zwei Werte für die Ausgabe auf dieses Register erlaubt,

11110000B, 0F0H,	Port A	inaktiv schalten
11110100B, 0F4H,	Port A	aktiv schalten.

Data-Path-Polarity-Register

Eine '1' in einer einzelnen Bitposition dieses Registers legt den entsprechenden Bitpfad des Port A als invertierend fest, eine '0' als nicht invertierend.

Data-Direction-Register

Eine '1' in einer einzelnen Bitposition dieses Registers legt das entsprechende Bit des Port A als Eingangsbit fest, eine '0' als Ausgangsbit.

Special-I/O-Control-Register

Die Bits in diesem Register wirken unterschiedlich, wenn sie sich auf ein als Eingangsbit definiertes oder ein als Ausgangsbit definiertes Bit des Ports A beziehen.
Wenn ein Bit ein Eingangsbit ist, wird durch eine '1' an der entsprechenden Position des Special-I/O-Control-Registers festgelegt, daß ein "1's-Catcher" in den Eingangspfad geschaltet wird. Der "1's Catcher" hält die "1", wenn sein Eingang (auch nur kurzzeitig) auf "1" geht und kann nur durch das Einschreiben einer "0" in das Eingangsdatenregister (einfache I/O-Ausgabe einer 0 auf die entsprechende Bitposition des Ports A) gelöscht werden. Wurde der Datenpfad als invertierend programmiert, dann bewirkt eine am Eingang anliegende "0" ein auf "1" Setzen des "1's Catchers". Die Ausgabe einer "0"in das Special-I/O-Control-Register legt das entsprechende Bit des Ports A als ein normales Eingangsbit fest.

Wenn ein Bit ein Ausgangsbit ist, wird durch eine '1'an der entsprechenden Position des Special-I/O-Control-Registers der Ausgang als open-Drain-Ausgang konfiguriert, es wird kein pull- up-Transistor bereitgestellt. Die Ausgabe einer "0" in das Special-I/O-Control-Register legt den entsprechenden Ausgang des Ports A als einen normalen Ausgang mit einem pull-up- und einem pull-down-Transistor fest.
Das folgende Programm zeigt, wie Port A umprogrammiert werden kann. Die im Beispiel benutzten Einstellungen gelten für die Nutzung von Port A als CENTRONICS-Schnittstelle. Damit ist die Wiederherstellung des im Betriebssystem benutzten Zustandes der CIO nach eigener Umprogrammierung möglich.

```
;
;Wiederherstellen des ursprünglichen Zustands der CIO
;
CIOPROG:LD      BC,0EE00H+TABEND-TABANF+1;I/O-Adresse, Steuerwort
                                         ;in B, Tablänge in C
        LD      HL,TABANF ;HL enthält aktuelle Adresse
LOOP:   LD      A,(HL)    ;Tabelle in A
        OUT     (C),A     ;und in Steuerregister CIO
        INC     HL        ;nächster Platz in Tabelle
        DEC     C
        JR      NZ,LOOP   ;Tabellenende noch nicht erreicht
        RET

        Tabelle der Ausgabewerte zur Programmierung der CIO,
        es stehen abwechselnd Registernummer und Registerinhalt

TABANF: DB      1         ;Master Configuration Control Register
        DB      0F0H      ;Sperren Port A
        DB      22H       ;Port A Data Path Polarity Register
        DB      80H       ;Bit 7 invertierend
        DB      23H       ;Port A Data Direction Register
        DB      0         ;Alle Bits als Ausgang
        DB      24H       ;Port A special I/O Control Register
        DB      0         ;Normaler Ausgang
        DB      1         ;Master Configuration Control Register
TABEND: DB      0F4H      ;Freigeben Port A
```

2.1.7. Das parallele Druckerinterface

Der KC compact enthält ein paralleles Druckerinterface zur Ansteuerung von Druckern mit CENTRONICS-Schnittstelle. In den Standardroutinen des Betriebssystems werden 7 Bit breite Daten übertragen. Hardwaremäßig ist aber die Übertragung von 8 Bit breiten Daten vorbereitet. Dazu wird als achte Datenleitung das Kassettenausgabesignal mitbenutzt. Im folgenden werden die Signale des parallelen Druckerinterfaces und ihre Lage innerhalb des I/O Adreßraumes vorgestellt.

Tab. 2.8 Die Signale des parallelen Druckerinterfaces im I/O-adreßraum des KC compact

Signal des parallelen Druckerinterfaces	I/O-Adr.	Bit Nr.	Bemerkung
Data 0	0EFXXH	0	Datenleitung zum Drucker
Data 1	0EFXXH	1	Datenleitung zum Drucker
Data 2	0EFXXH	2	Datenleitung zum Drucker
Data 3	0EFXXH	3	Datenleitung zum Drucker
Data 4	0EFXXH	4	Datenleitung zum Drucker
Data 5	0EFXXH	5	Datenleitung zum Drucker
Data 6	0EFXXH	6	Datenleitung zum Drucker
Data 7	0F6XXH	5	Datenleitung zum Drucker gleichzeitig Kassettenausgabe
BUSY	0F5XXH	6	wird vom Drucker auf low gelegt, wenn dieser bereit ist, Daten zu empfangen
/STROBE	0EFXXH	7	wird vom Sender (hier der KC compact) kurzzeitig auf low gelegt, wenn Daten gültig

Um auch Graphiken ausdrucken zu können, sind acht Bit nötig. Eine entsprechende Routine wird im Abschnitt 3.11.vorgestellt.

2.1.8. Das ROM Select

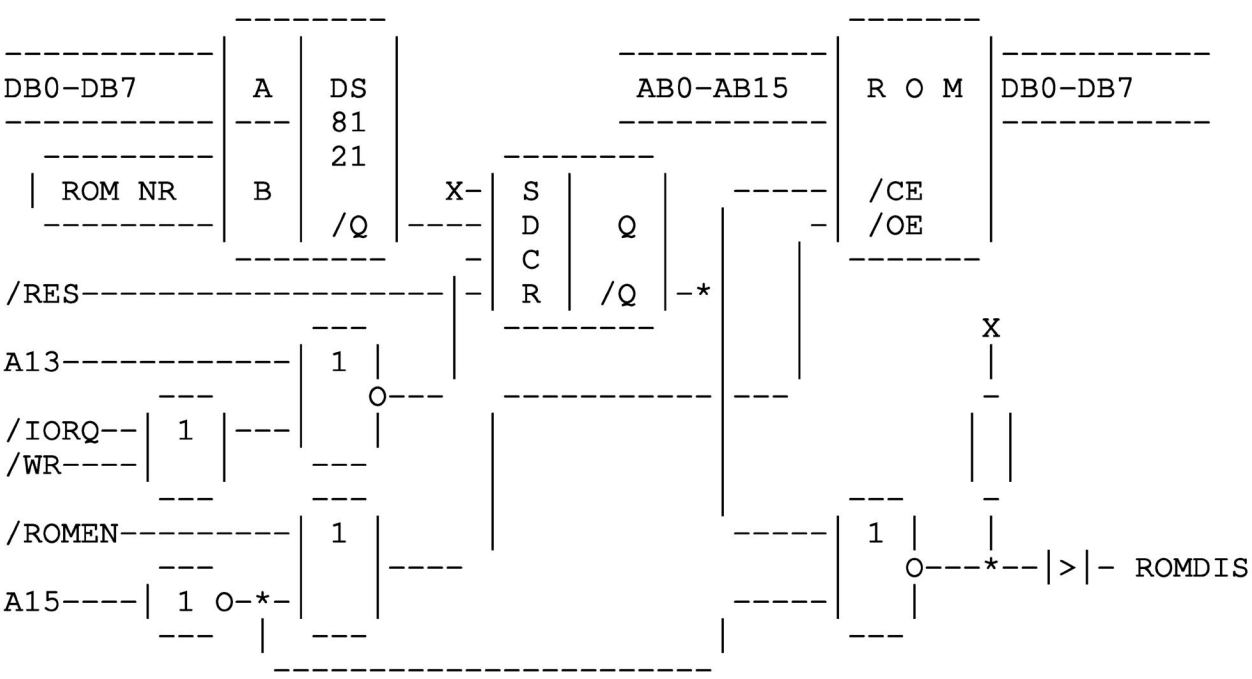


Abb. 2.4 Die Ansteuerschaltung für ein externes ROM

Der KC compact kann externe ROMs verwalten. Durch das Zusammenspiel der Signale /ROMEN und ROMDIS am Expansionsinterface wird dafür gesorgt, daß immer nur ein ROM selektiert wird. Ein externer ROM erhält eine Nummer, die von seiner Ansteuerschaltung erkannt wird. Durch die Ausgabe dieser Nummer auf die I/O-Adresse

0DFXXH wird der externe ROM auf die Adresse 0C000H geschaltet und steht für Lesezugriffe zur Verfügung. Der interne ROM bzw. ein anderer, vorher aktiver ROM, wird inaktiv geschaltet. Dazu muß der externe ROM über Schaltung nach Abb. 2.4 an den Expansionssteckverbinder angeschlossen werden.

2.1.9.Der parallele Dreitorbaustein /5/(PIO)KP580B55

Für die Bedienung des Soundcontrolers, der Tastatur, des extern anschließbaren Kassettengerätes, zum softwaremäßigen Erkennen eines Strahlrücklaufes beim ausgegebenen Monitor- oder Fernsehbild und zum Erkennen bestimmter Expansionsbaugruppen wird ein Baustein mit drei parallelen, acht Bit breiten Ein-/Ausgabeports verwendet, der KP580B55. Er ist zum I 8255 kompatibel. Er enthält ein Steuerregister und drei Datenregister. Da dieser Baustein fest in die Hardware des KC compact eingebunden ist, können nicht alle Möglichkeiten zur Programmierung der Bausteinfunktionen genutzt werden. Aus diesem Grunde werden in der folgenden Tabelle nur die für die vorgegebene Anwendung sinnvollen Ausgaben auf das Steuerregister beschrieben. Die I/O-Adresse des Steuerregisters ist 0F7XXH.

Tab.2.9 vom Anwender nutzbare Steuerbytes für die PIO

Steuerbyte		Funktion
10010010	92H	Port A Eingang, Port B Eingang, Port C Ausgang
10000010	82H	Port A Ausgang, Port B Eingang, Port C Ausgang
0XXX1111	0FH	Setzen Bit 7 in Port C alle anderen Bits konstant
0XXX1110	0EH	Reset Bit 7 in Port C alle anderen Bits konstant
0XXX1101	0DH	Setzen Bit 6 in Port C alle anderen Bits konstant
0XXX1100	0CH	Reset Bit 6 in Port C alle anderen Bits konstant
0XXX1011	0BH	Setzen Bit 5 in Port C alle anderen Bits konstant
0XXX1010	0AH	Reset Bit 5 in Port C alle anderen Bits konstant
0XXX1001	09H	Setzen Bit 4 in Port C alle anderen Bits konstant
0XXX1000	08H	Reset Bit 4 in Port C alle anderen Bits konstant
0XXX0111	07H	Setzen Bit 3 in Port C alle anderen Bits konstant
0XXX0110	06H	Reset Bit 3 in Port C alle anderen Bits konstant
0XXX0101	05H	Setzen Bit 2 in Port C alle anderen Bits konstant
0XXX0100	04H	Reset Bit 2 in Port C alle anderen Bits konstant
0XXX0011	03H	Setzen Bit 1 in Port C alle anderen Bits konstant
0XXX0010	02H	Reset Bit 1 in Port C alle anderen Bits konstant
0XXX0001	01H	Setzen Bit 0 in Port C alle anderen Bits konstant
0XXX0000	00H	Reset Bit 0 in Port C alle anderen Bits konstant

Port A

Der Port A ist mit den 8 Datenleitungen des Soundcontrolers verbunden. Je nach geforderter Aktion wird Port A als Eingang oder Ausgang programmiert. Die I/O-Adresse für Port A ist 0F4XXH.

Port B

Der Port B wird als Eingabeport betrieben. Die I/O-Adresse ist 0F5XXH. Die einzelnen Bits haben folgende Bedeutung:

Bit 0: VSYNC des CRTC. Durch einfaches Rotieren dieses Bits in das CY-Flag kann sehr schnell festgestellt werden, wann ein Strahlrücklauf stattfindet.
Bit 1-4: Reserviert für Test, Inbetriebnahme und Reparatur.
Bit 5: Dient zur Abfrage des Zustandes der Leitung EXP des Expansionsinterfaces. Durch Lowpotential auf dieser Leitung kann ein angesteckter Peripheriebaustein seine Existenz melden.
Bit 6: Ist mit der BUSSY-Leitung des Druckerinterfaces verbunden. Durch Lowpotential zeigt ein angeschlossener Drucker, daß er Zeichen empfangen kann.
Bit 7: Ist über einen Leseverstärker mit dem Kassetteninterface verbunden. Hier werden Daten und Programme vom Kassettengerät eingelesen.

Port C

Der Port C wird als Ausgabeport benutzt. Die I/O-Adresse ist 0F6XXH. Die einzelnen Bits haben folgende Bedeutung:

Bit 0-3: Diese Bits steuern einen 1-aus-10-Dekoder,der die gewünschte Zeilenleitung der Tastaturmatrix auf low legt.
Bit 4: Motorschaltspannung für Kassettengeräte mit schaltbarem Motor. 0:Motor aus, 1:Motor an.
Bit 5: Kassettenausgabe, Bitwechsel an diesem Ausgang werden bei Aufnahme vom Kassettengerät als Programme oder Daten aufgezeichnet. Gleichzeitig dient dieses Bit als DATA 7 im CENTRONICS-Interface.
Bit 6: Verbunden mit BC1 des Soundcontrolers (chip select).
Bit 7: Verbunden mit BDIR des Soundcontrolers.

2.1.10. Der Soundcontroler AY 9-8912 /6/

Die Tonerzeugung und die Tastaturabfrage wird im KC compact durch einen Soundcontroler durchgeführt. Über 14 Register können alle Klangmöglichkeiten dieses Schaltkreises programmiert werden. Über das I/O-Port des Soundgenerators wird der Zustand der Spaltenleitungen der Tastaturmatrix eingelesen. In Abb.2.5 wird ein einfaches Blockschaltbild des Soundcontrolers gezeigt. Der Soundcontroler wird durch die PIO angesteuert. Der Datenbus ist mit dem Port A gekoppelt, die Steuersignale BDIR und BC1 werden von Port C bereitgestellt. Sie haben folgende Funktionen:

Tab. 2.10 Die Bedeutung der Steuersignale BDIR und BC1 des Soundcontrolers

BDIR	BC1	Funktion
1	1	Setzen des Registerzeigers. Der Wert an den Datenleitungen des Soundcontrolers wird als Registernummer interpretiert, das entsprechende Datenregister steht bis zur nächsten Ausgabe auf den Registerzeiger für Ein/Ausgaben zur Verfügung.
1	0	WRITE, Daten können in das angewählte Register geschrieben werden.
0	1	READ, Daten können aus dem angewählten Register gelesen werden.
0	0	Der Datenbus des Soundcontrolers wird hochohmig geschaltet.

