

KLEINCOMPUTER



KC compact

SYSTEMHANDBUCH

Teil 1

KLEINCOMPUTER

KC compact

Systemhandbuch Teil1

veb mikroelektronik >wilhelm pieck<
mühlhausen
im veb kombinat mikroelektronik

veb mikroelektronik "wilhelm pieck" mühlhausen

Der Vertrieb dieser Druckschrift erfolgt ausschließlich durch den Herausgeber. Nachfragen bei der Druckerei sind zwecklos.

Ohne Genehmigung des Herausgebers ist es nicht gestattet, das Buch oder Teile daraus nachzudrucken oder auf fotomechanischem Wege zu vervielfältigen.

Hinweise, die zur Verbesserung dieser Dokumentation führen, werden gern entgegengenommen.

Redaktionsschluß: Dezember 1989

G l i e d e r u n g

1.	Einleitung	5
2.	HARDWARE	6
2.1.	Das Blockschaltbild des KC compact	6
2.1.1.	Die Speicherkonfiguration	7
2.1.2.	Die CPU UA 880	8
2.1.3.	Die I/O-Adressen im KC compact	8
2.1.4.	Die zentrale Zustandssteuerung	9
2.1.5.	Der Videocontroller (CRTC)CM 607	12
2.1.6.	Die CIO U82536	14
2.1.7.	Das parallele Druckerinterface	16
2.1.8.	Das ROM-Select	17
2.1.9.	Der parallele Dreitorbaustein (PIO)KP580B55	18
2.1.10.	Der Soundcontroller AY 9-8912	19
2.1.11.	Die Tastaturansteuerung	22
3.	SOFTWARE-BETRIEBSSYSTEM	24
3.1.	Das Systemkonzept	24
3.2.	Die Speicheraufteilung	24
3.3.	Die Magnetbandaufzeichnung	25
3.3.1.	Verfahren	25
3.3.2.	Dateiaufbau	25
3.3.3.	Fehlermeldungen	27
3.3.4.	Dateitypen	27
3.4.	Die Bildschirmausgaben	28
3.4.1.	Video-RAM	28
3.4.2.	Farben auf dem KC compact	30
3.5.	Der Druckertreiber	32
3.6.	Die RSX-Kommandos	32
3.7.	Interrupts	38
3.8.	Restarts	44
3.9.	Die Sprungleisten	45
3.9.1.	Zentrale Sprungleiste	45
3.9.1.1.	Tastatur-Routinen - KEY MANAGER (KM)	45
3.9.1.2.	Textausgabe-Routinen - TEXT VDU (TXT)	51
3.9.1.3.	Grafik-Routinen - GRAPHICS VDU (GRA)	60
3.9.1.4.	Bildschirm-Routinen - SCREEN PACK (SCR)	66
3.9.1.5.	Kassetten-Routinen - CASSETTE MANAGER (CAS)	74
3.9.1.6.	Soundausgabe-Routinen - SOUND MANAGER (SOUND)	79
3.9.1.7.	Zentrale -KERNEL (KL)	83
3.9.1.8.	Maschinennahe Routinen - MACHINE PACK (MC)	89
3.9.1.9.	Routine für die Sprungleiste - JUMPER (JUMP)	92
3.9.1.10.	Indirections der Betriebssystem-Packs (IND)	92
3.9.2.	Obere Sprungleiste des Kernel - HIGH KERNEL JUMP-BLOCK (HI KL)	95
3.9.3.	Untere Sprungleiste des Kernel - LOW KERNEL- JUMP-BLOCK (LOW)	97
3.9.4.	BASIC-Vektoren	101
3.9.4.1.	Der Editor	101
3.9.4.2.	Fließkomma-Routinen	102

3.9.4.3.	Integer-Routinen	106
3.9.4.4.	Konvertierungs-Routinen	108
3.10.	Die Arbeitszellen	110
3.10.1.	Betriebssystem-Arbeitszellen	111
3.10.2.	BASIC-Interpreter-Arbeitszellen	114
3.11.	Das Patchen von Vektoren	117
4.	SOFTWARE - BASIC-INTERPRETER	119
4.1.	Einleitung	119
4.2.	Die Speicheraufteilung bei der Arbeit mit BASIC ..	119
4.3.	Der Aufbau eines BASIC-Programms	120
4.4.	Variablentypen	120
4.5.	Der BASIC-Stack	121
4.6.	Die Verwaltung des HIMEM	121
4.7.	BASIC und Maschinencode	122
5.	Literaturhinweise	124
Anhang A:	Steuercodes	125
Anhang B:	Tastaturmatrix	126
Anhang C:	Vektoradressen	127
Anhang D:	BASIC-Token	130
Anhang E:	UA 880-Befehle und Laufzeiten	132
Anhang F:	Sachwortverzeichnis	141

