

Kurztitel:

David von Oheimb
Das Systemhandbuch für den PC-1600

ISBN 3-925641-08-4

Copyright (c) 1987 by:

Klaus Ditze

Hard- & Software für Mikrocomputer, Verlag
Nikolaus-Ehlen-Straße 6, D-5354 Wollerswist

Copyright for Hardware-Specifications and Circuit Diagramms by:

SHARP CORPORATION
Industried Instruments Group
Yamatokoriyana, Nava 639-11,
Japan

Deutschlandvertretung:

SHARP ELECTRONICS (EUROPE)
GmbH
Sonninstraße 3
2000 Hamburg 1

Alle Rechte vorbehalten, insbesondere die des Vortrags, der Übertragung durch elektronische Medien, der Übersetzung sowie des Nachdrucks (auch auszugsweise) und der Vervielfältigung. Diese Vorbehalte gelten für Text, Programmlistings und Diskettensoftware.

Für Fehler im Text, in den technischen Beschreibungen, den Listings etc., sowie deren Folgen, kann keine Haftung übernommen werden.

- A) Einführung
- B) Betriebssystem
- 1. Speicherorganisation
 - Bankswitching
 - Grobeinteilung des ROMs
 - RAM-Module
- 2. Reset und Booten
 - Reset-Ursache
 - Reset der Peripherie
 - Boot-Programme
 - CHECK-Meldungen, WAKE\$
- 3. Interrupt
 - Ports und Parameter -Interrupt-Ursachen
- 4. Zweitprozessor LH-5803
- 5. Anzeige
 - Hardware-Konzeption
 - Routinen
 - Zeichengenerator
- 6. Tastatur
 - Tastaturmatrix und Tastaturpuffer -Routinen
 - Tastentabellen
- 7. Buzzer (Lautsprecher) -Routinen
- 8. Timer, A/D-Wandler -Aufgaben -Routinen
 - Timer-RAM
- 9. Serielle Schnittstellen -Hardware
 - Routinen
- 10. Centronics-Interface
 - Konzeption
 - Routinen
 - Zeichensatz

- 11. Plotter
 - Schrittmotor-Steuerung
 - Routinen -Zeichencodierung
 - Interrupt-Tasten
- 12. Cassettenrecorder-Interface
 - Hardware
 - Logisches Aufzeichnungsformat
 - Routinen
- 13. Diskettenlaufwerk -
Routinen
- 14. Dateien
 - Struktur und Organisation
 - Routinen -RAM-Disk-Routinen
 - Disketten-Puffer
- 15. Editor (Ablaufbeschreibung)

C) BASIC-Interpreter

- 1. Programmspeicher und Variablenbereich
- 2. Interne Datendarstellung und Stacks
- 3. BASIC-Interrupts
- 4. Token
- 5. Befehlsverarbeitung und Funktionen
 - Direkte Befehle
 - Programm-Verarbeitung
 - Funktionen

D) Programmbeispiele

- 1. Allgemeine Tips zur Maschinenprogrammierung
- 2. Vom BASIC-PRINT zum Maschinenprogramm
- 3. POINT in Maschinensprache
- 4. Umleitung von Systemroutinen
- 5. Soundgenerator
- 6. Benutzung des Editors
- 7. Der NEW-Retter
- 8. Ein Boot-Programm
- 9. Hex- und Diskmonitor

E) Anhang

- 1.Referenzliste
- 2.Disketten-Inhalt
- 3.Vollständige Tasten-und ASCII-tabelle
- 4.Tastentabelle
- 5.Token-tabelle
- 6.I/O-Adressen (Ports)
- 7.System-RAM (Memory-Map)
- 8.Sprungtabelle
- 9.Schaltpläne

Folgende Abkürzungen werden benutzt:

(0123) : Inhalt der Adresse 0123
(FEDC/) : 2-Byte-Wert, der in FEDC/FEDD steht (Low vor High)
(HL) : Inhalt der Adresse, die im HL-Register gehalten wird
(HL+01) : Wie oben, aber die folgende Adresse
b0 : Bit 0 eines Byte (niederwertigstes Bit)
b7 : Höchstwertigstes Bit
n1 : Linkes Nibble (höherwertiger 4-Bit-Block eines Bytes)
n0 : Rechtes Nibble
adr : Adresse
hi : High-Byte
lo : Low -Byte
Areg : Z80-Register A
>DEreg : Der Returnwert einer Routine steht in Registerpaar DE
c12 : Routine mit dem Funktionscode 12 (im Creg)
sc : Carry-Flag gesetzt
nc : ----,----- zurückgesetzt
sz : Zero -Flag gesetzt
nz : ----,----- zurückgesetzt
^7F : Bit 7 zurückgesetzt (AND 7F)
v80 : Bit 7 gesetzt (OR 80)
50d : Die Dezimalzahl "50"

Zahlen- und Adressenangaben verstehen sich, wenn nicht anders vermerkt, grundsätzlich hexadezimal.

B.) B E T R I E B S S Y S T E M

1.SPEICHERORGANISATION

< BANKSWITCHING >

Der Z80 kann mit seinen 16 Adreßleitungen maximal 64 KB Speicher direkt adressieren. Da dies aber nicht ausreicht, muß auf die sog. Bankswitching-Methode zurückgegriffen werden. Dabei werden durch einen Port (mittels des OUT-Befehls) verschiedene Speicherbausteine, die die gleiche Adresse belegen, selektiert. Beim PC-1600 können durch den Port 31 die Adreßbereiche 4000-7FFF und 8000-BFFF unabhängig voneinander in 8 Bänke unterteilt werden. Die Funktion des Ports läßt sich am besten binär beschreiben:

b0	:	Bank 0-1 für Adressen 0000-3FFF=Page 0	(unbenutzt)
b1-b3	:	Bank 0-7 für Adressen 4000-7FFF=Page 1	
b4-b6	:	Bank 0-7 für Adressen 8000-BFFF=Page 2	
b7	:	Bank 0-1 für Adressen C000-FFFF=Page 3	(unbenutzt)

Beispiel:

3E 2E LD A,2E	=0010 1110 bin
D3 31 OUT (31),A	^^^ 010:Adr.8000-BFFF Bank 2
	^^^ 111:Adr.4000-7FFF Bank 7

Besonderheiten:

Port 3D

In diesem Port ist normalerweise b2 gesetzt (kann von adr F07D gelesen werden, mit IN nicht lesbar). Wird es zurückgesetzt, so wird auf Bank 3 ein verstecktes BASIC-ROM selektiert (Bank 3b).