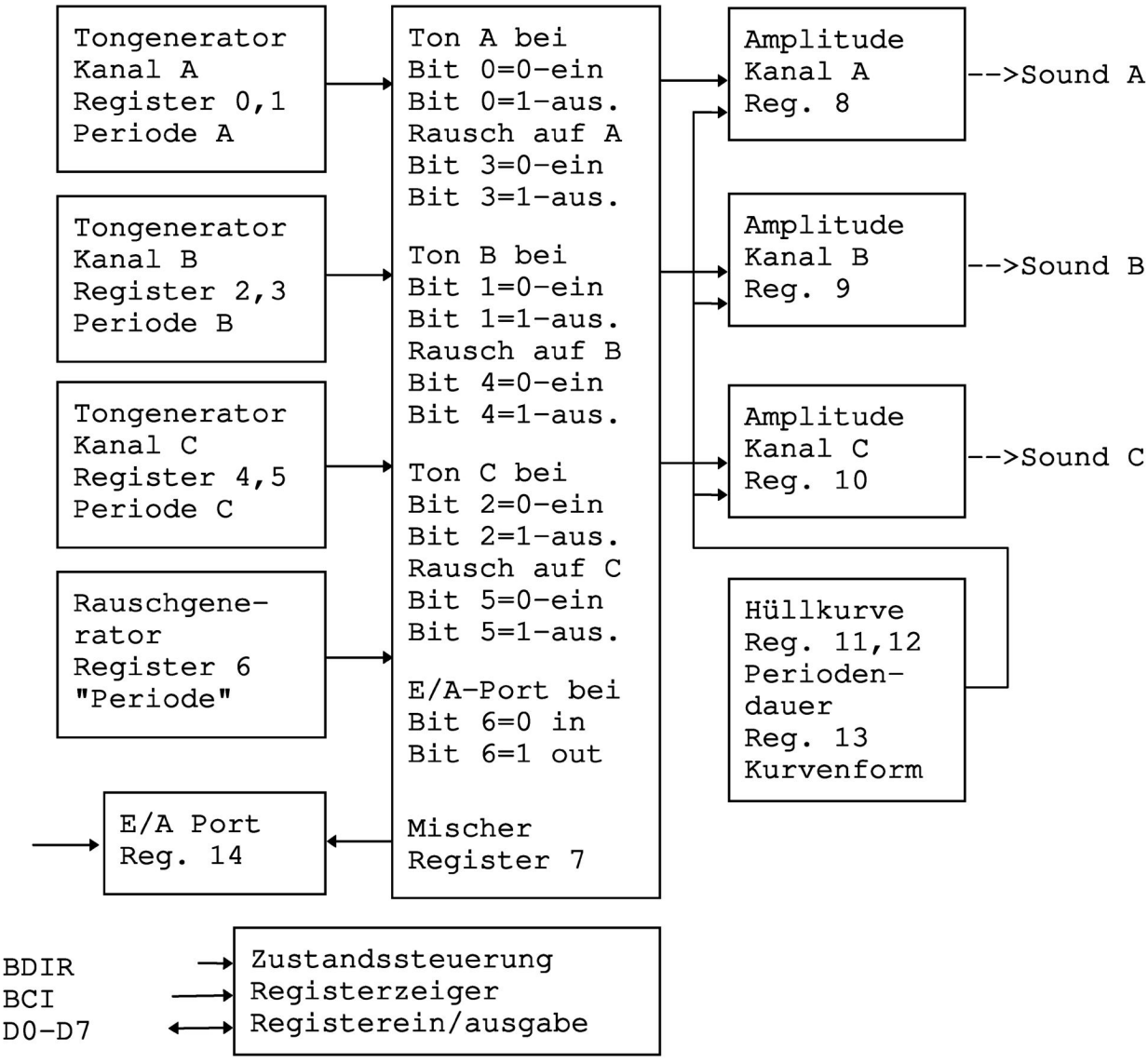


Abb.2.5 Blockschaltbild des Soundcontrolers

Der Soundcontroler wird mit einer Taktfrequenz von 1MHz betrieben. Der 12-Bit-Wert der Periodendauer wird für die einzelnen Kanäle nach der Formel P=ROUND(62500/FREQUENZ) berechnet (Frequenz in Hz).
Es folgt eine Beschreibung der einzelnen Register.

Tab.2.11 Die Steuerregister des Soundcontrolers

Reg Nr	Anz Bit	Funktion
0	8	Lowteil der Periodendauer für Kanal A
1	4	Highteil der Periodendauer für Kanal A
2	8	Lowteil der Periodendauer für Kanal B
3	4	Highteil der Periodendauer für Kanal B
4	8	Lowteil der Periodendauer für Kanal C
5	4	Highteil der Periodendauer für Kanal C
6	5	Periodendauer für das rauschähnliche Signal
7	7	Mischersteuerung, die Funktion der einzelnen Bits: Bit 6=0: Port A Eingang, =1: Port A Ausgang Bit 5=0: Rauschen zu Kanal C zuschalten, =1: abschalten Bit 4=0: Rauschen zu Kanal B zuschalten, =1: abschalten Bit 3=0: Rauschen zu Kanal A zuschalten, =1: abschalten Bit 2=0: Ton von Kanal C einschalten, =1: ausschalten Bit 1=0: Ton von Kanal B einschalten, =1: ausschalten Bit 0=0: Ton von Kanal A einschalten, =1: ausschalten
8	5	Lautstärke Kanal A Bit 4=0: Bit 0 bis 3 Lautstärke (logarithmisch) Bit 4=1: Lautstärke wird durch Hüllkurvengenerator bestimmt, Bit 0 bis 3 ohne Wirkung
9	5	Lautstärke Kanal B, wie Register 8 für Kanal A
10	5	Lautstärke Kanal C, wie Register 8 für Kanal A
11	8	Lowteil der Periodendauer der Hüllkurve
12	8	Highteil der Periodendauer der Hüllkurve
13	4	Kurvenform der Hüllkurve

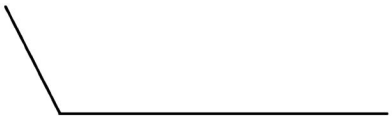
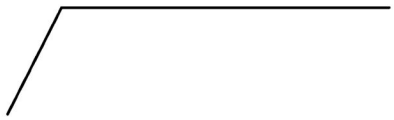
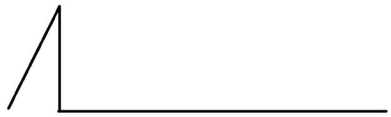
2.1.11 Die Tastaturansteuerung

Die Spalten der Tastaturmatrix werden von den Ausgängen eines 1-aus-10-Dekoders getrieben, der durch Port C der PIO angesteuert wird. Die Zeilen der Tastaturmatrix werden durch das I/O-Port des Soundcontrolers und im weiteren durch Port A der PIO eingelesen. Im folgenden wird an einem Beispielprogramm gezeigt, wie das geschieht. Die [SPACE]-Taste soll abgefragt werden. Diese große Taste eignet sich besser für schnelle Reaktionen als z.B. die [ESCAPE]-Taste. Das Programm kommt ohne Interrupt aus. Im allgemeinen empfiehlt sich aber die Nutzung der Systemprogramme.

Test auf [SPACE]-Taste. Die Routine benutzt nur AF und BC

```
SPACE:  LD    BC,0F40Eh   ;PIO-Port A in B,Soundregisternummer
                          ;I/O-Port in C
        OUT   (C),C       ;Anwahl des I/O-Ports des Soundcontr.
        LD    B,0F6h      ;PIO-Port C
        IN    A,(C)       ;Aktuelle Belegung merken
        AND   30H         ;Die Pegel auf der Motorschaltspannung
                          ;und der Kassettenausgabe sollen nicht
                          ;verändert werden
        OR    05H         ;Zeilenleitung 5 Tastaturmatrix
        LD    C,A         ;Anwählen und in C
        OR    0C5H        ;Setzen von BDIR und BC1
                          ;für Soundcontroler
        OUT   (C),A       ;Zeilenleitung wählen
        OUT   (C),C       ;Sound-Controler Setzen des Registerzei-
                          ;gers auf I/O-Port (siehe PIO-Port A)
        INC   B           ;PIO-Steuerwort
        LD    A,92H       ;PIO-Port A als Eingang
        OUT   (C),A       ;programmieren
        PUSH  BC
        SET   6,C         ;Nur BC1 für Sound-Controler setzen
        LD    B,0F6H      ;PIO Port C
        OUT   (C),C       ;BC1 an Sound-Controler legen (READ)
        LD    B,0F4H      ;PIO Port A
        IN    A,(C)       ;Einlesen Zeileninformation der Tastatur-
                          ;matrix
        CP    7FH         ;Ist Bit 7 auf Lowpotential?
        POP   BC
        LD    A,82H       ;PIO-Port A wieder als Ausgang
        OUT   (C),A       ;Programmieren
        LD    B,0F6H      ;I/O-Adresse PIO-Port C
        OUT   (C),C       ;Alte Belegung von PIO-PORT C wieder
        RET               ;War die [SPACE]-Taste gedrückt,
                          ;ist Z-Flag gesetzt.
```

Tab. 2.12 Hardware-Volumenhüllkurven des Sound-Controllers

Reg. 13 (binär)	Hüllkurvenform	Reg. 13 (binär)	Hüllkurvenform
xxxx1000		xxxx1100	
xxxx1001		xxxx1101	
xxxx1010		xxxx1110	
xxxx1011		xxxx1111	

3. S O F T W A R E - B E T R I E B S S Y S T E M

Der KC compact enthält neben dem linear verwalteten 64 KByte-RAM zwei 16 KByte große Softwarepakete als ROM. Das sind zum einen das Betriebssystem und zum anderen der BASIC-Interpreter. In den folgenden Abschnitten sollen die Bestandteile des Betriebssystems und die Einteilung aller Speicherbaugruppen näher beschrieben werden.

3.1. Das Systemkonzept

Der KC compact meldet sich beim Einschalten (Kaltstart mit dem BASIC-Interpreter und muß auch mit BASIC-Befehlen bedient werden. Durch das Nachladen von Maschinenprogrammen (Compiler, Assembler, Textverarbeitungsprogramme, Grafikprogramme, Spielprogramme usw.) ist der Anwender aber nicht allein auf die BASIC-Nutzung beschränkt.
Das Betriebssystem ermöglicht dem Anwender

- die Systemressourcen durch eine große Anzahl von Systemunterprogrammen vollständig zu nutzen,
- Systemunterprogramme problemlos durch eigene zu ersetzen,
- eigene Maschinenprogramme selbständig oder als Befehlserweiterungen (RSX) zu nutzen,
- den eigenen Unterprogrammen bei Aufruf von BASIC oder anderen Programmen aus bis zu 32 Parameter zu übergeben,
- extern angesteckte Hard-und Software (z.B. Diskettenstation, Zusatz-ROM) zu nutzen,
- die Anfangsadresse des Video-RAM von C000H auf 0, 4000H oder 8000H zu verschieben (nur 4000H ist sinnvoll), so daß zwei Bilder gleichzeitig im RAM gehalten werden können,
- zwischen maximal 20, 40 oder 80 Zeichen pro Bildschirmzeile zu wählen,
- jede Taste der Tastatur umzuprogrammieren (Zeichencodes und Repeat-Verhalten),
- die Funktionstasten mit Zeichenketten (z.B. BASIC-Befehle oder Abarbeitungsfolgen) zu belegen,
- für die Darstellung von Zeichen auf dem Bildschirm beliebige Zeichenbildtabellen (Zeichengeneratoren) zu verwenden (z.B. kyrillische Buchstaben oder Grafikzeichen).

3.2. Die Speicheraufteilung

Für das Betriebssystem und den BASIC-Interpreter sind einige Bereiche des RAM reserviert.Die folgende Tabelle gibt einen groben Überblick.

Tab.3.1 Speicheraufteilung im KC compact

Adreßbereich (hexadezimal)	Bemerkungen
0000 - 003F	Low Kernel Jumpblock (Restarts)
0040 - ABFF	Frei für Anwenderprogramme
AC00 - B0FF	RAM des BASIC-Interpreters
B100 - B113	RAM für Fließkommazahlenrechnung
B114 - B117	RAM des Zeileneditors
B118 - B1EC	RAM der Kassetten-Routinen
B1ED - B495	RAM der Sound-Routinen
B496 - B692	RAM der Keyboard-Routinen
B693 - B6B4	RAM der Grafik-Routinen
B6B5 - B733	Fenstervektorspeicher (Fenster 0 bis 7) und Vektor des aktuellen Fensters
B734 - B7C2	RAM der Text-Routinen
B7C3 - B803	RAM der Screen-Routinen
B804 - B82C	Drucker-Übersetzungstabelle
B82D - B8FF	RAM der Kernel-Routinen
B900 - B920	High Kernel Jumpblock
B921 - BAE3	In den RAM kopierte Betriebssystemroutinen
BB00 - BDF4	Vektoren für die Betriebssystem-Unterprogramme
BDF5 - BFFF	Systemstack
C000 - FFFF	Bildwiederholspeicher

3.3. Die Magnetbandaufzeichnung

3.3.1. Verfahren

Die Daten werden mit Rechteckschwingungen auf Kassette aufgezeichnet. Dazu werden die einzelnen Bits sequentiell geschrieben, wobei für jedes Bit eine volle Schwingung verwendet wird. Alle Bytes werden mit dem Bit 7 beginnend abgespeichert. Die Eins-Bits werden gegenüber den Null-Bits mit doppelter Periodenlänge aufgezeichnet. Die tatsächliche Periodenlänge bzw. Frequenz ist von der Aufzeichnungsgeschwindigkeit abhängig, die in weiten Grenzen wählbar ist. Übertragungsraten zwischen ca. 700 und 2500 Baud sind über das Betriebssystem wählbar, wobei BASIC mit den Standardwerten 1000 oder 2000 Baud arbeitet.
Die analoge Elektronik der Kassettenrecorder neigt dazu, die Signalflanken zwischen einer langen (Eins-Bit) und einer kurzen Periode (Null-Bit) zu verschieben. D.h., daß bei einem Wechsel von Null-auf Eins-Bit und umgekehrt die Unterschiede zwischen langer und kurzer Periode geringer werden. Das Aufzeichnungsverfahren berücksichtigt diese Tatsache und arbeitet mit einer Vorkompensation. Bei Eins-Null-Übergängen werden die kurzen Perioden noch kürzer und die langen Perioden noch länger gemacht, so daß beim Laden von der Kassette korrekte Werte gelesen werden.

3.3.2. Dateiaufbau

Jede Datei wird zum Abspeichern in 2 KByte lange Blöcke unterteilt. Ein Block setzt sich aus zwei Records,dem Header- und dem Datenrecord, zusammen.

Jeder Record ist wie folgt aufgebaut:

```
 2048 Eins-Bits : Vorton, der zur Synchronisation und zur
                  Ermittlung der Aufzeichnungsgeschwindigkeit
                  dient
    1 Null-Bit  : Vortonendekennung
    1 Byte      : Synchronisationszeichen :2CH für Headerrecord
                                           16H für Datenrecord
```

Danach folgen die eigentlichen Daten

```
Header:
  64  Bytes     : Headerdaten
   2  Bytes     : Prüfsumme über die Daten
Daten:
 256  Bytes     : Daten (der 2048 Bytes lange Block wird in 256
                  Bytes lange Abschnitte unterteilt)
   2  Bytes     : Prüfsumme über die vorangehenden 256 Datenbytes
   .
   .
   .
 256  Bytes     :Daten
   2  Bytes     :Prüfsumme
```

Ist der letzte Abschnitt kürzer als 256 Bytes, wird er mit Null-Bytes aufgefüllt. Nach dem letzten Abschnitt werden noch einige Eins-Bits ausgegeben. Sie gewährleisten, daß die letzten Datenbytes verzerrungsfrei aufgezeichnet werden.

Der Header enthält folgende Informationen:

```
Byte  0 bis 15 : Dateiname
Byte 16        : Blocknummer
Byte 17        : Kennbyte für letzten Block (<>0 -> letzter
                 Block)
Byte 18        : Dateityp
                 Bit 0   - Dateischutz ( =0 -> ungeschützt)
                 Bit 1-3 - =0 -> BASIC-Programm
                           =1 -> Maschinencode
                           =2 -> Bildschirmabzug
                           =3 -> ASCII-Datei
                           4-7 nicht definiert
Byte 19 und 20 : Länge des Datenrecords
Byte 21 und 22 : Quelladresse des Datenrecords
Byte 23        : Kennbyte für ersten Block (>0 ->erster Block)
Byte 24 und 25 : Gesamtlänge der Datei
Byte 26 und 27 : Startadresse bei Maschinencodeprogrammen
Byte 28 bis 63 : nicht verwendet
```

3.3.3. Fehlermeldungen

Bei der Nutzung von Kassetten-Systemunterprogrammen werden von diesen Routinen Meldungen zum Hauptprogramm zurückgegeben, die im Fehlerfall die Art des Fehlers erkennen lassen. Ist das CY-Flag gesetzt, liegt kein Fehler vor. Bei CY=0 wird im Z-Flag und im A-Register die Fehlerart gekennzeichnet. Die folgende Tabelle gibt einen Überblick.