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

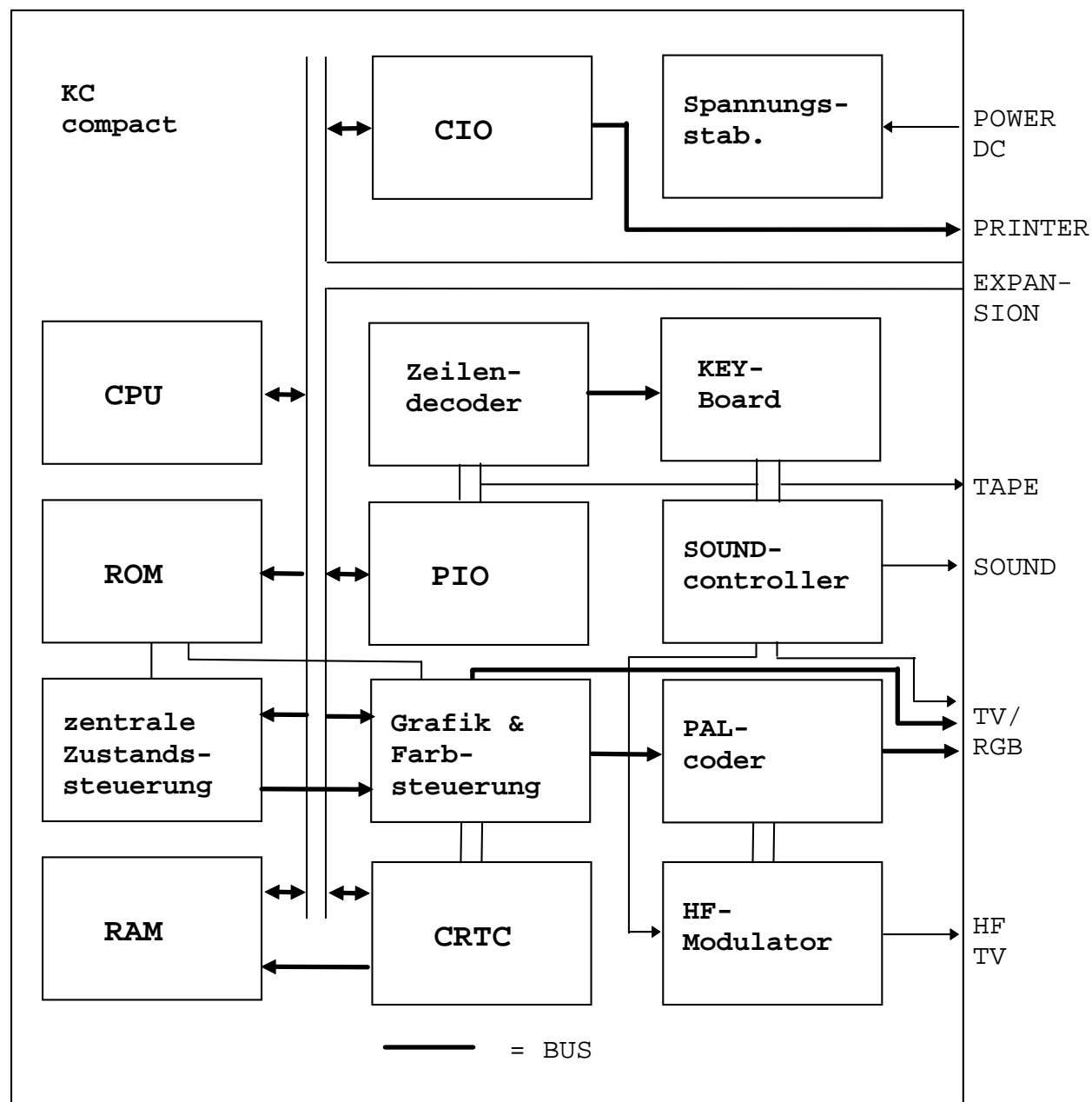
Das vorliegende Systemhandbuch gibt einen kurzen Überblick über die Ressourcen der Hardware und des Betriebssystems, die durch den KC compact bereitgestellt werden. Die Hardware wird im Überblick vorgestellt. Alle I/O-Bausteine, ihre Adressierung, die für den Anwender nutzbaren Register und die Wirkung 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 Controller (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 Kassettenein- und -ausgabe, wählt über einen Dekoder die Zeilen der Tastatur an und bedient einen Soundcontroller.