Port 28

Der Slot 2 (8000-BFFF, Bank 2+3) kann bei Modulen, die größer als 32KB sind, mit diesem Port weiter unterteilt werden.

Routinen:

0190 BANKSET

Setzt Bank Areg auf Page Breg

0193 BANKREAD

Lies die mom. selektierte Bank der Page Breg >Areg

018D MEMORYCHK

Teste Speicher Bank Dreg,hi-Byte der adr in Ereg (40-B8)
>sc,Areg=00:Kein Speicher vorhanden
>nc,Areg=01:RAM
>nc,Areg=03:ROM

019C BANKJUMP

Sprung zur Bank A'reg,adr HL'reg ;Zweitreg werden zerstört

019F BANKCALL

Call Bank A'reg,adr HL'reg : -----,-----

RST18,nn (Mit Doppelimmediate):BANKJP

Das Doppelimmediate gibt adr und Bank an:
0000-3FFF:Bank 0
4000-7FFF:Bank 3
8000-BFFF:Bank 6
C000-FFFF:Bank 0,adr 4000-7FFF

RST20,b,nn (Mit 3 Immediates) :BANKCAL

b : Bank
nn :adr
Beispiel :E7 05 08 40:CALL 05,4008

<< GROBEINTEILUNG DES ROMS >>

Bank 0,0000-3FFF

Vektoren Reset-Routine,wichtige Unterprogramme
(von allen Bänken erreichbar,wird nie umgeschaltet)

Bank 0,4000-7FFF

einige BASIC-Befehle,Editor

Bank 3,4000-7FFF

Interrupt-Routine,Print- und Tastenroutinen,Routinen für
Zweitprozessor,RAM-Disk und Schnittstellen

Bank3b,4000-7FFF

Nur BASIC-Befehle

Bank 4,4000-7FFF

Plotter (wenn angeschlossen)

Babk 4,4000-7FFF

Centronics-Iaterface (wenn angeschlossen)

Bank 5,4000-5FFF

Floppy (wenn Plotter oder Centronics-Iaterf. angeschl.)

Bank 5,6000-7FFF

Cassetteninterface (wenn Plotter angeschlossen)

Bank 6,8000-BFFF

Anzeigeroutinen, Steuerung von Timer und seriellen Schnittstellen, Zeichen- und Tokentabelle

Die ROMs der Peripheriegeräte (adr 4000-7FFF) sind modular organisiert. Beim Reset erkennt sie das Betriebssystem an einem bestimmten IDENTITY-Code (=ID-Code) und vermerkt sie in den System Adressen. Ihnen können eigene Arbeitsspeicher – Bereiche , BASIC Befehle, Reset- und Interrupt-Anforderungen zugewiesen werden.

Aufbau eines ROM-Moduls (Start 4000 oder 6000, im folgenden wird nur noch von 4000 die Rede sein, aber es gilt entsprechend auch für 6000):

```
4000 : 43 ID-Code
4001 : 16 ID-Code
4002 : JP XXXX Reset-Sprung
4005 : JP XXXX Interrupt-Sprung
4008 : JP XXXX Sprung zu wichtigen Routinen
400B : JP XXXX Nicht benutzt
400E : JP XXXX Sprung für AUTORUN.BAS -Suche
4011 : XX XX Pointer für Device Name (z.b."COM1:")
4013 : XX XX Pointer für Tokentabelle (+0036)
4015 : JP XXXX Sprung für File-Bearbeitung
```

"ROM-Bit:"

In F0AE stehen, bitweise codiert, die vorhandenen ROM-Module mit Startadr 4000, in F0AF die mit Startadr 6000. Dabei bedeutet b6 die Bank 1, b5 die Bank 2, ..., b0 die Bank 7.

Arbeitsspeicher:

Mit CALL 02DF kann den ROM-Modulen ein eigener Arbeitsspeicher zugeteilt werden. Dazu muß im DReg die Größe in Bytes und im Creg die Nummer des Moduls gesetzt werden.

Mögliche Werte für Creg:

```
00      : Standard-Systemspeicher
01-07   : ROM Bank 1-7, Start 4000 (EXROM1-EXROM7)
08-0E   : ROM Bank 1-7, Start 6000 (EXROM8-EXROME)
0F      : COM-Puffer ( wie INIT "COM:", Puffergröße)
10      : Filepuffer ( wie MAXFILES=n ; DReg = n*313d Bytes)
```

Die zugeteilte Startadresse steht in (F02E+ 2*Creg /), die Endadresse ist (F02E+ 2*Creg -2 /)-1.

Beispiel :

```
11 28 04  LD DE,0428  setzt die Standardgröße des Diskpuffers,
0E 05      LD C,05      (F038) : EBD7 Start
C3 DF 02  JP 02DF      (F036) : F000 Ende+1
```

System-Verwaltungsbereich für die ROM-Module:

F02E-F04F : Arbeitsspeicher-Pointer
F0AE-F0BA : ROM-Bits

<<< RAM-MODULE >>>

Das Standard-RAM ("S0:",Bank 0,C000-FFFF) kann in den Modulslots "S1:" und "S2:" durch RAM- bzw. EPROM-Module erweitert werden. Diese werden beim Reset erkannt und in folgenden Adreßbereichen verwaltet:

F015-F01E : S1

F01F-F028 : S2

F029-F02C : S0

F050-F053 : S1,nur EPROM oder Schreibschutz

F054-F055 : S1,nur RAM

F056-F059 : S2,nur EPROM oder Write Protected

F05A-F05B : S2,nur RAM

F123-F126 : Codierte Modulbestückung (für Reset)