Tab.3.2 Fehlermeldungen der Kassettenroutinen

Z-Flag	A-Register	Bedeutung
=1	=0	die Routine wurde mit der Betätigung der [ESC]-Taste unterbrochen (Break)
=0	=1	SPEED zu kleine Periodenlängen vereinbart (BASIC-Meldung: Write error a) - beim Lesen wurde eine unleserlich lange Periode erkannt (BASIC-Meldung: Read error a)
=0	=2	beim Lesen einer Datei trat ein Prüfsummen fehler auf (BASIC-Meldung: Read error b)
=0	=3	bei Vergleich der Daten auf der Kassette mit einem Speicherbereich wurde ein Unterschied festgestellt
=0	=0EH	die Datei ist nicht eröffnet bzw.es ist eine andere Datei eröffnet
=0	=0FH	das Ende der Datei ist erreicht

Eine Ausnahme besteht bei der Katalogfunktion (BC9BH CAS CATALOG). Hier muß die Routine mit [ESC]unterbrochen werden. Trotzdem wird kein Fehler mit CY=0 vermerkt.

3.3.4. Dateitypen

Um einzelne Dateitypen voneinander unterscheiden zu können, ist eine Kennzeichnung erforderlich. Hier haben sich bereits einige Kürzel eingebürgert bzw. bewährt. Im folgenden sind einige aufgeführt:

ASC - für ASCII-Dateien
BAS - für BASIC-Programme
BIN - für Maschinencode bzw.Speicherabzüge allgemein
COM - für ausführbare Maschinenprogramme
MAC - für Assembler-Quellcode
PAS - für PASCAL-Quellcode
PIC - wie SCR
SCR - für Abzüge des Video-RAM (Screen)
TXT - für Textdateien
usw.

3.4. Die Bildschirmausgaben

3.4.1. Video-RAM

Im KC compact wird eine Video-RAM-Größe von zusammenhängend 16 KByte verwendet.Hier ist für jeden Bildpunkt (Pixel)die Farbe, genauer die Tintennummer (siehe Abschnitt 3.4.2.), abgespeichert. Sämtliche Bildschirmausgaben (Text und Grafik)werden durch Änderung der Farb-(Tinten-)Information einzelner Bildpunkte realisiert. Die Lage des Video-RAM ist standardmäßig der Adreßbereich C000H bis FFFFH. Es können aber auch die Speicherbereiche 0 bis 3FFFH, 4000H bis 7FFFH und 8000H bis BFFFH dazu verwendet werden. Da jedoch in diesen RAM-Blöcken,außer im Block von 4000H bis 7FFFH, Sprungleisten, Arbeitszellen, Tabellen und die Interruptroutine des Betriebssystems und des BASIC-Interpreters liegen, kommt dafür nur der Bereich 4000H-7FFFH in Frage. Wie die Nutzung auch von BASIC aus möglich ist, zeigt das folgende kleine Beispiel:

```
 10 'zwei Bilder im KC compact
 20 MODE 1:MEMORY &3FFF
 30 PAPER 0:PEN 1:LOCATE 6,12:PRINT "Bild im Bereich C000H-FFFFH"
 40 GOSUB 90:PAPER 2:PEN 3
 50 CLS:LOCATE 6,12:PRINT "Bild im Bereich 4000H-7FFFH"
 60 GOSUB 130:GOSUB 70:GOSUB 90:GOSUB 70:GOTO 60
 70 'Warteschleife
 80 A=TIME:WHILE TIME<A+1000 :WEND:RETURN
 90 'Umschalten auf 4000H-7FFFH
100 POKE &B7C6,&40:'Basisadresse dem Betriebssystem mitteilen
110 OUT &BCFF,12:OUT &BDFF,&10:'Video-Controller umprogrammieren
120 RETURN
130 'Umschalten auf C000H-FFFFH
140 POKE &B7C6,&C0
150 OUT &BCFF,12:OUT &BDFF,&30
160 RETURN
```

Wie aus dem Beispiel zu sehen ist,muß der CRTC umprogammiert werden. Außerdem muß dem Betriebssystem die neue Basisadresse mitgeteilt werden, da sonst die "Bildschirmausgaben"weiterhin in den alten Adreßbereich erfolgen würden. Dieser Umstand kann auch ausgenutzt werden. Während das eine Bild angezeigt wird, kann das zweite beschrieben werden und umgekehrt.

Adreßrechnung im Video-RAM

Wie aus /7/ hervorgeht,können je nach eingestelltem Bildschirmmodus 200 *160 (16 aus 27 Farben), 200 *320 (4 aus 27 Farben) und 200 *640 (2 aus 27 Farben)Punkte mit derselben Anzahl von Speicherzellen dargestellt werden. Der Grund dafür ist, daß pro Byte im RAM 2, 4 oder 8 Pixel codiert sind. Deshalb auch die unterschiedliche Anzahl möglicher Farben in den einzelnen Modi. Aus der Pixelanzahl pro Byte und der maximalen horizontalen Pixelauflösung ergibt sich, daß pro Punkt-Zeile 80 Bytes vorgesehen sind. Die Abb. 3.1. zeigt die Adreßeinteilung für den Fall, daß noch nicht hardwaremäßig gescrollt wurde. Das ist immer dann der Fall, wenn der Bildschirm mit CLS oder MODE x gelöscht wurde. Aus dem Bild ist ersichtlich, daß der RAM in acht

2 KByte lange Bereiche eingeteilt ist. Ein Bereich entspricht immer einer bestimmten Pixel-Zeile aller Textzeilen. Im ersten Bereich von C000H bis C7FFH sind z.B. alle Grafikinformationen für die obersten Pixel-Zeilen aller 8*8-Punkte-Matrizen aller Zeichenpo- sitionen abgespeichert. Im folgenden Beispiel wird das deutlich:

```
10 CLS: FOR I=1 TO 24: PRINT "Test programm": NEXT
20 FOR I=&C000 TO &C7FF: POKE I, &FF: NEXT
30 WHILE INKEY$="": WEND
```

Weiterhin ist zu sehen, daß pro Bereich nur 80*25=2000 Bytes gebraucht werden. Da jedoch 2 KByte (2048)vorhanden sind, bleiben 48 Bytes unbenutzt. Das ist aber nur so lange der Fall, wie noch nicht hardwaremäßig gescrollt wurde.

```
C000 C001 C002 C003  ...  C04C C04D C04E C04F Pixel-Zeile 1
C800 C801 C802 C803  ...  C84C C84D C84E C84F Pixel-Zeile 2
D000 D001 D002 D003  ...  D04C D04D D04E D04F Pixel-Zeile 3
D800 D801 D802 D803  ...  D84C D84D D84E D84F Pixel-Zeile 4
E000 E001 E002 E003  ...  E04C E04D E04E E04F Pixel-Zeile 5
E800 E801 E802 E803  ...  E84C E84D E84E E84F Pixel-Zeile 6
F000 F001 F002 F003  ...  F04C F04D F04E F04F Pixel-Zeile 7
F800 F801 F802 F803  ...  F84C F84D F84E F84F Pixel-Zeile 8
C050 C051 C052 C053  ...  C09C C09D C09E C09F Pixel-Zeile 9
C850 C851 C852 C853  ...  C89C C89D C89E C89F Pixel-Zeile 10
D050 D051 D052 D053  ...  D09C D09D D09E D09F Pixel-Zeile 11
D850 D851 D852 D853  ...  D89C D89D D89E D89F Pixel-Zeile 12
E050 E051 E052 E053  ...  E09C E09D E09E E09F Pixel-Zeile 13
E850 E851 E852 E853  ...  E89C E89D E89E E89F Pixel-Zeile 14
F050 F051 F052 F053  ...  F09C F09D F09E F09F Pixel-Zeile 15
F850 F851 F852 F853  ...  F89C F89D F89E F89F Pixel-Zeile 16
C0A0 C0A1 C0A2 C0A3  ...  C0EC C0ED C0EE C0EF Pixel-Zeile 17
C8A0 C8A1 C8A2 C8A3  ...  C8EC C8ED C8EE C8EF Pixel-Zeile 18
        .                         .
        .                         .
        .                         .
F730 F731 F732 F733  ...  F77C F77D F77E F77F Pixel-Zeile 191
FF30 FF31 FF32 FF33  ...  FF7C FF7D FF7E FF7F Pixel-Zeile 192
C780 C781 C782 C783  ...  C7CC C7CD C7CE C7CF Pixel-Zeile 193
CF80 CF81 CF82 CF83  ...  CFCC CFCD CFCE CFCF Pixel-Zeile 194
D780 D781 D782 D783  ...  D7CC D7CD D7CE D7CF Pixel-Zeile 195
DF80 DF81 DF82 DF83  ...  DFCC DFCD DFCE DFCF Pixel-Zeile 196
E780 E781 E782 E783  ...  E7CC E7CD E7CE E7CF Pixel-Zeile 197
EF80 EF81 EF82 EF83  ...  EFCC EFCD EFCE EFCF Pixel-Zeile 198
F780 F781 F782 F783  ...  F7CC F7CD F7CE F7CF Pixel-Zeile 199
FF80 FF81 FF82 FF83  ...  FFCC FFCD FFCE FFCF Pixel-Zeile 200
```

Abb. 3.1 Adreßrechnung im Video-RAM, wenn der Video-RAM mit der Basis-Adresse C000H programmiert ist (hardwaremäßig wurde noch nicht gescrollt).

Beim KC compact gibt es zwei Möglichkeiten den Bildschirminhalt zu scrollen.Die langsamere ist, die entsprechenden Speicherbereiche durch Verschiebebefehle zu verschieben, was immer dann gemacht werden muß, wenn ein Fenster definiert wurde und gescrollt werden soll. Die schnellere Möglichkeit ist das hardwaremäßige Scrolling, indem der Video-Controller mit einer neuen Basis-Adresse programmiert wird. Das erfolgt immer dann, wenn der gesamte Bildschirminhalt gescrollt werden muß. Für das Scrollen des Bildschirms nach oben muß nur die Basis-Adresse um 80 Bytes erhöht werden. Nun ist auch ersichtlich, daß die ursprünglich freien 48 Bytes nach dem ersten Scrolling im "sichtbaren" Bereich liegen. Da die Basis-Adresse aber um 80 Bytes verschoben wurde, reicht die Endadresse um 32 Byte über den 2 KByte-Bereich hinaus. Die acht 2 KByte-Bereiche sind acht Ringspeicher. Das was oben aus dem Bildschirm "herausgeschoben" wird, erscheint um 48 Bytes versetzt am unteren Bildrand wieder. Die untere Textzeile muß deshalb auch beim Hardwarescrolling immer gelöscht bzw. neu beschrieben werden. Zur Adressierung von 2 KByte werden 11 Adreßbits benötigt. Das sind die Adreßleitungen MA0 bis MA9 und RA0 des Video-Controlers. Dadurch, daß die Adreßleitungen MA10 und MA11 des Controlers aber nicht angeschlossen sind, ergibt sich beim Incrementieren von 7FFH nicht 800H sondern 0. Die Bildausgabe wird also am Anfang des jeweiligen 2 KByte- Bereiches fortgesetzt. Die Adreßrechnung erschwert sich durch das Hardwarescrollen also erheblich.
Alle notwendigen Adreßrechnungen können durch die entsprechenden Unterprogramme des Betriebssystems (SCREEN PACK) ausgeführt werden.

3.4.2. Farben auf dem KC compact

Die Information, mit welcher Farbe welches Pixel dargestellt wird, ist im KC compact geteilt. Es wird wie im Abschnitt 3.4.1. bereits angedeutet, im Video-RAM zu jedem Pixel die Tintennummer abgespeichert. Die Anzahl der zur Verfügung stehenden Tinten ist vom eingestellten Modus abhängig:
Mode 0 - 16 Tinten
Mode 1 - 4 Tinten
Mode 2 - 2 Tinten

Jede der verfügbaren Tinten kann mit einer von 27 Farben belegt werden. Die Information, welche Farbe welcher Tinte zugeordnet würde, ist in der Zentralen Zustandssteuerung gespeichert (siehe Abschn. 1.1.4.) und kann dort sehr schnell umprogrammiert werden. Damit wird erreicht, daß alle Pixel, die mit der umprogrammierten Tinte dargestellt werden, sofort mit der neuen Farbe auf dem Bildschirm erscheinen.
Vom Betriebssystem sind standardmäßig für jede Tinte jedoch zwei Farben vorgesehen, die in regelmäßigen Zeitabständen wechseln (bliken). Soll die Tinte nicht blinken, müssen beide Farben übereinstimmen. Die Periodenlängen zum Umschalten von einer auf die andere Farbe kann eingestellt werden (SPEED INK). Das Umschalten selber wird immer dann vorgenommen, wenn ein Strahl

rücklauf von unten nach oben am Bildschirm stattfindet. Dazu wird eine der drei möglichen Interrupts, nämlich der FRAME FLYBACK (siehe Abschn.3.7.),verwendet.Das ist notwendig,damit nicht zufällig genau beim Bildauslesen die Farbe einer Tinte gewechselt wird. Ein und dieselbe Tinte würde dann für die kurze Zeit bis zum nächsten Bildauslesen oben auf dem Schirm eine andere Farbe haben als unten.
Beim Kaltstart des Betriebssystems werden alle Tinten mit Standardfarben belegt /7/.
Die BASIC-Farbwerte, die beim Festlegen der Farbe für eine Tinte angegeben werden, sind andere Werte als die, mit denen die Zentrale Zustandssteuerung programmiert wird. Die Farbwerte werden über eine Tabelle erst in die sogenannten Paletten-Farb- werte konvertiert. In BASIC werden die Farben nach ihrer Helligkeit auf einem monochromen Monitor sortiert und neu numeriert. Den Zusammenhang zwischen BASIC-Farbwert, Paletten- Farbwert und Farbe stellt Tabelle 2.2 im Abschnitt 2.1.4.dar.

Codierung der Tintennummer im Video-RAM

Wie bereits im Abschnitt 3.4.1. angedeutet, werden pro Byte 2, 4 oder 8 Pixel codiert. Am einfachsten ist die Codierung im Bildschirmmodus 2, da jedes Bit genau einem Pixel entspricht. Das Bit 7 ist dem Pixel ganz links zugeordnet. Mit einem Bit können zwei Zustände (2 Tinten)verschlüsselt sein. Ist z.B. ein Bit gesetzt, wird der entsprechende Punkt mit Tinte 1 dargestellt. Im Modus 1 wird die Tintennummer mit zwei Bit und im Modus 0 mit vier Bit codiert. Die Zuordnung der Bits zu den Pixeln und die Wertigkeit der Bits zur Tintennummernbestimmung ist der nachfolgenden Darstellung zu entnehmen:

Byte im Video-RAM:	7	6	5	4	3	2	1	0

Die Pixel sind in den Tabellen immer von links nach rechts aufgeführt.

Bildschirmmodus 0:

Pixel Bits im Video-RAM-Byte

1	1	5	3	7
2	0	4	2	6

Bildschirmmodus 1:

Pixel Bits im Video-RAM-Byte

1	3	7
2	2	6
3	1	5
4	0	4

Bildschirmmodus 2:

Pixel Bits im Video-RAM-Byte

1	7
2	6
3	5
4	4
5	3
6	2
7	1
8	0

Beispiel:
Das linke Pixel eines Bytes soll im Modus 0 die Tintennummer 13 und das rechte Pixel die Tintennummer 7 erhalten. Dazu sind zuerst die Tintennummern binär darzustellen:

```
13 = 1101B
 7 = 0111B
```

Danach sind die Bits entsprechend der obigen Darstellung zu ordnen und das Byte ist zusammenzustellen:

```
linkes Pixel  : 1 x 1 x 0 x 1 x
rechtes Pixel : x 1 x 1 x 1 x 0
zusammen      : 1 1 1 1 0 1 1 0 = F6H
```

Es ist also der Wert F6H in das Byte zu speichern.