Der Soundcontroller 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 Soundcontroller 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.

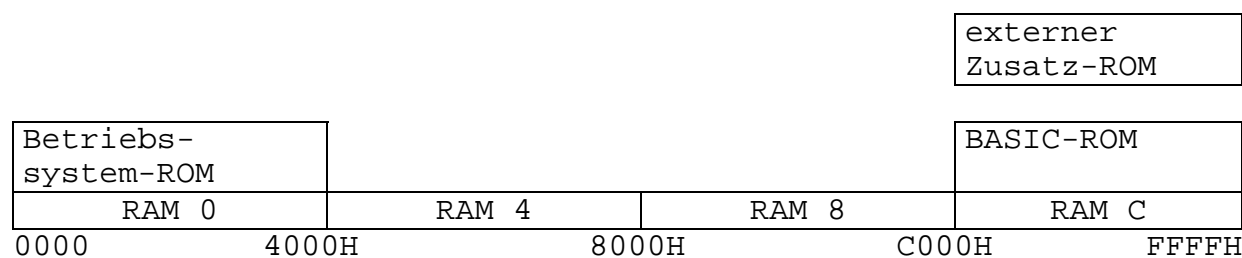


Abb. 2.2 Speicherkonfiguration im KC compact

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		DATENBUS	HEX		FUNKTION	R/W	
B	C	Datenreg	B	C	Reg.		
011111XX	XXXXXXXX	11XXXXXXXX	7F	XX	XX	reserviert!	W
011111XX	XXXXXXXX	10XXXXXXXX	7F	XX	XX	MF-Register	W
011111XX	XXXXXXXX	01XXXXXXXX	7F	XX	XX	FW-Register	W
011111XX	XXXXXXXX	00XXXXXXXX	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 Bildschirm 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 Farb- wert	Paletten- farbwert		grün	rot	blau	Farbe
	dez.	hex.				
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 Bildwiederholpeicher 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 Videocontroller (CRTC)CM 607 /4/

Der CRTC ist ein sehr leistungsfähiger Baustein. Er stellt die Adressen für den Bildwiederholtspeicher 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 (character 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 (character 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	horizontal total	8	Anzahl Taktperioden/totale Zeile -1, d.h. Bild, Bildrand und Strahlrücklauf
1	-/W	horizontal displ	8	Anzahl Taktperioden/sichtbare Bildzeile, d.h. ohne Bildrand und Strahlrücklauf
2	-/w	horizontal 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 Bildwiederholerspeichers im 16 KByte-Adreßbereich des CRTC. Mitten im Bildwiederholerspeicher kann ein Sprung vom Ende des 16 KByte-Bereiches auf den Anfang erfolgen.
13	R/W	start adress low	8	Lowteil der Anfangsadresse des Bildwiederholerspeichers im 16 KByte-Adreßbereich des CRTC
14	R/W	cursor adress high	6	Highteil der Adresse der momentanen Cursorposition im 16 KByte-Adreßbereich des CRTC.
15	R/W	cursor adress low	8	Lowteil der Adresse der momentanen Cursorposition im 16 KByte-Adreßbereich 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-Adreßbereich 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-Adreßbereich 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 Bildwiederholtspeicher 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 Einschalt-initialisierung 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      ;Sperrern 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 I/O-Druckerinterfaces	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
BUSY	0F5XXH	6	gleichzeitig Kassettenausgabe 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