F860-F863 : Module für PC-1500-Mode (MODE 1)

Zusatzinformationen stehen im Header-Bereich (8-Byte-Kopf), der je nach Modulgröße bei adr 8000, A000 oder B000 beginnt. Entsprechend der Initialisierung mit INIT "Sn:" ergibt sich folgendes Bild (hier am Beispiel der Adresse 8000):

"H"	"P"	"F"	"S" (System software)
8000 : FF	55	55	55
8001 : FF	Start hi^7F	80	00
8002 : FF	BASIC-	neg.Checksum(8000-801F)	Boot
8003 : FF	Start	C3 wenn Boot, sonst 00	adr
8004 : FF	Modullänge hi	Boot-	00
8005 : FF	BASIC-	adr	00
8006 : FF	Ende	00	00
8007 : FF	80	81 wenn RAM, 01 wenn WP	82
8008 : RESERVE-Bereich		log. Format (siehe RAM-Disk)	

Normalerweise sind den Slots folgende Adressen zugeordnet:

Bank 0	Bank 1	Bank 2	Bank 3
8000-BFFF Slot Ia	Slot Ib	Slot IIa	Slot IIB

Sie können durch zwei Routinen umadressiert werden:

```

0196 SLOT1MAP
  Areg=00 :Normalzustand
  Areg=01 :Slot 1b wird zusätzlich dem Adreßbereich
           4000-7FFF der Bank I zugeordnet

0199 SLOT2MAP
  Areg=00 :Normalzustand
  Areg=01 :Slot 1la wird dem Adreßbereich
           0000-3FFF der Bank 1 zugeordnet
  Areg=02 :Slot 1lb wird dem Adreßbereich
           0000-3FFF, Bank 1 zugeordnet
           Falls 4000-7FFF, Bank 1 durch SLOT1MAP noch nicht
           belegt ist, wird hier zusätzlich Slot Umadressiert

00E8 SLOTST (Headertest)
  Bank Areg, hi-Byte der Adresse (meist 80) im Dreg
  >Areg= 00:Fehler
      FO:"P"
      F1:"F"
      F2:"S"
      FF:"M"
    
```

2. RESET UND BOOTEN

<<< RESET-URSACHE >>>

Beim Reset (Start von 0000) wird zunächst der Interrupt verboten, der Stackpointer initialisiert, der Bildschirm abgeschaltet, die BRK-Taste abgewartet und vom Timer-Baustein mit

```
0E 15          LD C,15
CD D5 01       CALL 01D5
```

die Reset-Ursache abgefragt. Der Reset-Code wird in FA1B gespeichert und bleibt dort als wichtiger Parameter für die Reset Routinen verfügbar.

Dabei gilt:

```
b0 : ALL RESET
b1 : Interner RESET
b2 : Externer RESET (z.B.vom Plotter)
b3 : unbenutzt
b4 : POWER ON alt ON-Taste/CALL 0
b5 : Externer POWER ON
b6 : WAKE$(0)
b7 : WAKE$(I)=CI
```

<<< RESET DER PERIPHERIE >>>

- * Reset der wichtigsten Systemparameter (je nach Reset-Ursache)
- * Feststellung der angeschlossenen Geräte (ROM-Bit.)
- * CALL 4002 an alle existierenden Geräte Funktionscode In Areg
Areg=07:Gerät kann sich ggf. selbst wieder abmelden
- * Feststellung der Speichermodul-Bestückung
Falls sie sich beindert hat, wird b7 von FA23 gesetzt und später
"NEWO?" angezeigt (siehe CHECK-Heldungen)
- * Test, ob ein POWER ON möglich ist (d.h. das Programm
das CALL 0005 = POWER AOFF aufgerufen hat, kann ohne Reset
weiterlaufen)
Bedingungen:
 - Anschalten nach AUTO POWER OFF oder POWER AOFF-Kommando
 - Geräte- und Modulbestückung gleich geblieben
 - Alle Geräte haben volle Batterien (CALL 4002 Areg=02)
 - Die Routine APOTEST Bank (FOCB), adr (FOCC/) setzt als
Returnwert das Carry-Flag zurück
- * Geräte-Reset/Totalreset (CALL 4002, Areg=01/00)
- * Timer- und Schnittstellen-Reset/Totalreset

<<< BOOT-PROGRAMME >>>

Boot-Programme werden in folgender Reihenfolge gesucht und, falls vorhanden, ausgeführt:

- * Peripheriegeräte (z.B. Floppy) (CALL 4002, Areg=05)
falls gefunden (CALL 4002, Areg=06)
- * Systemsoftware (EPROM)
- * Systemsoftware (RAM)
- * Sprung nach Bank (F0DC), adr (F0DD/):
Hier steht nach einem Totalreset die Startadresse des BASIC
interpreters

Der BASIC-Interpreter setzt je nach Reset-Ursache seine Systemadressen und löscht bei einem Totalreset den internen und den mit INIT "Sn:", "M" selektierten Programmspeicher.

Falls keine Meldungen (z.B. CHECK-Meldung) auszugeben sind, durchsucht er mit dem Sprung 400E die Peripheriegeräte nach der BASIC Datei AUTOBUN.BAS, lädt sie ggf. in den Speicher und startet sie automatisch.

Andernfalls, wenn ein schon im Speicher befindliches Programm mit ARUN beginnt, wird dieses mit JP 028F ausgeführt.

Ansonsten beginnt die Direkteingabe mit JP 0253.

<<< CHECK-MELDUNGEN, WAKE\$ >>>

Ist im Bereich FA21-FA24 mindestens 1 Bit gesetzt, so wird eine entsprechende CHECK-Meldung angezeigt.

Dabei sind den Bits folgende Zeichen zugeordnet:

	b7	b6	b5	b4	b3	b2	b1	b0
FA21 :	0	6	N	H	L	K	J	I
FA22 :	V	U	T	S	R	Q	P	8
FA23 : NEWO?	1	2	3	4	5	7	9	
FA24 :	A	B	C	D	E	F	G	H

JP 0023 gibt die Meldung aus und wartet auf die CL-Taste.