3.5. Der Druckertreiber

Wie aus /8/ ersichtlich, wird im KC compact als Drucker-Port eine Parallelschnittstelle (Centronics-Norm) verwendet. Dieser Port wird vom Betriebssystem mit einem entsprechenden Druckertreiber unterstützt. Diese Ausgabemöglichkeit kann auch von BASIC aus über Stream 8 (#8)genutzt werden. Als Besonderheit am KC compact ist zu beachten, daß standardmäßig nur 7 Bit breite Daten von den Betriebssystemroutinen ausgegeben werden. Damit können ASCII-Codes von 0 bis 7FH übertragen werden. Im Abschnitt 3.11. ist als Beispiel eine Treiberroutine aufgeführt, mit der acht Bit breite Daten (Codes von 0 bis FFH) übertragen werden können. Das ist insbesondere bei Grafikausgaben auf den Drucker vorteilhaft.

3.6. Die RSX-Kommandos

Allgemeines
Vom Betriebssystem wird eine Programmtechnik unterstützt, die es erlaubt, den Kommandovorrat des BASIC-Interpreters beliebig zu erweitern. Diese werden RSX-Programme genannt und sind entweder durch ROM-Erweiterungen oder nachgeladen im RAM ständig vorhanden (RSX =Resident System Extension).
Von BASIC aus werden diese Kommandos mit einem vorangestellten senkrechten Balken ""aufgerufen. Sie können wie ein normaler

BASIC-Befehl verwendet werden (in Programmen oder über Direktaufruf). Ein nachgeladenes Kreisprogramm könnte folgenden Syntax haben:

```
CIRCLE,x,y,r
```

Wie aus dem Beispiel ersichtlich ist,können den RSX-Kommandos Parameter (bis zu 32)übergeben werden.Ein weiterer Vorteil ist, daß der Programmierer die Einsprungadressen der RSX-Kommandos nicht kennen muß.
RSX-Kommandos können natürlich auch von anderen Maschinenprogrammen oder höheren Programmiersprachen aufgerufen werden. Dazu existiert ein Unterprogramm (BCD4H KL FIND COMAND), dem man den Kommandonamen übergibt. Wurde die Routine gefunden, bekommt man deren Adresse und ein ROM-Select-Byte zurück. Das ROM- Select-Byte wird dann benötigt, wenn die Routine in einem ROM liegt.

Aufbau einer RSX-Routine

Alle RSX-Pakete sind im Betriebssystem miteinander verkettet, so daß die Suche nach einem RSX erleichtert wird. Die Suche beginnt immer mit dem zuletzt angehängten RSX-Paket. Ein RSX- Paket hat im allgemeinen folgenden Aufbau:

- 4 Byte für RSX-Verkettung
- Tabelle der Einsprungvektoren
- Namenstabelle
- eigentliche Routinen

Die Tabelle der Einsprungvektoren hat folgenden Aufbau:

```
TABLE1:    DEFW TABLE2 ;Anfangsadresse der Namenstabelle
           JP   ROUT1  ;Sprung zur Routine 1
           JP   ROUT2  ;Sprung zur Routine 2
           ...
           JP   ROUTn  ;Sprung zur letzten Routine
```

Die Namenstabelle ist wie folgt aufzubauen:

```
TABLE2:    DEFB "R","O","U","T","1"+128 ;im letzten Buchstaben
           DEFB "R","O","U","T","2"+128 ;muß zur Trennung
           ...
           DEFB "R","O","U","T","n"+128 ;Bit 7 gesetzt werden
           DEFB 0                       ;Abschlußkennung
```

Zum besseren Verständnis folgt nun ein RSX-Beispielprogramm, das ein Umschalten der Schriftart bei Bildschirmausschriften ermöglicht. Verändert werden alle ASCII-Zeichenmatrizen, die im RAM liegen (SYMBOL AFTER).

```
;--------------------------------------
;RSX-Kommandos zur Zeichensatzänderung
;
;NORMAL -ursprünglicher Zeichensatz
;KURSIV -Kursivschrift
;FETT -Fettschrift
;SCHMAL -Schmalschrift
;
;Vereinbarungen:
KLLOGEXT:     EQU     0BCD1H      ;Erweiterung bekanntgeben
TXTGETT:      EQU     0BBAEH      ;Anfangsadresse selbstdefi-
                                  ;nierter Zeichenmatrizen
ROMON:        EQU     0B906H      ;Betriebssystem-ROM ein
ROMOFF:       EQU     0B909H      ;Betriebssystem-ROM aus
;
              ORG     0A000H      ;Startadresse
;
INIT:         LD      HL,KETTE    ;4 Bytes für RSX- Verkettung
              LD      BC,TABLE1   ;Vektortabelle
              JP      KLLOGEXT    ;RSX anmelden und Return
KETTE:        DEFS    4
;
TABLE1:       DEFW    TABLE2      ;Adresse der Namenstabelle
              JP      NORMAL
              JP      KURSIV
              JP      FETT
              JP      SCHMAL
;
TABLE2:       DEFM    "NORMA"
              DEFB    "L"+80H
              DEFM    "KURSI"
              DEFB    "V"+80H
              DEFM    "FET"
              DEFB    "T"+80H
              DEFM    "SCHMA"
              DEFB    "L"+80H
              DEFB    0
;
;Umschalt-Routine auf normalen Zeichensatz
;
NORMAL:       CALL    TXTGET      ;Matrixtabellenanfang
              RET     NC          ;keine Zeichen im RAM
              PUSH    HL          ;merke Anfangsadresse
              LD      HL,3800H    ;Anfangsadresse im ROM
              LD      B,0         ;BC:=erster ASCII-Code
              LD      C,A         ;im RAM
              SLA     C
              RL      B
              SLA     C
              RL      B
              SLA     C           ;BC:=BC*8 (=Anzahl nicht
              RL      B           ;zu kopierender Zeichenbytes)
              ADD     HL,BC       ;HL:=Adresse erstes Byte im
```

```
                ADD     HL,BC       ;HL:=Adresse erstes Byte im
                                    ;ROM
                LD      DE,4000H
                EX      DE,HL       ;DE:=HL,HL:=4000H
                SBC     HL,DE       ;HL:=Anzahl zu kop.Bytes
                EX      DE,HL       ;HL:=1.Byte im ROM
                PUSH    DE
                POP     BC          ;BC:=Anzahl zu kop.Bytes
                POP     DE          ;DE:=Anfangsadresse im RAM
                CALL    ROMON       ;ROM ein
                LDIR                ;Zeichensatz kopieren
                JP      ROMOFF      ;ROM aus und Return
;
;Umschalten auf Kursivschrift
;
KURSIV:         CALL    TXTGET      ;siehe
                RET     NC          ;oben
                LD      B,A         ;1.ASCII-Code im RAM
                XOR     A
                SUB     B           ;Anzahl Zeichen,die
                LD      B,A         ;geändert werden müssen
LOOP1: mal      SRL     (HL)        ;1.Pixelzeile 2
                SRL     (HL)        ;rechts rotieren
                INC     HL
                SRL     (HL)        ;2.Zeile 1 mal
                INC     HL
                SRL     (HL)        ;3.Zeile 1 mal
                INC     HL
                INC     HL          ;4.Zeile bleibt
                INC     HL          ;5.Zeile bleibt
                SLA     (HL)        ;6.Zeile 1 mal
                INC     HL          ;links rotieren
                SLA     (HL)        ;7.Zeile 1 mal
                INC     HL
                SLA     (HL)        ;8.Zeile 2 mal
                SLA     (HL)
                INC     HL
                DJNZ    LOOP1
                RET                 ;fertig
;
;Umschalten auf Fettschrift
;
FETT:           CALL    TXTGET      ;siehe
                RET     NC          ;oben
                NEG
                LD      C,A         ;Anzahl Zeichen
LOOP2:          LD      B,8         ;8 Bytes pro Zeichen
LOOP3:          LD      A,(HL)      ;Bytes 1 mal
                SRL     A           ;rechts rotieren und
                OR      (HL)        ;mit altem Wert
                LD      (HL),A      ;ODER verknüpfen
                INC     HL
                DJNZ    LOOP3
```

```
                DEC     C           ;letztes Zeichen?
                JR      NZ,LOOP2
                RET                 ;fertig
;
;Umschalten auf Schmalschrift
;
SCHMAL:         CALL    TXTGET      ;siehe
                RET     NC          ;oben
                NEG
                LD      C,A
LOOP4:          LD      B,8
LOOP5:          LD      A,(HL)      ;Bytes 1 mal
                SRL     A           ;rechts rotieren und
                AND     (HL)        ;mit altem Wert
                LD      (HL),A      ;AND verknüpfen
                INC     HL
                DJNZ    LOOP5
                DEC     C           ;letztes Zeichen?
                JR      NZ,LOOP4
                RET                 ;fertig
;
```

Nachdem das Programm als Maschinenprogramm (Binärfile) auf Kassette gerettet wurde, kann es bei anderen Programmen nachgeladen werden. Wurde die Anfangsadresse aus dem Listing verwendet (ORG 0A000H), dann wird die Routine mit CALL A000H initialisiert. Das heißt, sie wird dem Betriebssystem angemeldet. Danach stehen folgende Erweiterungen zur Verfügung:

```
NORMAL      = ursprünglicher Zeichensatz wird aus dem ROM in den
              RAM kopiert
FETT        = Umschalten auf Fettschrift
SCHMAL      = Umschalten auf Schmalschrift
KURSIV      = Umschalten auf Kursivschrift
```

Folgende kleine BASIC-Routine zeigt die Anwendung:

```
 10 SYMBOL AFTER 256:MEMORY &9FFF
 20 LOAD"textrsx.bin",&A000:CALL &A000
 30 SYMBOL AFTER 32:a$="abcdefghijk 1234567890"
 40 INK 0,0:INK 1,15:INK 2,21:MODE 1
 50 PEN 1:LOCATE 14,5:NORMAL:FETT:KURSIV
 60 PRINT "Textbeispiel":NORMAL
 70 PEN 2:LOCATE 4,9:PRINT "normal:"a$
 90 LOCATE 4,11:PRINT "fett :";:FETT
100 PRINT a$
100 NORMAL:LOCATE 4,13:PRINT "schmal:";:SCHMAL
110 PRINT a$
120 NORMAL:LOCATE 4,15:PRINT "kursiv:";:KURSIV
130 PRINT a$
140 NORMAL:SCHMAL:KURSIV:LOCATE 3,20
150 PRINT "Kombinationen ";:NORMAL:FETT:KURSIV
160 PRINT "sind ";:SCHMAL:PRINT "auch ";:NORMAL
170 FETT:SCHMAL:PRINT "erlaubt!"
180 GOTO 180
```

Parameterübergabe

Von BASIC aus können 32 Parameter an eine RSX-Routine übergeben werden. BASIC legt dazu nacheinander die Parameter auf den Stack und übergibt danach den Inhalt vom Stackpointer SP in das Indexregister IX. Weiterhin werden der RSX-Routine im Register A die Anzahl der übergebenen Werte mitgeteilt. Das Ablegen der Werte auf den Stack bedingt, daß nur 2-Byte-Integerzahlen verwendet werden können. Dadurch, daß sich der Stackpointer beim "Kellern" nach unten (Adresse wird kleiner) bewegt, werden die Parameter in der Reihenfolge vertauscht. Der erste Parameter liegt also an der obersten Adresse, und das IX-Register zeigt auf das unterste Byte des letzten Parameters:

```
(IX+00) = Low-Teil vom letzen Parameter
(IX+01) = High-Teil vom letzten Parameter
  ...
(IX+2*(n-1)) = Low-Teil vom ersten Parameter (n=Parameteranzahl)
(IX+2*n-1)   = High-Teil vom ersten Parameter
```

Eine Übernahme von zwei Parametern in einer RSX-Routine könnte z.B. wie folgt aussehen:

```
START:  CP  2          ;zwei Parameter ?
        RET NZ         ;nein
        LD  E,(IX+0)
        LD  D,(IX+1)   ;DE:=2.Parameter
        LD  L,(IX+2)
        LD  H,(IX+3)   ;HL:=1.Parameter
        ...
```

Neben der Möglichkeit nur Intgerzahlen zu übergeben, können in BASIC auch Fließkommazahlen und Strings benutzt werden. Das BASIC hält aber auch eine Möglichkeit bereit Fließkommazahlen und Strings zu übergeben. Wird nämlich vor eine Variable das geschrieben, dann wird die Adresse ermittelt, ab der die Variable im Speicher liegt. Bei Strings wird die Adresse des String-Descriptors ermittelt. In ihm stehen dann Länge und Adresse der Zeichenkette bereit (siehe Abschn. 4.3.). Vor Stringvariablen braucht bei einem RSX-Aufruf kein angegeben werden.
Folgendes Beispielprogramm verdeutlicht das:

```
10 a$="Test pr ogr amm"
20 b=12. 008
30 TEST, b, a$
...

TEST: CP     2
      RET    NZ
      LD     L,(IX+0)
      LD     H,(IX+1)
      LD     A,(HL)      ;A:=Länge des Strings
      INC    HL
      LD     E,(HL)
      INC    HL
      LD     H,(HL)
      LD     L,E         ;HL:=Stringanfangsadresse im RAM
      ...
```

In einer RSX-Routine dürfen bis auf IY alle Vordergrundregister verändert werden.

3.7. Interrupts

Der KC compact enthält nur eine einzige Interruptquelle, die in seiner zentralen Zustandssteuerung lokalisiert ist. Dazu werden die Impulse für den horizontalen Strahlrücklauf gezählt. Nach jeweils 52 Zeilen wird ein Interrupt ausgelöst. Da ein Bild insgesamt (auch die unsichtbaren Teile) 312 Zeilen enthält, treten während eines Bildes genau sechs Interrupts auf. Diese schnelle Interruptquelle (300 mal pro Sekunde) wird FAST TICKER genannt. Aus dem FAST TICKER werden softwaremäßig noch vier weitere Interruptquellen abgeleitet. Der Zeilenzähler wird so eingestellt, daß ein Interrupt genau nach dem zweiten Horizontalsynchronimpuls während des Vertikalsynchronimpulses angemeldet wird. Dieser Interrupt während des Strahlrücklaufes wird bevorzugt für Aktionen zur Veränderung des Bildinhaltes benutzt. Dadurch können flimmerfreie Bewegungen und Farbänderungen realisiert werden. Er wird FRAME FLY genannt. Ein weiterer (beliebiger) Interrupt während eines Bildes wird benutzt, um alle anderen Ereignisse zu bedienen. Da er nur 50 mal pro Sekunde auftritt, wird er TICKER genannt. Während des TICKER wird auch die Tastatur abgefragt. Wird dabei die gedrückte [ESC]-Taste erkannt, dann wird auf die Abarbeitung eines BREAK EVENTS verzweigt. Als letztes kann sich der Anwender in seinem Programm durch den Aufruf von BCF2H KL EVENT seine eigene Interruptquelle definieren. Alle diese Software-Unterbrechungen können benutzt werden, um beliebig viele einmalige oder periodische Ereignisse (Events)zu simulieren, die sofort oder erst nach einer Verzögerungszeit wirksam werden.
Alle Events werden über Datenblöcke gesteuert, die das Betriebssystem in eine seiner vielen Listen einreiht. Das sind:

1. die FAST TICKER CHAIN,
2. die FRAME FLY CHAIN,
3. die TICKER CHAIN und
4. eine Liste, die ausschließlich dem SOUND MANAGER zur Verfügung steht.