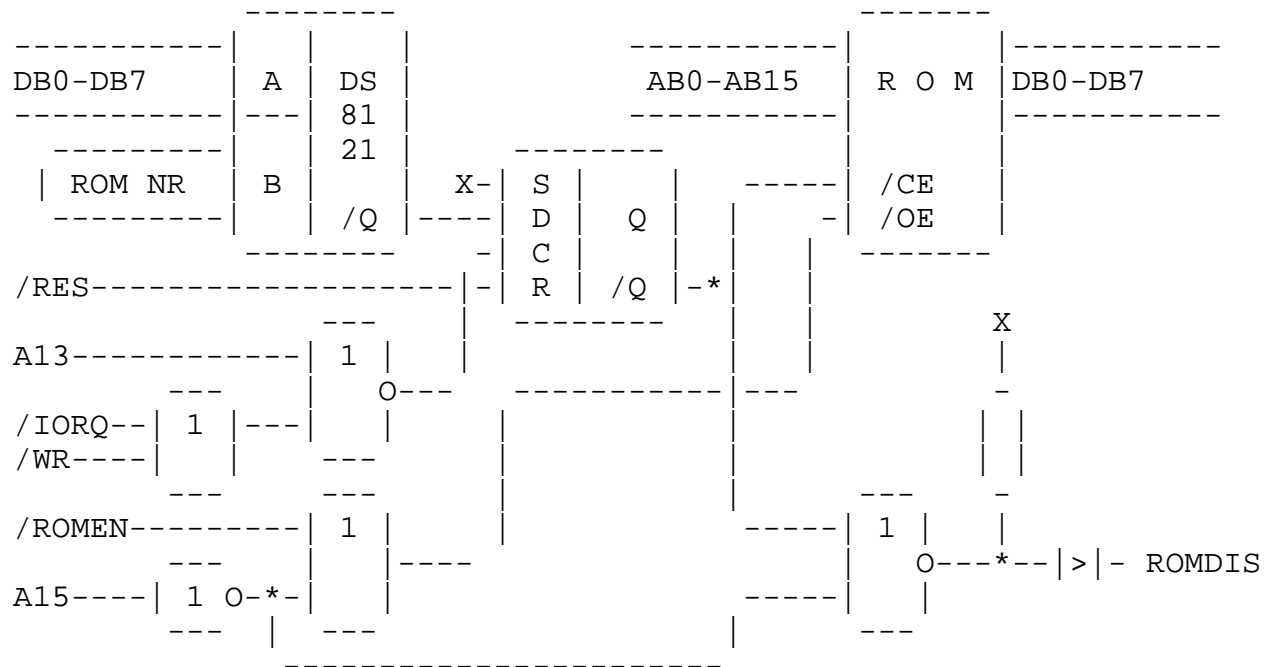


Abb. 2.4 Die Ansteuerschaltung für ein externes ROM

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 Soundcontrollers, 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 Soundcontrollers 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 Soundcontrollers (chip select).
- Bit 7: Verbunden mit BDIR des Soundcontrollers.

2.1.10. Der Soundcontroller AY 9-8912 /6/

Die Tonerzeugung und die Tastaturabfrage wird im KC compact durch einen Soundcontroller 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 Soundcontrollers gezeigt. Der Soundcontroller 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 BCI des Soundcontrollers

BDIR	BCI	Funktion
1	1	Setzen des Registerzeigers. Der Wert an den Datenleitungen des Soundcontrollers 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 Soundcontrollers wird hochohmig geschaltet.