Liegt ein WAKE\$-Ereignis vor, so wird der entsprechende String in den Tastaturpuffer eingelesen und die Direkteingabe des BASIC Interpreters aufgerufen.

Adresse des Befehisstrings:

FF00-FF1F:WAKE\$(0)
FF26-FF3F:WAKE\$(I)

3. INTERRUPT

<< PORTS UND PARAMETER >>

Die Adresse der Interrupt-Routine für den normalen, maskierbaren Interrupt wird indirekt ermittelt. Dabei steht das High-Byte der Speicherstelle, in der die Interrupt-Startadresse gehalten wird, im I-Register des Z80. Das Low-Byte steht im Port 39. Es wird also IM2 verwendet.

Ein Beispiel:

```
Ireg      : 03
Port 39    : 21   >adr 0321
(0321/)    : 082C >Startadr der Interruptroutine:082C
```

Die Interrupt-Ursache wird aus dem Port 32 gelesen. Dieses Byte wird mit dem Inhalt des Ports 35 (=Interrupt-Maske) durch die AND-Funktion verknüpft und somit nur die Interrupts erlaubt, deren Bits in beiden Ports gesetzt sind. Der Timer besitzt eigene Register für Interruptursache und -maske (siehe TIMER). Durch LD A,01 und OUT (1B),A kann ein sofortiger Interrupt ausgelöst werden.

<<< INTERRUPT-URSACHEN >>>

Die Bits in den Ports 32 und 35 haben folgende Bedeutung:

- b0 : Daten auf den Schnittstellen empfangen Sie werden im Puffer abgelegt.
- b1 : PC-1600-Peripherie, z.B. Druckertasten
Die durch die Bits im "ROM-Bit" und in der ROM-Interrupt Maske 0 angegebenen ROM-Module werden aufgerufen
(CALL 4005, Areg=00)
- b2 : PC-1500-Peripherie
- b3 : Zweitprozessor LH-5803 übergibt die Kontrolle wieder an den Z80

b4 : 1/64-sek-Interrupt:
* Zyklische Abfrage der Tastatur und Cursorblinken,
nur wenn (F0B6) v80
* ROM-Interrupt 2 (CALL 4005,Areg=02)
* Wenn (F0B7) v80:User-Interrupt 2
Bank (F0BC),adr (F0BD/)
Bei den User-Interrupts müssen in den Bytes, die die
Bank an geben,die b6 und b7 gesetzt sein (also vC0)

b5 : unbenutzt

b6 : TIMER-Interrupt
- 0,5-sek-Interrupt:
* Batterietest:Setzt ggf. das BATT-Symbol
* ROM-Interrupt I (CALL 4005,Areg=01)
* Wenn (F0B5)v80:User-Interrupt 1
Bank (F0BF),adr (F0C0/)
* ON ADIN- und ON PHONE-Interrupt
* Setze APO-Zähler (F0AC/),ggf. POWER AOFF
* Setze Timeout-Counter (F156) für Schnittstellen

- 1-sek-Interrupt:Tastenabfrage für KEYSTAT I
- User-Interrupt 3,ON TIME\$: Bank (F0C2),adr (F0C3/)
- User-Interzupt 4 ALARM\$: Bank (F0C5),adr (F0C6/)
- User-Interrupt 5 WAKE\$(0) : Bank (F0C8),adr (F0C9/)

Der NMI (CALL 0066) dient zum Zugriff des LH-5803 auf den Bildschirm, wenn der Z80 in Ruhe ist (Emulation des PC-1500 Anzeige Modus, MODE 1)

4. ZWEITPROZESSOR LH-5803

Dieser Prozessor sorgt für die Kompatibilität zum LH-5801 des PC-1500. Er wird für die meisten BASIC-Funktionen und als Treiber für die PC-1500-Peripherie eingesetzt. Sein ROM entspricht weitgehend dem des PC-1500. Von ihm aus gesehen, liegen die RAM-Module zwischen 0000 und 3FFF, und das System-RAM zwischen 4000 und 7FFF. Außerdem legt der LH-5803 die Adressen einer anderen Byte-Reihenfolge ab als der Z80. Deshalb werden Adressen im Systemspeicher des reinen BASIC-Interpreters in der Reihenfolge hi-lo-Byte und mit invertiertem b15 gespeichert.

Aufruf eines LH-5803-Programms:

1. Parameter setzen

F002 : 20:Keine Parameter-übergabe
30:Der Inhalt der Register wird übergeben (F005-P00B)
F00C/ : Startadresse (LH-5803-Notation!)
F00E : Bank (00:RPV,01:SPV)

Folgende Parameter können übergeben werden:

F004 : STATUS(T)
F005 : Areg A
F006/ : HLreg X
F008/ : DEreg Y
F00A/ : BCreg U
F00F/ : IXreg
F011/ : IYreg

2. CALL 01C6 des Z80 (CALLH)

Nach Beendigung des LH-5803 kehrt die Kontrolle zum Z80 zurück. Beide Prozessoren können nicht gleichzeitig arbeiten, da sie den gleichen Adreß- und Datenbus benutzen. Einer der beiden Prozessoren befindet sich daher immer im HALT-Modus.

5. ANZEIGE

<<< HARDWARE-KONZEPTION >>>

Die Original-Anzeigenroutinen sind für manche Anwendungen (z.B. Spiele) zu langsam. Deshalb möchte ich zunächst aufzeigen, wie auf die Hardware direkt zugegriffen werden kann.

Der LCD-Bildschirm ist in Abschnitte geteilt:

	BLOCK I x=00..3F	BLOCK II x=40..7F	BLOCK I x=80..BF	BLOCK II x=C0..FF
y=00..07	Zeile 0	Zeile 0	Zeile 4	Zeile 4
y=0B..0F	Zeile 1	Zeile 1	Zeile 5	Zeile 5
y=10..17	Zeile 2	Zeile 2	Zeile 6	Zeile 6
y=18..1F	Zeile 3	Zeile 3	Zeile 7	Zeile 7

Die Bytes ab x=9C (=156d) sind nicht sichtbar.