Durch das Einreihen eines Datenblocks in die entsprechende Liste wird festgelegt, durch welche Quelle ein Event angestoßen wird. Das Einreihen geschieht so, daß das Betriebssystem die Adresse des nächsten Blockes in der Liste in einem "Hangel-Pointer" ablegt. So kann es sich von einem Block zum nächsten "durchhangeln", obwohl diese über den gesamten zentralen RAM verteilt sein können. Diese Listen enthalten u.a. einen Event Block für jedes Event, der für die Reaktion auf den Anstoß zuständig ist. Mit ihm werden z.B. Prioritäten festgelegt, ob ein Event normal oder express ist oder ob ein Evenwt synchron oder asynchron ist. Da ein Interrupt zu einem unvorhergesehenen Zeitpunkt eintrifft, kann nicht garantiert werden, daß alle Systemressourcen zur Verfügung stehen. Wenn z.B. während der Abarbeitung eines Systemunterprogramms, das eigene Speichervariablen verändert, ein Interrupt das selbe Systemunterprogramm aufruft, kann es zu un

definierten Zuständen kommen. Deshalb wurden drei Arten von Events eingeführt, die zu unterschiedlichen Zeitpunkten abgearbeitet werden und unterschiedliche Systemressourcen nutzen können. Express Events werden in der Ausführung den normalen vorgezogen. Die Behandlung eines echten Interrupts (FAST TICKER) durch das Betriebssystem geschieht in zwei Schritten. Nach dem Sperren einer weiteren Interruptannahme und der Klassifizierung des eingetroffenen Interrupts werden die entsprechenden Listen durchsucht. Express-asynchrone Events werden sofort ausgeführt. Alle anderen, benötigten Eventblöcke werden in eine von zwei möglichen "abhängigen" Ketten, den Asynchronous und Synchronous Pending Queues eingereiht, wenn sie nicht bereits drin enthalten sind. Dann wird die Annahme neuer Interrupts zugelassen, die Asynchronous Pending Queue abgearbeitet und zurück zum laufenden Vordergrundprogramm gesprungen. Die Abarbeitung der Synchronous Pending Queue muß vom laufenden Vordergrundprogramm zu einem diesem genehmen Zeitpunkt vorgenommen werden.
Nach dem Aufruf von BCF2H KL EVENT wird die zugehörige Behandlungsroutine sofort angesprungen wenn ein asynchrones Event vorlag oder der Eventblock wird in die Synchronous Pending Queue eingereiht.
Für die praktische Anwendung ist es normalerweise unwichtig, ob ein Event sofort abgearbeitet wird oder in eine Pending Queue eingereiht wird, da der Nutzer davon nichts spürt. Anders verhält es sich, wenn man während der Interruptbearbeitung Änderungen z.B. am Eventblock vornehmen will, um z.B. einen einmaligen Vorgang, einen "one shot"zu programmieren.
Ein Express Asynchron Event wird noch, während die Interruptannahme verboten ist, abgearbeitet. Die Interruptannahme darf in der Eventbehandlungsroutine natürlich nicht zugelassen werden. Aus diesem Grund stehen die Restarts für diesen Eventtyp nicht zur Verfügung. Die Eventbehandlungsroutine sollte so kurz wie möglich sein und muß sich im zentralen RAM zwischen den Adressen 4000H und 0BFFFH befinden. Systemunterprogramme können i.allg. nicht genutzt werden.
Normale asynchrone Events werden dann ausgeführt, wenn die Interruptannahme erlaubt ist. Die Systemunterprogramme können aber nicht vollständig genutzt werden. Es muß garantiert werden, daß die Speichervariablen für den aktuellen Zustand des Vordergrundprogrammes erhalten bleiben.
Synchrone Events werden nach ihrem Anstoßen durch einen Interrupt erst dann ausgeführt, wenn sie vom Vordergrundprogramm dazu aufgefordert werden. Das geschieht am besten an einer Stelle im Vordergrundprogramm, wo alle Systemressourcen ohne Bedenken genutzt werden können. Das ist auch der Sinn für die Einführung synchroner Events. BASIC benutzt für seine Interrupts ausschließlich synchrone Events. Dabei wird immer zwischen zwei BASIC-Statements nachgesehen, ob Events auf ihre Ausführung warten, wenn ja, werden sie gestartet. Synchronen Events wird eine Priorität,die im Bereich von 1 bis 15 liegen kann, zugeordnet. Das laufende Vordergrundprogramm hat die Priorität 0. Wird die Synchronous Pending Queue durch die Routine KL POLL SYNCRONOUS abgefragt (polling), dann werden nur Events höherer Priorität

beachtet. Das sind beim Polling im Vordergrundprogramm alle Events, in einer Eventbehandlungsroutine aber nur die Events mit höherer Priorität. Außerdem sind alle express-synchronen Events dringender als alle normalen. Mit den Vektoren BD04H KL EVENT DISABLE und BD07H KL EVENT ENABLE kann die Behandlung von normalen synchronen Events verboten und wieder zugelassen werden. Das entspricht den BASIC-Befehlen DI und EI. Alle Unterbrechungen in BASIC sind normal synchron, nur "ON BREAK GOSUB"ist express synchron.
Im weiteren wird gezeigt, wie man den Eventmechanismus in eigenen Programmen ausnutzen kann. Zunächst muß das Programm für das Event vorliegen. Je nach gewünschter Interruptquelle wird ein Block für die entsprechende verkettete Liste bereitgestellt und eingebunden. Er hat folgendes Aussehen:

1. Fast Ticker Block
   Byte 0,1 Platz für den Hangel-Pointer in der Fast Ticker Chain
   Byte 2,ff der Event-Block
2. Frame Flyback Block
   Byte 0,1 Platz für den Hangel-Pointer in der Frame Flyback Chain
   Byte 2,ff der Event-Block
3. Ticker Block
   Byte 0,1 Platz für den Hangel-Pointer in der Ticker Chain
   Byte 2,3 count down Zähler für die Ticker Zahl bis zum nächsten simulierten Ereignis
   Byte 4,5 Nachladewert für Byte 2,3,wenn diese 0 werden (steht hier 0,dann wird nur ein einmaliges Ereignis erzeugt).
   Byte 6,ff der Event-Block

Man erkennt schon am Aufbau der Blöcke,daß mit dem Ticker Block die langperiodischen oder einmaligen und die erst nach einer bestimmten Verzögerungszeit wirksamen Ereignisse bequem programmiert werden können.
Diese Blöcke werden mit Hilfe der Routinen BCE3H KL ADD FAST TICKER, BCDAH KL ADD FRAME FLY und BCE9H KL ADD TICKER in die entsprechende Liste eingereiht.

Der Event Block hat folgendes Aussehen:

Byte 0,1 Platz für den Hangel-Pointer in der Pending Queue
Byte 2 Zähler und Steuerbyte
Byte 3 Typ-Byte im Event Block
Byte 4,5 Adresse der Eventbehandlungsroutine
Byte 6 ROM Nummer (nur bei Bedarf)
Byte 7,ff lokale Variable der Behandlungsroutine (nur bei Bedarf)

Ein solcher Event Block wird, beim Anstoßen (kicken) eines Ereignisses in eine der beiden Pending Queues eingereiht, bis auf die Fälle, in denen die Eventbehandlungsroutine sofort ausgeführt wird. Gleichzeitig wird das Zähl- und Steuerbyte erhöht.

Dadurch können kurzzeitig mehr Anforderungen dieses Events eingehen, als die CPU abarbeiten kann. Die Anzahl der noch ausstehenden Kicks wird in diesem Byte gespeichert. Wird diese Zahl größer als 127 bzw. negativ, dann wird dieser Event Block ruhiggestellt und kann nicht mehr angestoßen werden. Das Betriebssystem benutzt zum Ruhigstellen eines Events immer den Wert C0H.

Das Type-Byte im Event Block hat folgenden Aufbau:

```
Bit 0 =0  Es wird eine ROM-Nummer erwartet (far adress).
      =1  Keine ROM-Nummer nötig (near adress).
          Die Eventbehandlungsroutine liegt im Betriebssystem-ROM
          oder im zentralen RAM von 4000H bis C000H.
Bit 1-4   Priorität (nur bei synchronen Events von Bedeutung)
Bit 5     muß auf 0 gesetzt sein
Bit 6 =0  normales Event
      =1  express Event
Bit 7 =0  synchrones Event
      =1  asynchrones Event
```

Es folgt ein Programmbeispiel zum Einreihen eines express-asynchronen Events in die FRAME FLYBACK CHAIN. Die Eventbehandlungsroutine wird dann mit jedem Strahlrücklauf ausgeführt.

```
          LD   HL,FRBLOCK
          CALL 0BCDAH    ;KL ADD FRAME FLY
     ...
;
;Der FRAME FLYBACK Block
;
FRBLOCK: DEFW 00        ;Platz für den Hangel-Pointer in der
                        ;FRAME FLYBACK Chain
         DEFW 00        ;Platz für Hangel-Pointer im Pending Queue
         DEFB 00        ;Zählerbyte auf 0 gesetzt.
         DEFB 11000001B ;asynchron, express, near adress
         DEFW ADRESSE
     ...
ADRESSE:                ;Beginn der Eventbehandlungsroutine.
```

Das nächste Beispielprogramm zeigt das Polling und den Aufruf einer synchronen Eventroutine.

```
     ...
;an passender Stelle im Vordergrundprogramm
;Nachfragen, ob etwas in der Synchron Pending Queue ist (pollen)
          CALL 0B921H   ;KL POLL SYNCHRONOUS
          CALL C,SYNLOOP;wenn CY=1, liegt etwas vor, behandeln
     ...
;
;Behandlungsroutine für synchrone Events
;
SYNLOOP: CALL 0BCFBH    ;KL NEXT SYNC,nächstes Event aus
                        ;Synchronous Pending Queue holen
         RET  NC        ;wenn CY=0, liegt nichts mehr vor, fertig
         PUSH AF        ;alte Priorität retten
```

```
PUSH HL        ;Zeiger auf den Eventblock retten
CALL 0BCFEH    ;KL DO SYNC, ausführen der Eventbe-
               ;handlungsroutine
POP  HL        ;Zeiger auf den Eventblock zurückholen
POP  AF        ;alte Priorität zurückholen
CALL 0BD01H    ;KL DONE SYNC, Zählerbyte zurückstellen
               ;und Priorität restaurieren
JR   SYNLOOP   ;Schleife
```

Die Eventbehandlungsroutine darf alle Vordergrund-Register außer IX und IY verändern. Das Betriebssystem übergibt an die Eventbehandlungsroutine eine Adresse im Eventblock.

Für die Lage der Behandlungsroutine gilt:
im RAM:
Eingabe: DE: zeigt auf Byte 5 des Eventblocks. Ab DE+2 befinden sich die lokalen Variablen.
im ROM:
Eingabe: HL: zeigt auf Byte 4 des Eventblocks. Ab HL+3 befinden sich die lokalen Variablen.

Weit schwieriger als die Initialisierung eines (regelmäßigen) Ereignisses gestaltet sich das Abstellen. In der Ticker Chain kann man zwar durch das Beschreiben des Zählbytes im Eventblock einmalige Ereignisse programmieren, eine Bearbeitung der Ticker Chain führt aber auch weiterhin zu geringen Zeitverlusten, da sich das Betriebssystem ständig an diesem Block "vorbeihangeln" muß.
Mit den Programmen BCDDH KL DEL FRAME FLY, BCE6H KL DEL FAST TICKER und BCECH KL DEL TICKER kann man Blöcke wieder vollständig aus den Event-verursachenden Listen "aushängen", auch durch die Eventbehandlungsroutine (außer bei einem express asynchronen Event, das die Systemresourcen nicht nutzen darf). Danach werden die entfernten Eventblöcke zwar nicht mehr in die Pending Queue gekickt, es kann aber sein, daß ein solches Event in einer Pending Queue noch auf seine Ausführung wartet. Es wird dann trotzdem noch ausgeführt. Das kann bei synchronen Events immer der Fall sein, bei asynchronen Events nur, wenn die Eventbehandlungsroutine sich selbst aushängen will. Sollen auch noch die ausstehenden Kicks ignoriert werden, muß man die Eventblocks selbst ruhigstellen. Express asynchrone Events werden durch das Setzen des Zähl-und Steuerbytes im Eventblock auf einen negativen Wert ruhiggestellt. Bei normalen asynchronen Events kann das durch die Routine BD0AH KL DISARM EVENT geschehen. Synchrone Events werden durch die Routine BCF8H KL DEL SYNCHRONOUS ruhiggestellt. Diese Routine setzt das Zähl- und Steuerbyte des Eventblockes auf einen negativen Wert und hängt den Eventblock aus der Synchronous Pending Queue aus, falls er noch drin ist.
Soll ein Eventblock erneut initialisiert werden, muß er abgeschaltet und aus seiner Pending Queue entfernt worden sein. Das ist bei asynchronen Events aber nur im Vordergrundprogramm mit Sicherheit möglich. Bei synchronen Events kann das durch das

Vordergrundprogramm aber auch durch die Eventbehandlungsroutine erfolgen, indem KL DEL SYNCHRONOUS aufgerufen wird.
Soll der Eventblock selbst verändert werden, muß sichergestellt sein, daß er in der Zwischenzeit nicht gekickt werden kann. Dazu hängt man den Event-verursachenden Block aus seiner Chain oder verbietet für diese Zeit den Interrupt.
Im folgenden wird die Behandlung weiterer Interruptquellen beschrieben. Dem Soundmanager muß man mit dem Vektor BB45H SOUND ARM EVENT einen Eventblock übergeben. Der BREAK-Mechanismus des Key-Managers benutzt einen festen Eventblock, in den man mit der Routine BB45H KM ARM EVENT die far adress der Behandlungsroutine eintragen kann. Eigene Eventquellen lassen sich mit BCF2H KL EVENT konstruieren. Diese Routine läßt keine Interrupts zu, wenn sie beim Aufruf verboten waren. Dazu ist aber die Adresse der Routine aus dem Vektor zu lösen, da dieser ja über einen Restart funktioniert, und in einen eigenen Aufruf einzubauen. Will man an den KC compact eine externe Interruptquelle über das Expansionsinterface anschließen, muß man dafür sorgen, daß sich diese Interruptquelle von der internen unterscheidet. Dazu darf die externe Interruptquelle ihre Interruptanforderung nicht bei der Interruptquittierung zurücknehmen, sondern erst auf ausdrückliche Anweisung ihrer Behandlungsroutine. Erkennt das Betriebssystem, daß ein externer Interrupt vorliegt, ruft es den Vektor auf der Adresse 003BH auf. Für jede interrupterzeugende Systemerweiterung muß an dieser Stelle eine Behandlungsroutine eingetragen werden, wobei gleichzeitig eine Kopie des alten Eintrages angelegt werden muß, die angesprungen werden kann, wenn man feststellt, daß der Interrupt doch nicht für die eigene Routine war. Liegt ein externer Interrupt vor, dann wird das dafür vorgesehene Event mit KL EVENT aktiviert. Im folgenden Beispielprogramm wird gezeigt, wie das ohne Restarts geschehen kann.

```
;Initialisierung der Kick-Routine für Eventroutine für externes
;Interrupt
;
INIT:    LD   HL,(OBCF2H+1) ;Adresse aus KL EVENT holen
         RES  7,H           ;ROM select Bits für Restart löschen
         RES  6,H
         LD   (KICK+1),HL   ;und in den eigenen Aufruf kopieren
;
;alten Eintrag am External Interrupt Entry retten
;
         LD   HL,3BH
         LD   DE,KOPIE
         LD   BC,5
         LDIR
;
;und durch Sprung zur eigenen Interruptroutine ersetzen.
;
        LD   HL,ERSATZ
        LD   DE,3BH
        LD   BC,3
        LDIR
;
;hier der externen Hardware Generieren des Interrupts erlauben
;
```

```
      ...
ERSATZ: JP    EXT       ;Patch für den External Interrupt
KOPIE:  DEFS 5
      ...
;
;KICKROUTINE FÜR EXTERNAL INTERRUPTS
;
EXT:    EX    AF,AF'    ;aktueller ROM-Status und Bildschirmmodus
        LD    C,A       ;ist in A', nach C laden.
        EX    AF,AF'
        LD    B,7Fh     ;ROM auf Adresse 0 einschalten
        RES   2,C
        OUT   (C),C
;
;Hier testen, ob Interrupt von der eigenen Hardwareerweiterung
;kommt!
;
        JP    NZ,KOPIE  ;wenn Interrupt nicht von der eigenen
                        ;Hardware stammt, dann Kopie anspringen!
        LD    HL,EVBLOCK ;sonst Zeiger auf den Eventblock zur
                        ;Reaktion auf externes Interrupt laden
KICK:   JP    0000      ;und KL EVENT direkt anspringen, d.h.
                        ;Eventblock kicken. Die Adresse nach JP
                        ;wird bei der Initialisierung der Kick-
                        ;routine aktualisiert.
```