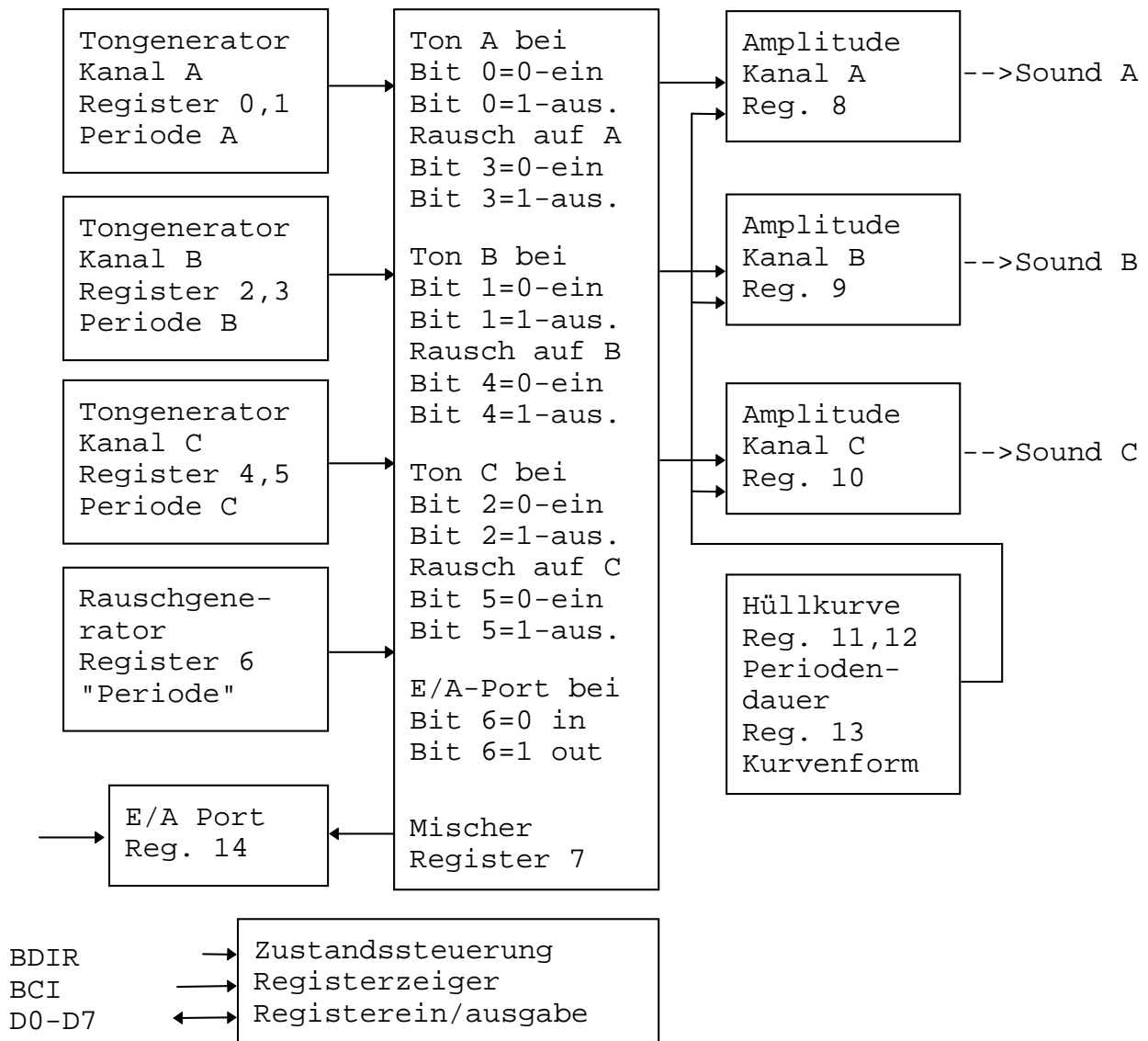


Abb.2.5 Blockschaltbild des Soundcontrollers

Der Soundcontroller wird mit einer Taktfrequenz von 1MHz betrieben. Der 12-Bit-Wert der Periodendauer wird für die einzelnen Kanäle nach der Formel $P = \text{ROUND}(62500/\text{FREQUENZ})$ berechnet (Frequenz in Hz).

Es folgt eine Beschreibung der einzelnen Register.

Tab.2.11 Die Steuerregister des Soundcontrollers

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



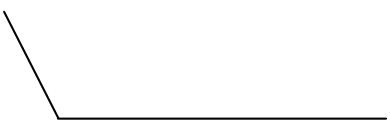



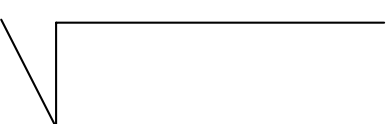

2.1.11. Die Tastatursteuerung

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 Soundcontrollers und im weiteren durch Port A der PIO eingelesen. Im folgenden wird an einem Beispielpogramm 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 Soundcontroller
        OUT   (C),A      ;Zeilenleitung wählen
        OUT   (C),C      ;Sound-Controller 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-Controller setzen
        LD    B,0F6H     ;PIO Port C
        OUT   (C),C      ;BC1 an Sound-Controller 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-RAMs 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	Bildwiederholtspeicher

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 umprogrammiert 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 Grafikinformatoren für die obersten Pixel-Zeilen aller 8*8-Punkte-Matrizen aller Zeichenpositionen abgespeichert. Im folgenden Beispiel wird das deutlich:

```
10 CLS:FOR I=1 TO 24:PRINT "Testprogramm":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	EFCE	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-Controllers. Dadurch, daß die Adreßleitungen MA10 und MA11 des Controllers aber nicht angeschlossen sind, ergibt sich beim Inkrementieren 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 Tintenummer 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-Farbwerte konvertiert. In BASIC werden die Farben nach ihrer Helligkeit auf einem monochromen Monitor sortiert und neu nummeriert. 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 Treiberoutine 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-
;niierter 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 0A00H), 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 letzten 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 Integerzahlen 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$="Testprogramm"
20 b=12.008
30 TEST,b,a$
...

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 Farbbänderungen 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 einreicht. 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 Event 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 SYNCHRONOUS 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 Systemressourcen 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. Synchroner 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 gebraucht. 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

mikroelektronik

RFT



veb mikroelektronik · wilhelm pieck · mühlhausen
im veb kombinat mikroelektronik