Die Bytes mit x=FF (Block II, Zeilen 4, 6, 7) steuern die Status Symbole. Sie sind auch im System-RAM gespeichert:

	b7	b6	b5	b4	b3	b2	b1	b0	Zeile
F3C7 : KBI1					S	x	CTRL	BATT	4
F64F :		RUN	PRO	RESERVE		RAD	G	DE	6
F64E : DEF	I	II	III		SML	x	SHIFT	BUSY	7

Der Zugriff auf die Anzeige erfolgt ähnlich wie mit GPRINT bzw. POINT. Zuerst werden Zeile und Spalte angewählt, dann können die Bytes, die jeweils 8 Punkte untereinander darstellen, geschrieben bzw. gelesen werden.

Die Portadressen sind dabei vom Block abhängig:

50-53 : BLOCK I+II 54-57: BLOCK II 58-5B: BLOCK I

50/54/58 : Steuerregister (OUT)
 3E : Anzeige aus
 3F : Anzeige An
 40-7F : Spalte 00-3F anwählen
 B8-BF : Zeile 0-7 anwählen
 C0-FF : Hardware-Scrolling 00-3F

51/55/59 : Busy-Test (IN)
 b7 zurückgesetzt :Anzeigen-Prozessor bereit

52/56/5A : Ausgabe von B-Bit-Daten (OUT)

57/5B : Lesen von 8-Bit-Daten (IN)

Nach dem Setzen von Zeile und Spalte ergibt die erste Lese-Operation noch keine gültigen Daten;es kann erst ab dem zweiten Zugriff gelesen werden.

Beim Lesen bzw. Schreiben von Daten werden die X-Koordinaten automatisch inkrementiert, so daß, wenn die Koordinaten einmal angewählt sind, fortlaufend gelesen bzw. geschrieben werden kann.

Durch das Scrolling verschieben sich die Zeilen. Die absolute Position der Zeile 0 ist nur dann (0,0 ,d.h.links oben), wenn im Port 50 der Wert C0 ausgegeben wurde.

Wurde beispielsweise stattdessen der Wert C8 ausgegeben, so scrollt die Anzeige um 8 Bit nach oben, und an der Position (0,0) steht nun die Zeile 1. Deshalb wird in F05C die Numeer der Zeile die momentan am oberen linken Bildschirmrand steht, zwischengespeichert und bei jedem Scrolling aktualisiert.

Daraus folgt die Formel für die absolute Zeile (hier in BASIC notiert):

OUT &50,((REL_ZEILE+PEEK &F05C) AND &07) OR &B8

Vor jedem OUT-Befehl sollte entweder das Busy-Signal abgefragt werden oder eine Zeit von mindestens 60d Zyklen verstreichen.

Während einer Anzeigen-Routine darf die Interrupt-Routine nicht auf die Anzeige zugreifen Deshalb ist entweder DI zu verwenden oder b1 von F05E zu setzen.

Beispiele:

- Streifenmuster im rechten unteren Eck:

```

F3          DI
3A 5C F0    LD A,(F05C)  Scrollpointer
C6 07       ADD A,07     Zeile 7 wählen
E6 07       AND 07
F6 B8       OR B8
D3 58       OUT (58),A   in Port für den Block I ausgeben
E3          EX (SP),HL   warten
E3          EX (SP),HL
3E 40       LD A,40Spalte 0 anwählen
D3 58       OUT (58),A
06 1C       LD B,1C
3E AA       LD A,AAWert für Streifenmuster
E3          EX (SP),HL
E3          EX (SP),HL
D3 5A       OUT (5A),A   Daten ausgeben
10 FA       DJNZ -06     28d mal wiederholen
FB          EI
    
```

- Block II um 2 Bit nach oben scrollen (Statussymbole werden mit verschoben):

```

F3          DI
3A 5C F0    LD A,(F05C)  Scrollpointer *8
07          RLCA
07          RLCA
07          RLCA
C6 02       ADD A,02      um 2 Bit scrollen
E6 3F       AND 3F        Wertebereich setzen
F6 C0       OR C0
D3 54       OUT (54),A    im Port für den Block II ausgeben
FB          EI
    
```

<<< ROUTINEN >>>

Allgemein gilt:

Tritt in einer Systemroutine ein Fehler auf, so wird vor dem Rücksprung das Carry-Flag gesetzt.

Die meisten Routinen sichern die Register (bis auf das AFreg), so daß deren Inhalt nicht unnötig zerstört wird. Es werden also oft nur register verändert, die einen Returnwert enthalten.