3.8. Restarts

Im KC compact liegen teilweise ROM und RAM in gleichen Adreßbereichen, d.h., die Speicherbänke liegen parallel. Um den komplizierten Prozeß der Bankverwaltung bzw. -umschaltung zu vereinfachen, werden dazu vom Betriebssystem die notwendigen Routinen angeboten. Soll z.B. aus dem unteren RAM-Bereich eine Routine des Betriebssystems aufgerufen werden, müssen von einer Schaltroutine im mittleren oder oberen RAM-Bereich das Betriebssystem zugeschaltet, die Routine aufgerufen, das Betriebssystem abgeschaltet und der Rücksprung zum Hauptprogramm übernommen werden. Die Handhabung dieser Betriebssystem-Routinen ist einfach. Verwendet werden dazu die Restart-Befehle des U880. Ein Problem stellt dabei die Parameterübergabe dar, denn es müssen ja die gewünschte Speicherkonfiguration und die Unterprogrammadresse übergeben werden. Aus dem 1-Byte-Befehl (Restart) wird ein 3-Byte-Befehl (wie JUMP und CALL), indem unmittelbar nach dem Restart die Unterprogrammadresse anzugeben ist. Ein Assemblerquelltext kann dann z.B. so aussehen:

```
      ...
      RST 8
      DEFW adresse
      ...
```

Die Übergabe der Speicherkonfiguration hängt vom verwendeten Restart und damit davon ab, in welcher Speicherbank das gewünschte Unterprogramm liegt. Wird zum Beispiel eine Betriebssystem-Routine gewünscht, wird die Konfiguration in den Bits 14 und 15 übergeben. Diese werden für Adressen von 0 bis 3FFFH nicht ge-

braucht. Die Handhabung der verschiedenen Restartbefehle ist dem Abschnitt 3.9.3. zu entnehmen.
Da die Restarts ähnlich wie JUMP-und CALL-Befehle verwendet werden, dürfen die Registerinhalte nicht verändert werden. Alle notwendigen Berechnungen werden deshalb vom Zweitregistersatz ausgeführt. Da diese jedoch auch für den Interrupt-Mechanismus eingesetzt werden, wird für die Zeit der Zweitregisterbenutzung der Interrupt gesperrt und nach dem Rücktausch wieder zugelassen. Es ist also zu beachten, daß alle Restarts den Interrupt wieder zulassen!
Da die Restarts auch von ROM-Routinen (Betriebssystem)genutzt werden, sind alle Restart-Routinen sowohl im Betriebssystem-ROM als auch im RAM vorhanden.
Vor der Rückkehr zum Hauptprogramm wird die ursprüngliche Speicherkonfiguration wieder hergestellt.

3.9. Sprungleisten

siehe Teil 2 der Systembeschreibung

KLEINCOMPUTER

KC compact

Systemhandbuch Teil2

veb mikroelektronik >wilhelm pieck<
mühlhausen
im veb kombinat mikroelektronik

3.9. Sprungleisten

Alle Unterprogramme des Betriebssystems und des BASIC-Interpreters, sofern deren Nutzung von außerhalb sinnvoll ist, werden dem Anwender über Sprungleisten zur Nutzung angeboten. Das Betriebssystem und der BASIC-Interpreter nutzen selbst diese Sprungleisten. Das System wird dadurch sehr transparent und leicht veränderbar (siehe Abschn. 3.11.). Die Sprungleisten setzen sich aus den einzelnen Vektoren zu jedem Unterprogramm zusammen. In der Regel besteht ein Vektor aus einem Restart-Befehl und der eigentlichen, absoluten Unterprogrammadresse (siehe Abschn.3.8.).
Im RAM sind zwei Sprungleistenbereiche vorhanden,die untere und die obere (zentrale) Sprungleiste. Die zentrale Sprungleiste ist noch einmal so unterteilt, daß thematisch gleiche Unterprogramme zusammengefaßt sind, z.B. die Tastatur-Programme, die Grafik-Programme oder die BASIC-Routinen.
Nachfolgend werden die Unterprogramme (UP's), deren Vektoradressen und zu übergebende Parameter (Register) bzw. vom Unterprogramm zurückgegebene Parameter (Register) beschrieben. Es gelten folgende Abkürzungen:

PE: ... Parameterübergabe an das UP
PA: ... Parameterübernahme vom UP
UR: ... unveränderte Register

Alle UP-Beschreibungen haben folgende Kopfzeile:

Vektoradresse Name des UP Kurzcharakteristik

3.9.1. Zentrale Sprungleiste

3.9.1.1. Die Tastatur-Routinen - KEY MANAGER (KM)

BB00 KM INITIALISE Tastaturroutinen-Initialisierung

PE: keine
PA: keine
UR: IX,IY

Alle Variablen und Puffer des Key-Managers werden initialisiert.
Das betrifft: IND KM TEST BREAK
Tastenpuffer
Expansions-Puffer und -Strings
Tasten-Übersetzungstabellen
Repeat-Verzögerung und -Geschwindigkeit
SHIFT und CAPS LOCK
BREAKs werden ignoriert

BB03 KM RESET Rücksetzen der Tastatur-Routine

PE: keine
PA: keine
UR: IX,IY

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=Tastennummer
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=Tastennummer
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=Tastennummer
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=Tastennummer
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 unbemerkt 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 ausgeben
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 Benutzerebene

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 Benutzerebene

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 Betriebssystem-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 Textfensters 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=Tintennummer
PA: keine
UR: BC,DE,IX,IY

Die Tintennummer 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 Zeichenbildmatrixadresse 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 (Anwenderebene)
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

BBC0 GRA MOVE ABSOLUTE Festlegen der Grafikcursor-Position

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

Die Koordinaten werden relativ zum Ursprung (Origin) angegeben und sind vorzeichenbehaftet (-32768 bis +32767).

BBC3 GRA MOVE RELATIVE Cursor relativ bewegen

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

Die Koordinaten werden relativ zur aktuellen Grafikcursor-Position angegeben und sind vorzeichenbehaftet (-32768 bis +32767).

BBC6 GRA ASK CURSOR Ermitteln der Grafikcursor-Position

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

Die Koordinaten werden relativ zum Ursprung (Origin) angegeben und sind vorzeichenbehaftet (-32768 bis +32767).

BBC9 GRA SET ORIGIN Festlegen des Koordinatenursprungs (ORIGIN)

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

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

BBCC GRA GET ORIGIN Ermitteln des Koordinatenursprungs

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

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 Grafikfenstergrenze 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 Bildwiederholspeicher) gerundet.

BBD2 GRA WIN HIGHT obere und untere Grafikfenstergrenze 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 Grafik-vordergrund-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 Vordergrund-modus 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 Bildschirmecke

PE: DE=X-Koordinate relativ zum Ursprung
HL=Y-Koordinate relativ zum Ursprung
PA: DE=X-Koordinate relativ zur Bildschirmecke
HL=Y-Koordinate relativ zur Bildschirmecke
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 Bildwiederholspeicher wird auf C000H eingestellt
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 BASE Bildwiederholspeicher in ein anderes 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 Sroll-Offset des alten Bildschirms wird mit übernommen.

BC0B SCR GET LOCATION Ermitteln des Bildwiederholspeicherviertels 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 Grafikcursor und der Ursprung (Origin) in die linke untere Ecke gestellt.

BC11 SCR GET MODE Ermitteln des eingestellten Bildschirmmodus

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 Zeilenanzahl 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 Bildwiederholspeicheradresse aus einer Zeichen-Position

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

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

BC1D SCR DOT POSITION Ermitteln der Bildwiederholspeicheradresse 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 Bildwiederholspeicher

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

BC23 SCR PREV BYTE Ermitteln der Adresse links neben der angegebenen Adresse im Bildwiederholspeicher

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

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

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

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

PE: HL=Bildwiederholspeicheradresse
PA: HL=Bildwiederholspeicheradresse 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=Tintennummer
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 Byte-positionen)

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

Bildschirmausschnitte (Fenster) werden immer softwaremäßig gescrollt.

BC53 SCR UNPACK Expandieren einer Zeichenmatrix entsprechend dem Bildschirmmodus

PE: HL=Anfangsadresse der Zeichenmatrix
DE=Pufferanfang
PA: Bytes im Puffer
UR: IX,IY

Je nach Bildschirmmodus wird die Zeichenmatrix auf 8, 16 oder 32 Bytes expandiert. Dementsprechend muß die Puffergröße 8, 16 oder 32 Bytes betragen.

In den enstandenen Bytes sind alle Bits der zu setzenden Pixel gesetzt. Diese Bytemasken können mit Farb-Bytes (BC2C SCR INK ENCODE) verknüpft werden, um den Vordergrund- und Hintergrund-Pixeln entsprechende Tinten zu geben.

BC56 SCR REPACK Komprimieren von expandierten Bytes zu einer Zeichematrix

PE: A =Farb-Byte
H =Zeichen-Spalte
L =Zeichen-Zeile
DE=Pufferanfang (8 Bytes)
PA: Matrix im Puffer
UR: IX,IY

In der Zeichenmatrix werden alle Bits der entsprechenden mit dem Farb-Byte gesetzten Pixel gesetzt.

BC59 SCR ACCESS Vordergrundmodus für Grafikausgaben setzen

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

Mit dem Modus wird festgelegt, wie die Vordergrund-Tinte zu setzender Pixel mit deren alter Tinte verknüpft werden soll. Es gelten folgende Festlegungen:

A=0 : deckend neue Tinte =Vordergrund-Tinte
A=1 : XOR neue Tinte =alte Tinte XOR Vordergrund-Tinte
A=2 : AND neue Tinte =alte Tinte AND Vordergrund-Tinte
A=3 : OR neue Tinte =alte Tinte OR Vordergrund-Tinte

BC5C SCR PIXELS Setzen eines Punktes

PE: B=Farb-Byte (BC2C SCR INK ENCODE)
C=Masken-Byte für das (die)Pixel
HL=Bildwiederholspeicheradresse
PA: keine
UR: BC,DE,HL,IX,IY

BC5F SCR HORIZONTAL waagerechte Linie zeichnen

PE: A =Farb-Byte (BC2C SCR INK ENCODE)
DE=linke X-Koordinate
BC=rechte X-Koordinate
HL=Y-Koordinate
PA: keine
UR: IX,IY

Bezugspunkt für die Koordinaten ist die linke untere Bildschirmecke (0,0).

BC62 SCR VERTICAL senkrechte Linie zeichnen

PE: A =Farb-Byte (siehe BC2C SCR INK ENCODE)
DE=X-Koordinate
HL=untere Y-Koordinate
BC=obere Y-Koordinate
PA: keine
UR: IX,IY

Bezugspunkt für die Koordinaten ist die linke untere Bildschirmecke (0,0).

BD55 SCR SET POSITION Festlegen der Bildwiederholspeicheranfangsadresse für die Software (verdeckte Bildschirmausgaben)

PE: HL=Scroll-Offset
A =Bildschirmbasis
PA: A und HL legalisiert
UR: BC,DE,IX,IY

Der alte Bildschirm wird weiter angezeigt, nur die Bildschirmausgaben gehen an den neuen Bildwiederholspeicher. Gültige Werte für A sind 0, 4, 8 und CH.

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.

BC89 CAS TEST EOF Abfrage, ob Ende der Eingabedatei erreicht ist

PE: keine
PA: CY=0 Fehler (siehe Abschn. 3.3.)
CY=1 Ende noch nicht erreicht
UR: BC,DE,HL,IY

Diese Routine ist beim zeichenweisen Lesen sinnvoll.

BC8C CAS OUT OPEN Ausgabedatei eröffnen

PE: B =Länge des Dateinamens
HL=Adresse des Dateinamens im RAM
DE=Adresse eines 2-KByte-Ausgabe-Puffers
PA: CY=0 Fehler (siehe Abschn. 3.3.)
UR: IY

Nur zur zeichenweisen Ausgabe wird der Ausgabe-Puffer benutzt. Siehe BC77 CAS IN OPEN.

BC8F CAS OUT CLOSE Ausgabedatei schließen

PE: keine
PA: CY=0 Fehler (siehe Abschn. 3.3.)
CY=1 kein Fehler
UR: IY

Alle Ausgabedateien müssen geschlossen werden. Auch die mit BC98 CAS OUT DIRECT geschriebenen.

BC92 CAS OUT ABANDON Ausgabedatei vergessen

PE: keine
PA: keine
UR: IX,IY

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

BC95 CAS OUT CHAR ein Zeichen in die Ausgabedatei schreiben

PE: A =Zeichencode
PA: CY=0 Fehler (siehe Abschn. 3.3.)
CY=1 kein Fehler
UR: BC,DE,HL,IY

BC98 CAS OUT DIRECT Ausgabedatei mit einem Mal schreiben

PE: A =Dateityp
BC=Einsprungadresse (Selbststartadresse)
DE=Länge des abzuspeichernden Bereiches
HL=Anfangsadresse
PA: CY=0 Fehler (siehe Abschn.3.3.)
CY=1 kein Fehler
UR: IY

Diese Routine darf nach der Eröffnung nur einmal aufgerufen werden und die Ausgabedatei muß mit BC8F CAS OUT CLOSE geschlossen werden. Zeichenweise Ausgabe und Ausgabe in einem Zuge dürfen nicht gemischt verwendet werden.

BC9B CAS CATALOG Inhaltsverzeichnis erstellen

PE: DE=Adresse eines 2KByte-Puffers
PA: CY,Z und A enthalten Informationen über Erfolg oder Fehler
UR: DE,IY

Bei Aufruf dieser Routine darf keine Eingabedatei eröffnet sein. Die Kassettenmeldungen auf dem Bildschirm werden durch diese Routine zugelassen. Da ja nicht ein Kassettenende getestet werden kann, muß diese Routine mit [ESC] unterbrochen werden. Es wird deshalb bei einem Break kein Fehler vermerkt (CY=1).

Ein Fehler (CY=0, Fehlernummer 14) wird vermerkt, wenn eine Eingabedatei eröffnet war. Alle anderen Fehler (siehe Abschn. 3.3.) werden im Inhaltsverzeichnis vermerkt.

BC9E CAS WRITE Speicherbereich auf Kassette ausgeben

PE: HL=Anfangsadresse des Datenblocks
DE=Länge des Blocks
A =Synchronisationszeichen
PA: CY=0 Fehler, A enthält Fehlercode (siehe Abschn. 3.3.)
CY=1 kein Fehler
UR: IY

Mit dieser Routine können beliebig lange Blöcke zusammenhängend auf Band gespeichert werden. Die Blocklänge 0 wird dabei als 65536 interpretiert.
mögliche Synchronisationszeichen:
2CH -Header
16H -Daten

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 Schrittanzahl (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-Vordergrund-programm, wird bei einem Fehlerfall immer zum BASIC (Kaltstart des Systems) zurückgekehrt.

BCCB KL ROM WALK Initialisierung aller Hinter-grund-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 Vorder-grund-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 Hinter-grund-ROMs

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

PA: DE=Adresse des ersten Bytes des freien Speicherbereichs
HL=Adresse des letzten Bytes (inkl.)
UR: C,IX,IY

Siehe BCCB KL ROM WALK.

BCD1 KL LOG EXT RSX-Erweiterung anmelden

PE: BC=Anfangsadresse der Kommandotabelle
HL=Anfangsadresse eines 4 Bytes-Puffer für Kernel
PA: keine
UR: AF,BC,HL,IX,IY

Siehe Abschnitt 3.6.

BCD4 KL FIND COMAND Startadresse eines RSX-Kommandos ermitteln

PE: HL=Anfangsadresse des Kommandonamen
PA: CY=0 Kommando wurde nicht gefunden
CY=1 Kommando gefunden, HL=Adresse und C=ROM-Selekt-Byte
UR: IX,IY

Es werden alle initialisierten RSX-Kommandotabellen durchsucht. Das Durchsuchen beginnt in der zuletzt angemeldeten Tabelle und so weiter (zu beachten bei gleichnamigen Kommandos). Wurde das Kommando in einem Vordergrund-ROM gefunden, wird ein Kaltstart dieses Programms durchgeführt und BCD4 KL FIND COMAND kehrt nicht zurück.

BCD7 KL NEW FRAME FLY Initialisierung eines Datenblocks für die Frame-Flyback-Chain

PE: HL=Anfangsadresse des Frame-Flyback-Blocks
B =Event-Klasse (siehe Abschn. 3.7.)
C =ROM-Selekt-Byte für die EVENT-Routine
DE=Adresse der Event-Routine
PA: keine
UR: BC,IX,IY

Der Block (HL)wird entsprechend B, C und DE zu einem Frame- Flyback-Datenblock initialsiert. Der Kick-Counter wird auf 0 gestellt, und der Datenblock wird in die Frame-Flyback-Chain eingehängt.

BCDA KL ADD FRAME FLY Einfügen eines Frame-Flyback-Datenblocks in die Frame-Flyback-Chain

PE: HL=Anfangsadresse des Frame-Flyback-Blocks
PA: keine

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)

PA: keine
UR: BC,IX,IY

Siehe Abschnitt 3.7.

BCEC KL DEL TICKER Entfernen eines Ticker Datenblocks aus der Chain

PE: HL=Anfangsadresse des Ticker Blocks
PA: CY=1 Block war in der Chain, DE=Rest des Count Down
CY=0 Block war nicht in der Chain
UR: BC,IX,IY

Siehe BCDD KL DEL FRAME FLY.

BCEF KL INIT EVENT Initialisierung eines Event-Blocks

PE: HL = Anfangsadresse des Event-Blocks
B = Type-Byte
C = ROM-Select-Byte für die Event-Routine
DE = Adresse der Event-Routine
PA: HL zeigt hinter den Event-Block, wurde also um 7 erhöht
UR: AF,BC,DE,IX,IY

Entsprechend B, C und DE wird der Event-Block initialisiert.
Siehe auch Abschnitt 3.7.

BCF2 KL EVENT Anstoßen eines Event-Blocks

PE: HL=Anfangsadresse des Event-Blocks
PA: keine
UR: IX,IY

Diese Routine ist zum Aufrufen aus der Interruptbehandlungsroutine vorgesehen.

BCF5 KL SYNC RESET Löschen der Liste aller synchronen Events, die noch auf ihre Abarbeitung warten

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

Der Zähler (Byte 2 des Event-Blocks) wird nicht auf 0 gestellt. Wird der Eventblock danach angestoßen, wird er nicht in die nueue eingehängt. Der Kernel nimmt an, daß er sich bereits darin befindet und erhöht nur den Zähler, d.h., der Event-Block gelangt nicht mehr zur Abarbeitung (ist ruhiggestellt). Der Zähler muß vom Anwenderprogramm auf 0 gestellt werden (siehe Abschn. 3.7.).

BCF8 KL DEL SYNCHRONOUS Löschen eines synchronen Events aus der Synchronous Pending Queue

PE: HL=Anfangsadresse des Event-Block
PA: keine
UR: IX,IY

Soll ein synchroner Event abgestellt werden, muß er erst aus der jeweiligen Chain und danach mit BCF8 KL DEL SYNCHRONOUS aus der Synchronous Pending Queue entfernt werden, falls er bereits gekickt wurde (siehe Abschn.3.7.).

BCFB KL NEXT SYNC Nächstes synchrones Event aus der Synchronous Pending Queue holen

PE: keine
PA: CY=0 Synchronous Pending Queue ist leer
CY=1 Block war vorhanden,HL=Anfangsadresse des Event-Blocks
und A=alte Event-Priorität
UR: BC,IX,IY

Wurde ein Block in der Queue gefunden,wird er daraus entfernt. Mit den erhaltenen Werten kann mit BCFE KL DO SYNC die Event-Routine ausgeführt werden. Danach ist BD01 KL DONE SYNC aufzurufen (benötigt auch die Werte von BCFB KL NEXT SYNC). Diese Routine dekrementiert den Zähler im Event-Block und hängt den Eventblock wieder in die Synchronous Pending Queue, falls der Zähler noch nicht 0 erreicht hat.

BCFE KL DO SYNC Synchrones Event ausführen

PE: HL=Anfangsadresse des Event-Blocks
PA: keine
UR: IX,IY

Siehe BCFB KL NEXT SYNC.

BD01 KL DONE SYNC Abschluß der Abarbeitung eines synchronen Events

PE: HL=Anfangsadresse des Event-Blocks
A =alte Event-Priorität
PA: keine
UR: IX,IY

Siehe BCFB KL NEXT SYNC.

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.

BD0A 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 Bildwiederholspeicher (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 Soundcontroler-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 Drukker

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=Tintennummer 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.

BDE5 IND SCR READ Ermitteln der Tinte eines Punktes

PE: HL=Adresse im Bildwiederholspeicher
C =Pixel-Maske
PA: A =Tintennummer des Pixels
UR: BC,DE,HL,IX,IY

BDE8 IND SCR WRITE Setzen eines oder mehrerer Punkte

PE: HL=Adresse im Bildwiederholspeicher
C =Maske für das oder die Pixel
B =Farb-Byte
PA: keine
UR: BC,DE,HL,IX,IY

Bei dieser Routine wird der eingestellte Vordergrund-Modus beachtet.

BDEB IND SCR MODE CLEAR Bildschirm mit Tinte 0 löschen

PE: keine
PA: keine
UR: IX,IY

BDEE IND KM TEST BREAK Break-Test

PE: C=Status von SHIFT und CTRL
PA: keine
UR: BC,DE,IX,IY

Es gelten folgende Festlegungen für das Register C:
Bit 5 = 1 - SHIFT ist gedrückt
Bit 7 = 1 - ESCape ist gedrückt

Diese Routine wird standardmäßig während der Interruptbehandlung aufgerufen. Werden [CTRL]-[SHIFT]-ESC] gleichzeitig gedrückt, erfolgt ein Kaltstart des KC compact.

BDF1 IND MC WAIT PRINTER Zeichenausgabe auf Drucker

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

Die Routine wartet 0,4 Sekunden. War der Drucker in dieser Zeit nicht zur Zeichenübernahme bereit, wird mit CY=0 zurückgekehrt.

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 JUMPBLOCK (HI KL)

B900 HI KL U ROM ENABLE Einblenden 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 DISABLE ROM 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 RESTORE ROM-Konfiguration wieder herstellen

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 JUMPBLOCK (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 Parameter-übergabe.