Die Zahl vor der Adresse gibt die Bank an.

```

0,00E5 DSPCTR
    Areg v01 :Display an
    sonst    :Display aus

0,0112 CLS
    Cls,Status-Symbole bleiben

0,0145 ERS1LN
    C1ear Zeile Areg

0,0142 INS1LN
    Zeile Areg einfügen

0,012D UPSCRL
    Scroll vorwärts

0,0130 DWNSCRL
    Scroll rückwärts

0,0109 SETANK
    In 8*6-Matrix schalten, Home, Cursor löschen

0,010C DBLHATR
    ggf. in 16*6-Matrix umschalten, Home, Cursor löschen

0,010F LWIDTH
    Home, Cursor löschen, Areg v01 :40d-Zeichen-Mode
                                Sonst :26d-Zeichen-Mode
    
```

```

0,0133 CGMODE
    Areg v01 :PC-1500-Zeichensatz
    sonst    :IBM-Zeichensatz
0,0136 CGSET80
    Zeichensatz ASC>80 setzen: Bank Areg,adr Dereg
0,020B CGNORM80
    Zeichensatz ASC>80 wieder normal

0,020E STATSYMGET
    Lies Statusanzeige der Zeile Areg (4,6,7) >Areg
0,0211 STATYSSET
    Setze Statusanzeige der Zeile Areg mit dem Wert Breg
0,0139 SMBLREAD
    Lies Statusanz. Nr.Breg >Areg
0,013C SMBLSET
    Setze Statusanz. Nr.Breg mit Areg
    adr  Nr.der Statusanz. Zeile
    F3C7 2          4
    F64F 1          6
    F64E 0          7

0,011E CRSRSTAT
    Setze Cursor-Blinkstatus (F067)
    Areg=00 : Cursor aus
    Areg=01 : Unterstrich
    Areg=02 : BIinkendes Viereck
    Areg=03 : BIinkendes Leerzeichen
0,0208 Blinkcouter (F068) dekrementieren und ggf. Cursor blinken
6,809C CRSRCL
    Cursor löschen
6,80A2 CRSRSTATN
    Setze Cursor wie 011E,aber vorher Cursor nicht löschen
0,0115 CRSRSET
    Setze Printcursor Dereg (Dreg=x,Ereg=y) in F05F/
0,0118 CRSRPOS
    Hole mocentanen Printcursor >Dereg
0,0100 PRTANK
    Prine Zeichen Areg,Cursor (F05F/)
0,0103 PRTDBL
    Print Doppelzeichen Dereg
0,0106 PRTASTR0
    Print String ab (DE) bis 00, nur 8*6-Matrix
0,00EB PRTASTR
    Print (DE) bis zun im Areg gegebenen Endcode,
    nur 8*6 Matrix
0,0163 PRTBSTR0
    Print (DE)-00
0,00F1 PRTBSTR
    Print (DE)-Areg
0,013F ERSSTR
    Areg=00:Cursor Dereg,Breg mal SPACE printen
    sonst :Gcursor (Dereg,HLreg) Breg* SPACE printen
0,011B RVSCHR
    Cursor Dereg,Areg* Zeichen invertiere
    
```

0,0148 GCRSRPOS
Hole nom. Gprint-Cursor aus (F099/),(F09B/)>Dereg,BCreg

0,014B GCRSRSET
Gcursor (Dereg,BCreg)

0,014E PRTGCHR
Gprint Zeichen Areg auf mom. Gcursor

0,0151 PRTGSTRO
Gprint String (DE) bis 00

0,00EE PRTGSTR
Gprint (DE)-Areg

6,809F CHARADR
Pixelcode des Zeichens Aregermitteln,Startadresse>HLreg

0,0154 PRTGPTN
Gprint Byte Areg, Gcursor (F099/),(F09B/)
Set-Code in F096: 00=SET, 01=OR , 02=XOR

0,015A GPTNiEAD
8-Bit-Point (F099/),(F09B/)>Areg

0,0127 DOTSET
Pset ((F08E/),(F090/))
Set-Code in Areg: 00=SET , 01=RES , 02=XOR

0,012A DOTREAD
1-Bit-Point ((F08E/),(F090/))>Areg (0 oder 1)

0,0121 LINE
Line [(F08E/).(F090/))-((F092/)(F094/))
Set-Code (F096) wie Pset , Bitmuster (F097/)

0,0124 BOX
Boxfill,sonst wie Line

0,015D SVELCD
Save Bitmuster von Zeile Areg in (DE)-(DE+9B)

0,0160 LOADLCD
Load Bitmuster von (DE)-(DE+9B, nach Zeile Areg)

0,0157 CPY1500LCD
Kopiere unterste Zeile in den PC-1500-kompatiblen LCD
Speicher F600-F64F,F700-F74F

Arbeitsspeicher:

F05C-F078
F08E-F09C

<<< ZEICHENGGENERATOR >>>

Der PC-1600 besitzt 3 Zeichentabellen, in denen die Daten punktweise in dem Format gespeichert sind, wie es auch beim BASIC-Befehl GPRINT verwendet wird. Pro Zeichen werden 6 Bytes benötigt (8*6-Matrix).

1. Die Tabelle der ASC-Zeichen 20-7F sind in ROM (Bank 6) festgelegt. Die Startadresse der Tabelle kann wie folgt ermittelt werden:

```
3E 20          LD A,20
E7 06 9F 80    CALL 06,809F
```

Nach diesem Aufruf steht sie im HLreg.

2. Die Startadresse der Tabelle von ASC 80-FF liegt im RAM und kann geändert werden. Bank (F066),adr (F064/)
3. Die Zeichen ASC 00-1F sind nicht vorgesehen (blank),ihnen kann aber in (F061/),Bank (F063) eine eigene Tabelle zugeordnet werden, wenn in F05D das b3 gesetzt wird.

6. TASTATUR

<<< TASTATURMATRIX UND TASTATURPUFFER >>>

Die Tasten sind in einem 8*8-Matrix verschaltet. Dabei sind den 9 Spalten das b7-b0 der Port' 1C/1E und das b6 der Ports 1D/1F zugeordnet. Zum Lesen einer Reihe wird in den Datenrichtungs-Ports 1C bzw. 1D das entsprechende Bit gesetzt (OUTPUT-Mode) und in den Ports 1E bzw. 1F das Komplement dieses Werts als Strobe ausgegeben. Im Port 37 kann jetzt der Status der 8-Tasten-Reihe gelesen werden. Die Bits der gedrückten Tasten sind zurückgesetzt. In der Ruhestellung müssen die Bits in den Datenrichtungs-Ports wieder zurückgesetzt werden (INPUT-Mode).

Bits in Port 37:	b7	b6	b5	b4	b3	b2	b1	b0
Port 1D/1F:								
Strobe 8	b6:					BS	KBII	CTRL
Port 1C/1E:								
Strobe 7	b7: Pf_U	B	T	\$	G	9	6	3
Strobe 6	b6: SML	Z	Q	DEF	A	CL	MODE	Pf_R
Strobe 5	b5: SPACE	V	R	#	F	P	Pf_L	=
Strobe 4	b4: RCL	C	E	"	D	/	*	+
Strobe 3	b3: ENTER	(	I	&	K	O	L	)
Strobe 2	b2: 0	M	U	X	J	7	4	1
Strobe 1	b1: D_Pf	X	W	!	S	OFF	-	.
Strobe 0	b0: Pf_O	N	Y	SHIFT	H	8	5	2

während der Abfrage darf kein Tasteninterrupt auftreten. Deshalb ist entweder DI zu verwenden oder b0 von F079 zu setzen.