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

Beispiel:
```
RST 18H
DEFW FADRESS              ;Zeiger far adress
...
FADRESS: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 adress 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
RST 30H

Speichern des ROM-Status in 2BH, 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
RST 38H

Einsprungadresse für alle Hardware-Interrupts (Interruptmode 1)

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

Siehe Abschnitt 3.7.

003B LOW EXT INTERRUPT

Einsprungadresse 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 Zeichenkette

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 Überlauffehler
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 Überlauffehler
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) = negativ
A=0, Z=1 wenn FLO(HL) - FLO(DE) = null
A=1 wenn FLO(HL) - FLO(DE) = 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 Überlauffehler
UR: HL

Funktion: FLO(HL) = LOG10 (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 Überlauffehler
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 Überlauffehler
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 Überlauffehler
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 Überlauffehler
UR: IX,IY
Funktion: HL = HL / DE
DE = HL MOD DE
(muß über RST aufgerufen werden)

DDA3 INT MOD VZ Division zweier Zahlen

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

Funktion: HL = HL 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 Überlauffehler
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 Überlauffehler
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 Überlauffehler
     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 Überlauffehler
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 Überlauffehler
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 Überlauffehler
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 Überlauffehler
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.Kommaposition
     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 Bildwiederholspeicher-Anfangsadresse (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 Grafikcursor
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	Kanalbearbeitungsflag
B1F1-B1F7	Event-Block Tonausgabe
B1F8-B236	Parameterblock Kanal A
B237-B275	Parameterblock Kanal B
B276-B2B4	Parameterblock Kanal C
B2B5	Kontroll-Register des Soundcontrolers
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 Druckerroutine 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 Druckerroutine zugreift. Alle anderen Routinen des Betriebssystems und des BASIC benutzen dann die neue Druckerroutine. Das Ändern von Vektoren der Sprungleiste 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. S O F T W A R E -B A S I C -I N T E R P R E T E R

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   :Programmanfangsmarkierung
     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 Variablenfeld bereichs
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 Controler,Advance Information,Motorola

/5/ F.Claßen,H.Oeffler;Wissensspeicher 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

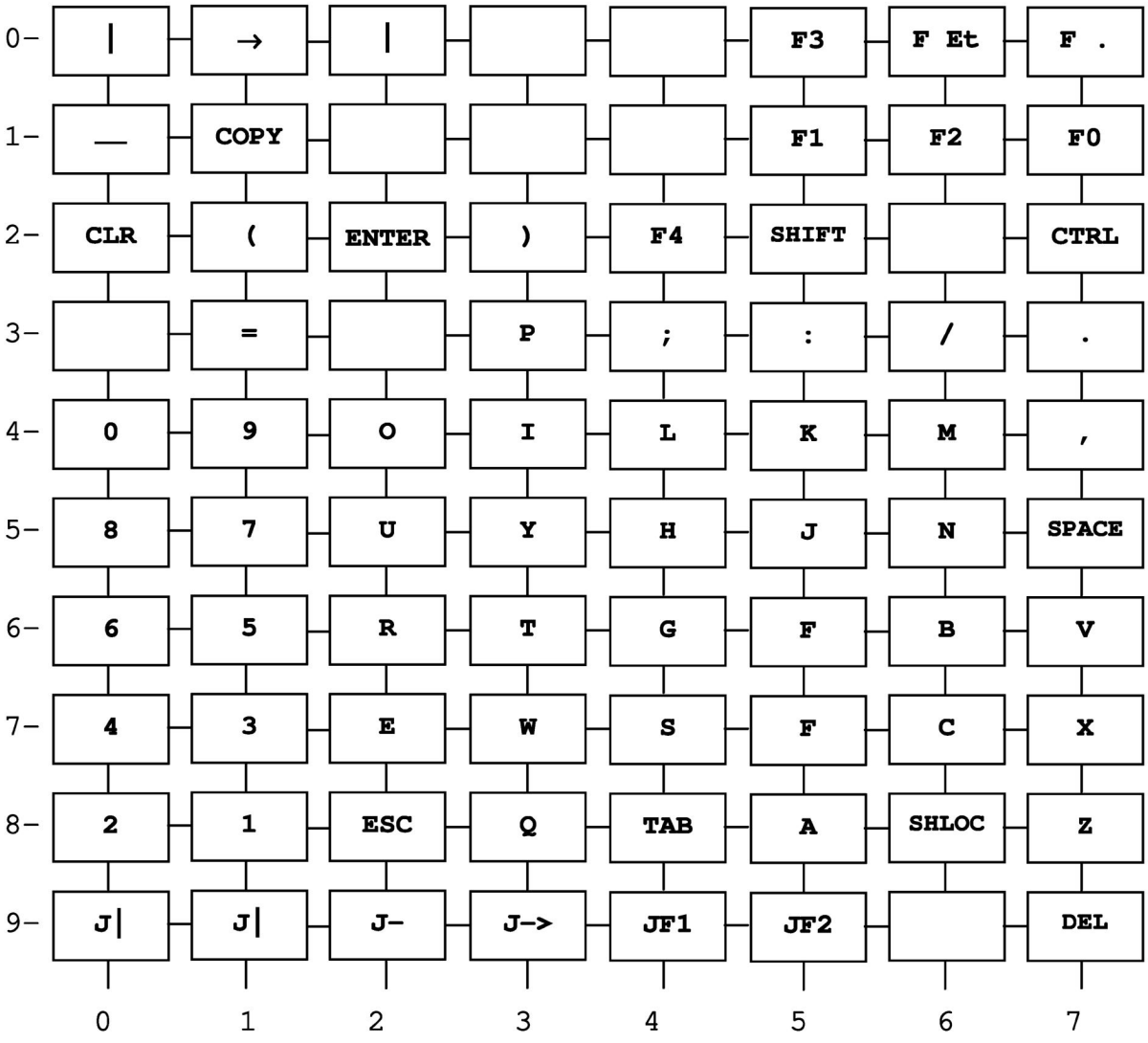
/8/ Gerätebeschreibung KC compact, VEB Mikroelektronik "Wilhelm Pieck"Mühlhausen

Anhang A
Steuercodes

Code hex	Code 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 Ziehwiderstände 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 Soundcontroler und PIO-Port A eingelesen und das eingelesene Byte mit dem Sollwert verglichen werden.



Anhang C
Vektoradressen

(alle Adressen sind hexadezimal angegeben)

Adr.	Bezeichnung
0000	LOW RESET ENTRY
0008	LOW LOW JUMP
000B	LOW KL LOW PCHL
000E	LOW PCBC INSTRUCTION JP (BC)
0010	LOW SIDE CALL
0013	LOW KL SIDE PCHL
0016	LOW PCDE INSTRUCTION JP (DE)
0018	LOW FAR CALL
001B	LOW KL FAR PCHL
001E	LOW PCHL INSTRUCTION JP (HL)
0020	LOW RAM LAM
0023	LOW KL FAR ICALL
0028	LOW FIRM JUMP
0030	LOW USER RESTART
0038	LOW INTERRUPT ENTRY
003B	LOW EXT INTERRUPT
349A	FLO SUB
B900	HI KL U ROM ENABLE
B903	HI KL U ROM DISABLE
B906	HI KL L ROM ENABLE
B909	HI KL L ROM DISABLE
B90C	HI KL ROM RESTORE
B90F	HI KL ROM SELECT
B912	HI KL CURR SELECTION
B915	HI KL PROBE ROM
B918	HI KL ROM DESELECTION
B91B	HI KL LDIR
B91E	HI KL LDDR
B921	HI KL POLL SYNCHRONOUS
B92A	HI KL SCAN NEEDED
BB00	KM INITIALIZE
BB03	KM RESET
BB06	KM WAIT
BB09	KM READ CHAR
BB0C	KM CHAR RETURN
BB0F	KM SET EXPAND
BB12	KM GET EXPAND
BB15	KM EXP BUFFER
BB18	KM WAIT KEY
BB1B	KM READ KEY
BB30	KM GET SHIFT
BB33	KM SET CONTROL
BB36	KM GET CONTROL
BB39	KM SET REPEAT
BB3C	KM GET REPEAT
BB3F	KM SET DELAY
BB42	KM GET DELEAY
BB45	KM ARM BREAK
BB48	KM DISARM BREAK
BB4B	KM BREAK EVENT
BB4E	TXT INITIALISE
BB51	TXT RESET
BB54	TXT VDU ENABLE
BB57	TXT VDU DISABLE
BB5A	TXT OUTPUT
BB5D	TXT WR CHAR
BB60	TXT RD CHAR
BB63	TXT SET GRAPHIK
BB66	TXT WIN ENABLE
BB69	TXT GET WINDOW
BB6C	TXT CLEAR WINDOW
BB6F	TXT SET COLUMN
BB72	TXT SET ROW
BB75	TXT SET CURSOR
BB78	TXT GET CURSOR
BB7B	TXT CUR ENABLE
BB7E	TXT CUR DISABLE
BB81	TXT CUR ON
BB84	TXT CUR OFF
BB87	TXT TXT VALIDATE
BB8A	TXT PLACE CURSOR
BB8D	TXT REMOVE CURSOR
BB90	TXT SET PEN
BB93	TXT GET PEN
BB96	TXT SET PAPER
BB99	TXT GET PAPER
BB9C	TXT INVERSE
BB9F	TXT SET BACK
BBA2	TXT GET BACK
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
BB1E	KM TEST KEY
BB21	KM GET STATE
BB24	KM GET JOYSTICK
BB27	KM SET TRANSLATE
BB2A	KM GET TRANSLATE
BB2D	KM SET SHIFT
BBC9	GRA SET ORIGIN
BBCC	GRA GET ORIGIN
BBCF	GRA WIN WIDTH
BBD2	GRA WIN HIGHT
BBD5	GRA GET W WIDTH
BBD8	GRA GET W HIGHT
BBDB	GRA CLEAR WINDOW
BBDE	GRA SET PEN
BBE1	GRA GET PEN
BBE4	GRA SET PAPER
BBE7	GRA GET PAPER
BBEA	GRA PLOT ABSOLUTE
BBED	GRA PLOT RELATIVE
BBF0	GRA TEST ABSOLUTE
BBF3	GRA TEST RELATIVE
BBF6	GRA LINE ABSOLUTE
BBF9	GRA LINE RELATIVE
BBFC	GRA WR CHAR
BBFF	SCR INITIALISE
BC02	SCR RESET
BC05	SCR SET OFFSET
BC08	SCR SET BASE
BC0B	SCR GET LOCATION
BC0E	SCR SET MODE
BC11	SCR GET MODE
BC14	SCR CLEAR
BC17	SCR CHAR LIMITS
BC1A	SCR CHAR POSITION
BC1D	SCR DOT POSITION
BC20	SCR NEXT BYTE
BC23	SCR PREV BYTE
BC26	SCR NEXT LINE
BC29	SCR PREV LINE
BC2C	SCR INK ENCODE
BC2F	SCR INK DECODE
BC32	SCR SET INK
BC35	SCR GET INK
BC38	SCR SET BORDER
BC3B	SCR GET BORDER
BC3E	SCR SET FLASHING
BC41	SCR GET FLASHING
BC44	SCR FILL BOX
BC47	SCR FLOOD BOX
BC4A	SCR CHAR INVERT

Adr.	Bezeichnung
BBB7	TXT SWAP STREAMS
BBBA	GRA INITIALISE
BBBD	GRA RESET
BBC0	GRA MOVE ABSOLUTE
BBC3	GRA MOVE RELATIVE
BBC6	GRA ASK CURSOR
BC6E	CAS START MOTOR
BC71	CAS STOP MOTOR
BC74	CAS RESTORE MOTOR
BC77	CAS IN OPEN
BC7A	CAS IN CLOSE
BC7D	CAS IN ABANDON
BC80	CAS IN CHAR
BC83	CAS IN DIRECT
BC86	CAS RETURN
BC89	CAS TEST EOF
BC8C	CAS OUT OPEN
BC8F	CAS OUT CLOSE
BC92	CAS OUT ABANDON
BC95	CAS OUT CHAR
BC98	CAS OUT DIRECT
BC9B	CAS CATALOG
BC9E	CAS WRITE
BCA1	CAS READ
BCA4	CAS CHECK
BCA7	SOUND RESET
BCAA	SOUND QUEUE
BCAD	SOUND CHECK
BCB0	SOUND ARM EVENT
BCB3	SOUND RELEASE
BCB6	SOUND HOLD
BCB9	SOUND CONTINUE
BCBC	SOUND AMPL ENVELOPE
BCBF	SOUND TONE ENVELOPE
BCC2	SOUND A ADDRESS
BCC5	SOUND T ADDRESS
BCC8	KL CHOKE OFF
BCCB	KL ROM WALK
BCCE	KL INIT BACK
BCD1	KL LOG EXT
BCD4	KL FIND COMAND
BCD7	KL NEW FRAME FLY
BCDA	KL ADD FRAME FLY
BCDD	KL DEL FRAME FLY
BCE0	KL NEW FAST TICKER
BCE3	KL ADD FAST TICKER
BCE6	KL DEL FAST TICKER
BCE9	KL ADD TICKER
BCEC	KL DEL TICKER
BCEF	KL INIT EVENT

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

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
		A1	IF
0E	Konstante 0	A2	INK
0F	Konstante 1	A3	INPUT
10	Konstante 2	A4	KEY
11	Konstante 3	A5	LET
12	Konstante 4	A6	LINE
13	Konstante 5	A7	LIST
14	Konstante 6	A8	LOAD
15	Konstante 7	A9	LOCATE
16	Konstante 8	AA	MEMORY
17	Konstante 9	AB	MERGE
19	Ein-Byte-Wert	AC	MID$
1A	Zwei-Byte-Wert, dezimal	AD	MODE
1B	Zwei-Byte-Wert, binär	AE	MOVE
1C	Zwei-Byte-Wert, hexadez.	AF	MOVER
1D	Zeilenadresse	B0	NEXT
1E	Zeilennummer	B1	NEW
1F	Fließkommawert	B2	ON
80	AFTER	B3	ON BREAK
81	AUTO	B4	ON ERROR GOTO 0
82	BORDER	B5	ON SQ

T	Bedeutung
83	CALL
84	CAT
85	CHAIN
86	CLEAR
87	CLG
88	CLOSEIN
89	CLOSEOUT
8A	CLS
8B	CONT
8C	DATA
8D	DEF
8E	DEFINT
8F	DEFREAL
90	DEFSTR
91	DEG
92	DELETE
93	DIM
94	DRAW
95	DRAWR
96	EDIT
97	ELSE
98	END
99	ENT
9A	ENV
CE	STOP
CF	SYMBOL
D0	TAG
D1	TAGOFF
D2	TRON
D3	TROFF
D4	WAIT
D5	WEND
D6	WHILE
D7	WIDTH
D8	WINDOW
D9	ZONE
DA	WRITE
DB	DI
DC	EI
DD	FILL
DE	GRAPHICS
DF	MASK
E0	FRAME
E1	CURSOR
E2	-
E3	ERL
E4	FN
E5	SPC
E6	STEP

T	Bedeutung
B6	OPENIN
B7	OPENOUT
B8	ORIGIN
B9	OUT
BA	PAPER
BB	PEN
BC	PLOT
BD	PLOTR
BE	POKE
BF	PRINT
C0	'
C1	RAD
C2	RANDOMIZE
C3	READ
C4	RELEASE
C5	REM
C6	RENUM
C7	RESTORE
C8	RESUME
C9	RETURN
CA	RUN
CB	SAVE
CC	SOUND
CD	SPEED
E7	SWAP
E8	-
E9	-
EA	TAB
EB	THEN
EC	TO
ED	USING
EE	>
EF	=
F0	>=
F1	
F2	>
F3	=
F4	+
F5	-
F6	*
F7	/
F8	
F9	
FA	AND
FB	MOD
FC	OR
FD	XOR
FE	NOT
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

einem besonderen zeitlichen Regime betrieben. Durch das Anlegen des Signals WAIT über drei Takte in einem Zyklus von vier Takten (Pseudomaschinenzyklus, Siehe 2.1.2.) werden die Maschinenzyklen der CPU UA 880 auf vier oder acht Takte eingestellt. Dadurch ändert sich die Ausführungszeit fast aller Befehle. Ihre Kenntnis ist aber für die Programmierung sehr zeitkritischer Vorgänge erforderlich.
Die Abarbeitungszeit der Befehle in Pseudomaschinenzyklen (Mikrosekunden)und die Flagreaktionen werden für Gruppen von Befehlen angegeben. Dabei werden mit r Einzelregister (A bis L), mit rr Doppelregister, mit i eines der Register I oder R, mit ii eines der Register IX oder IY bezeichnet und cc steht für eine der Bedingungen NZ (nicht Null Z=0), Z (Null Z=1), NC (kein Übertrag CY=0), C (Übertrag CY=1), PO (Parität ungerade P/V=0), PE (Parität gerade P/V=1), P (Vorzeichen positiv S=0) und M (Vorzeichen ist negativ S=1). Die Reaktion der Flags auf die Befehle wird durch folgende Symbole dargestellt: 0=Flag wird auf 0 gesetzt, 1=Flag wird auf 1 gesetzt, .=Flag bleibt unverändert und |=Flag wird entsprechend seiner Bedeutung gesetzt. /1,2,3/

LD >, >	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n	I	R
A	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A nl nh	3E n	ED 57	ED 5F
B	47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n		
C	4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n		
D	57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n		
E	5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d	1E n			
H	67	60	61	62	63	64	65	66			DD 66 d	FD 66 d	26 n			
L	6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d	2E n			
(HL)	77	70	71	72	73	74	75	76					36 n			
(BC)	02															

LD >, >	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n	I	R
(DE)	12															
(IX+d)	DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d	DD 76 d						DD 36 d n		
(IY+d)	FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d	FD 76 d						FD 36 d		
(nn)	32 nl nh															
I	ED 47															
R	ED 4F															

8 Bit Ladebefehle
Beispiel: LD A,(HL)
Opcode = 7EH
Quelle = (HL)
Ziel = A
Alle Befehle außer LD A,I und LD A,R ändern die Flags nicht.
Für diese beiden Befehle gilt:

S	Z	H	P/V	CY
|	|	.	*	.

*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 21 nl nh	FD 2A nl nh	FD E1
(nn)		ED 43 nl nh	ED 53 nl nh	22 nl nh	ED 73 nl nh	DD 22 nl nh	FD 22 nl nh			
(SP) PUSH >	F5	C5	D5	E5		DD E5	FD 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 A0	(DE) -> (HL) DE -> DE+1, HL -> HL+1, BC -> BC-1	.	.	0	*	0	.
LDIR	ED B0	(DE) -> (HL) DE -> DE+1, HL -> HL+1, BC -> BC-1 bis BC 0 wird	.	.	0	0	0	.
LDD	ED A8	(DE) -> (HL) DE -> DE-1, HL -> HL-1, BC -> BC-1	.	.	0	*	0	.
LDDR	ED B8	(DE)-> (HL) DE -DE-1,HL -HL-1,BC -BC-1 bis BC 0 wird	.	.	0	0	0	.
CPI	ED A1	Vergleich, ob A=(HL)? HL -> HL+1, BC -> BC-1		#		*	1	.
CPIR	ED B1	Vergleich, ob A=(HL)? HL -> HL+1, BC -> BC-1 bis BC 0 wird oder A=(HL)		#		*	1	.
CPD	ED A9	Vergleich, ob A=(HL)? HL -> HL-1, BC -> BC-1		#		*	1	.
CPDR	ED B9	Vergleich, ob A=(HL)? 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	CB	CB	CB	CB	CB	CB	CB	DD	FD		|	|	0	P	0	|
	07	00	01	02	03	04	05	06	CB	CB							
									d	d	CY	-	-7..		-..0		-
RLCA	07	|							06	06							
RRC	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD		|	|	0	P	0	|
	0F	08	09	0A	0B	0C	0D	0E	CB	CB							
									d	d	->	7..->..0->->CY					
RRCA	0F	|							0E	0E							
RL	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD		|	|	0	P	0	|
	17	10	11	12	13	14	15	16	CB	CB							
									d	d	-CY	-7..		-..0		-	
RLA	17	|							16	16							
RR	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD		|	|	0	P	0	|
	1F	18	19	1A	1B	1C	1D	1E	CB	CB							
									d	d	->	7..->..0->CY->					
RRA	1F	|							1E	1E							
SLA	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD		|	|	0	P	0	|
	27	20	21	22	23	24	25	26	CB	CB							
									d	d	CY	-7..-..0 -0					
									26	26							

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

Verschiebebefehle

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	CB	CB	CB	CB	CB	CB	CB	DD	FD	
	4F	48	49	4A	4B	4C	4D	4E	CB	CB	
								d	d		
								4E	4E		
BIT 2, >	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD	
	57	50	51	52	53	54	55	56	CB	CB	
								d	d		
								56	56		
BIT 3, >	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD	
	5F	58	59	5A	5B	5C	5D	5E	CB	CB	
								d	d		
								5E	5E		
BIT 4, >	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD	
	67	60	61	62	63	64	65	66	CB	CB	
								d	d		
								66	66		
BIT 5, >	CB	CB	CB	CB	CB	CB	CB	CB	DD	FD	
	6F	68	69	6A	6B	6C	6D	6E	CB	CB	
								d	d		
								6E	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	|
ADD IX,	DD	DD		DD	DD		.	.	X	.	0	|
	09	19		39	29							
ADD IY,	FD	FD		FD		FD	.	.	X	.	0	|
	09	19		39		29						
ADC HL,	ED	ED	ED	ED			|	|	X	V	0	|
	4A	5A	6A	7A								
SBC HL,	ED	ED	ED	ED			|	|	X	1	0	|
	42	52	62	72								
INC / /	03	13	23	33	DD	FD	.	.	.	.	.	.
					23	23						
DEC / /	0B	1B	2B	3B	DD	FD	.	.	.	.	.	.
	2B	2B										

IN A,n	DB
	n
IN A,(C)	ED
	78
IN B,(C)	ED
	40
IN C,(C)	ED
	48
IN D,(C)	ED
	50
IN E,(C)	ED
	58
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
INI	ED A2	(HL) =(C) B -B-1,HL -HL+1	X	X	X	1	.	
INIR	ED B2	(HL)=(C) B -B-1,HL -HL+1 bis B=0	X	1	X	X	1	.
IND	ED AA	(HL)=(C) B -B-1,HL -HL-1	X	X	X	1	.	
INDR	ED BA	(HL)=(C) B -B-1,HL -HL-1 bis B=0	X	1	X	X	1	.
OUTI	ED A3	(C)=(HL) B -B-1,HL -HL+1	X	X	X	1	.	
OUTIR	ED B3	(C)=(HL) B -B-1,HL -HL+1 bis B=0	X	1	X	X	1	.
OUTD	ED AB	(C)=(HL) B -B-1,HL -HL-1	X	X	X	1	.	
OUTDR	ED BB	(C)=(HL) B -B-1,HL -HL-1 bis B=0	X	1	X	X	1	.

Befehl	Opcode	n
OUT (C),A	ED 79	
OUT (C),B	ED 41	
OUT (C),C	ED 49	
OUT (C),D	ED 51	
OUT (C),E	ED 59	
OUT (C),H	ED 61	
OUT (C),L	ED 69	
OUTF (C),Flags	ED 71	

Block-I/O-Befehle I/O Befehle

Flags: Z=1, wenn B null wird, sonst 0
Laufzeiten: 5, bei repit. :6 wenn B=0

Flags: konstant
Laufzeit :3

Bef Wirkung	unbed.	NZ	Z	NC	C	PO	PE	P	M
RET > Rückk.	C9	C0	C8	D0	D8	E0	E8	F0	F8
JP >,nn Sprung	C3 nl nh	C2 nl nh	CA nl nh	D2 nl nh	DA nl nh	E2 nl nh	EA nl nh	F2 nl nh	FA nl nh
CALL >,nn CD UP Ruf	C4 nl nh	CC nl nh	D4 nl nh	DC nl nh	E4 nl nh	EC nl nh	F4 nl nh	FC nl 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
DAA	27	Dezimalkorr.	|	|	|	P	.	|
CPL	2F	A -ßA)	.	.	1	.	1	.
SCF	37	CY -1	.	.	1	.	1	1
CCF	3F	CY -ßCY	.	.	X	.	1	|
NEG	ED 44	A -ßA+1	|		|	V	1	|

Befehl	Opcode
RST 0	C7
RST 8	CF
RST 10	D7
RST 18	DF
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
JP (HL)	Sprung zum Inhalt eines Speicher- platzes	E9	1
JP (IX)		DD E9	2
JP (IY)		FD E9	2
RETI	Rückk. von Int.	ED 4D	4
RETN	Rückk. von MNI	ED 45	4

Flags: unbeeinflußt

Befehl	Oc	Zeit
NOP	00	1
HALT	76	1
DI	F3	1
EI	FB	1

Befehl	Oc	Zeit
IM0	ED 46	2
IM1	ED 56	2
IM2	ED 5E	2

Sonderbefehle, Flags: unbeeinfußt

Laufzeiten bei Interruptquittierung: NMI, IM1 :4
IM0 :abh.vom Befehl 4-6
IM2 :6

Sachwortverzeichnis

mikroelektronik

RFT

VEB Mikroelektronik »Wilhelm Pieck« Mühlhausen
im VEB Kombinat Mikroelektronik