Die BRK-Taste hat eine Sonderstellung:

Wird momentan die BRK- (ON-) Taste gedrückt, so ist in Port 1F das b7 gesetzt. Zusätzlich speichert b1 des Ports 1B dieses Ereignis so lange, bis es zurückgesetzt wird.

Wird bei der zyklischen Abfrage der Tastatur (durch 1/64-sek Interrupt) eine Taste betätigt, so wird deren Code in einem 64d Bytes großen Puffer (First in-First out-Struktur) abgelegt und kann bei Bedarf gelesen werden. Der Puffer steht in F0DF-F11E. Die Nummer des ersten gültigen Wertes steht in F080 (Lesepointer). Sie wird zu F0DF dazugezählt und ergibt so die Adresse des ersten zu lesenden Bytes. Der Schreibpointer steht in F07F.

Sind beide Pointer gleich, dann ist entweder der Puffer leer, oder, falls b7 von F07F gesetzt ist, ist er mit 64d Bytes gefüllt. Während des Zugriffs auf den Puffer muß b3 von F07A gesetzt sein.

<<< ROUTINEN >>>

- 0,0178 KEYSTRB
Liest den Strobe Areg (08) der Tastaturmatrix
Die Bedeutung der Bits in den einzelnen Strobes siehe
oben. Nur der Returnwert von Strobe 8 ist um 1 Bit nach
links geschoben, so daß in b0 das Bit für die BRK-Taste
Platz findet. Ist eine der 8 Tasten gedrückt, wird das
entsprechende Bit im >Areg zurückgesetzt.
- 0,0172 CURUDCHK
Test der Cursortasten
>nc :Keine Taste
>Areg v80:Cursor down
>Areg v40:Cursor up
- 0,0184 OFFCHK
>sc :OFF-Taste gedrückt
- 0,016F BREAKCHK
Falls BRK-Taste gedrückt wurde:Tastaturpuffer löschen
und >sc
- 0,018A BREAKRESET
Lösche Tastaturpuffer und BRK-Taste
- 0,016C KBUFSET
Kbuff\$:Lösche den Tastenpuffer und lese ab der Startadr
die in Dereg gegeben ist,Areg * Zeichen ein
- 0,017B KEYAUX
Keystat Setzt die erwartete Herkunft einer Eingabe
Areg =00: Nur Tastatur
Areg v01:Zusätzlich Sub-CPU (Timer)
Areg v02:Zusätzlich RS-232C-Schnittstelle
- 0,017E KEYSTATSET
Setze Wiederholfunktion unt CClick,Code in Areg
b2:Repeat für alle Tasten
b1:Repeat für tie meisten Tasten
b0:Tasten-Click
- 0,0181 KEYSTATREAD
Lies Keystat >Areg
b4 :Keystat 2
b3 :Keystat 1
b2-b0:Wie bei 017E
- 3,4051 KEYSTRBSCAN
Frage die momentan gedrückte Taste ab >Areg
keine Taste: >Areg=00
mehrere Tasten gleichzeitig gedrückt: >sc
- 0,0175 KEYDIRECT
Wie vor, aber außer der Matrix werden auch die durch
Keystat selektierten Quellen abgefragt

0,0166 KEYGET
Lies 1 Zeichen aus dem Puffer >Areg
oder warte, bis eine Taste gedrückt wird

0,0169 KEYGETR
Inkey\$, Taste aus dem Puffer >Areg
>sc , falls kein Wert vorhanden

0,0187 KEYGETND
Wie vor, jedoch bleibt der Inhalt des Puffers erhalten

6,80B7 KEYSKAN
Routine für Tasteninterrupt:
Fragt die Tasten ab und speichert sie im Puffer
Achtung: Bank muß mit LD A,66 ;OUT (31),A selektiert sein

0,0217 KEYNORM
Setzt die Standard-Tastentabellen

3,4054 KEYCODCNV
Konvertiert den Tastencode in Areg je nach der Stellung
der SHIFT-, SML-, DEF-, CTRL- und KB11-Umschalter >Areg

Arbeitsbereich:

F079-F08C
F0DF-F121

<<< TASTENTABELLEN >>>

Die Tabelle zur Umrechnung der Matrix in den ASCII-Code steht in Bank (F11F), adr (F120/). Der aus dieser Tabelle resultierende Code wird im Tastaturpuffer gespeichert. Die Tabelle beginnt mit den Codes der BRK-Taste. Es folgen die Tasten von Strobe 8 bis Strobe 0, jeweils von b0 bis b7.

Bei der Entnahme der Codes aus dem Puffer werden sie mit der Routine KEYCODCNV (3,4054) umgerechnet. Dazu gibt es 3 Tabellen:

Tabelle für: Bank: Startadresse:

SHIFT	(F086)	(F084/)
KBII	(F089)	(F087/)
SHIFT+KBII	(F08C)	(FOBA/)

Vom ursprünglichen Code wird 08 abgezogen und als Index zur Startadresse addiert. Der Inhalt dieser neuen Adresse ist der neue, durch die Umschalttasten erzeugte, ASCII-Code. Die Tabellen gelten von Code 08 (= Pf_L) bis 5A (= Z).

7.BUZZER

Der eingebaute miniaturlautsprecher wird von mehreren Geräten benutzt:

- Timer (Click Alarm und Wake Funktionen)
- Beep Kommando
- Cassettenrecorder - Interface

Zur Erzeugung eines Tones wird b7 des Port 18 mit der gewünschten Frequenz ab und angeschaltet.

Wird b6 von Port 18 zurückgesetzt ist der Buzzer blockiert.

Übrigens können auch die durch das Beep erzeugten Töne mit dem Cassetenrecorder Interface aufgezeichnet werden.

Während der Tonerzeugung sollte der Interrupt verboten werden, da der Ton sonst durch die Unterbrechungen verfälscht wird.

<<< ROUTINEN >>>

01B7 SOUT

l*Beep,Tonhöhe in Areg,Länge in BReg (wie in BASIC)

>sc wenn die BRK gedrückt wird

Frequenz in Hz: $1300000d / (22d * Areg + 166d)$

Dauer in sek : BReg/Frequenz

01B4 BOUT

DReg * Beep, sonst wie oben

01BD BONOFF

Beep on/off, je nach Stand des Beep-off-Bits b0 von F86B

01C0 BON

Beep on

01C3 BUZON

Buzzer an, aber Beep bleibt

01BA SWAIT

Wait BReg (in $1/64$ - sek -Einheiten)

>sc wenn die BRK Taste gedrückt wird

8.TIMER,A/D Wandler

<<< AUFGABEN >>>

Die Sub-CPU mit eingebautem Timer und A/D-Wandler hat folgende Funktionen:

- Steuerung des Resetablaufs des Z80
- Interruptsteuerung
- Echtzeitfunktionen
- Tastenclick, Stundensignal
- Passwort
- A/D Wandler
- Batterietest für interne und externe Stromversorgung

<<< ROUTINEN >>>

Alle Timer-Routinen haben die Startadresse 01D5.
Der entsprechende Funktionscode ist im Creg zu setzen.

```
c00 SINIT
    Areg=00 : Totalreset des Timers
    Areg=01 : Reset nach Power off,Nornalreset
    Areg=02 : Reset nach Power Aoff
c01 SBEEP
    Tastenclick

c02 SWRT
    Time setzen, (HL) : Mon,Tag,Std,Min,Sek
    *(HL+00)          : Monat binär (0..C)
    *(HL+01)-(HL+4)   : Angaben in BCD System
    *Ist ein Nibble=0F (alle 4 Bits gesetzt),
    so bleibt der alte Wert erhalten

c03 SRRT
    Time holen        >(HL)-(HL+04)
c04 SWWT
    Wake$(0) setzen, (HL):Mon,Tag,Std,Min
c05 SRNT
    Wake$(0) holen    >(HL)-(HL+03)
c06 SHA1T
    On Time$ setzen, wie Wake$(0)
c07 SRAIT
    On Time$, holen   >wie Wake$(0)
c08 SHA2T
    Alarm$ setzen, wie Wake$(0)
c09 SRA2T
    Alarms$ holen     >wie Wake$(0)
```

```
c0A SWPASS
    Password setzen, (DE)-(DE+07)
c0B SPASSCHK
    Password vergleichen (DE)-(DE+07) und ggf. löschen, sonst >sc

c10 SWMSK
    Setze Interrupt-Maske Areg, die Bits der erlaubten Interrupts
    müssen gesetzt sein:
        b7 : Wake$(0)
        b6 : On Time$
        b5 : Alarm$
        b1 : 1.0-Sek-Interrupt
        b0 : 0.5-Sek-Interrupt
c11 SRMSK
    Interrupt-Maske lesen >Areg
c12 SRIRQ
    Interrupt-Ursache lesen >Areg

c13 SRINP
    Status lesen >Areg
        b5 : CI-Signal
        b2 : PASS gesetzt
        b1 : Adin im gewählten Bereich (Range)

c14 SUPON
    Setze Power-On-Conditions Areg
        b3 : Stundensignal an
        b2 : Beep beim automatischen Einschalten (Wake$)
        b1 : Einschalten durch Wake$(0) erlaubt
        b0 : Einschalten durch Wake$(1) erlaubt

c16 SKEYCHK
    Test, ob Keystat 1 möglich ist ja:>nc nein:>sc
c17 SKEYGET
    Eingabe von Keystat 1 >Areg

c18 SRA0
    Spannung der internen Stromversorgung >Areg
    Areg<AF: leer Areg>BE: voll
c19 SRAI
    AIN >Areg
c1A SRA2
    Spannung der externen Stromversorgung >Areg
    Areg<A8: Ni-Cd-Batterie leer
```

Die mit dem A/D-Wandler gelesenen Werte schwanken sehr stark. Sie sollten am besten mehrmals gelesen und ein Mittelwert gebildet werden.

```
c21 SRPON
    Power-On-conditions lesen >Areg
c22 SWAB
    Setze Alarm-Beep-Bedingung Areg
    b1 : Alarm, -Beep an
    b0 : On Time$-Beep an
c23 SRAB
    Alarm-Beep-Bedingung lesen >Are8
c24 SWA1A
    Setze On Adin-Range
    Lreg:Untergrenze
    Hreg:Obergrenze
```

<<< TIMER-RAM >>>

Mit ein paar Tricks läßt sich ein wichtiger Teil des Timer-RAMs ins normale RAM einlesen bzw. wieder herausschreiben. Dazu müssen Sie erst 3 Adressen im ROM ihres Rechners finden.

Suchen Sie auf Bank 6 ab 8000 folgende Bytefolgen:

- 1.) 79 C6 90 C3
- 2.) CD xx xx ED 6F CD xx xx ED 6F
- 3.) 3E 80 ED 6F CD

Bauen Sie die Startadressen der Bytefolgen in dieses Programm ein:

```
F3      DI
21 xx xx LD HL,gewünschte schreib-/Leseadr. im RAM
01 80 38 LD BC, 3880
3E 60    LD A,60
D3 31    OUT (31),A
CD xx xx CALL adr 1
CD xx xx CALL adr 2 zum Lesen, adr 3 zum Schreiben
3E 5F    LD A,5F
D3 35    OUT (35),A
FB       EI
C9       RET
```

Für zwei ROM-Versionen kann ich die Adressen angeben:

- 1.) A8C6 A87B
- 2.) A9EC A9A1
- 3.) AA1B A9D0

Beim Aufruf des Programms werden ab (HL) 38 Bytes gelesen bzw. geschrieben.

Einige Interessante Adressen:

(HL+00)-(HL+07) : Password
(HL+16)-(HL+17) : Adin Range (lo/hi)
(HL+18)-(HL+1B) : Wake\$(0)
(HL+20)-(HL+23) : On Time\$
(HL+23)-(HL+26) : Alarm\$
(HL+27) : Interrupt-Maske
(HL+28)-(HL+2C) : Time
(HL+2C),b0 : Power-On-Conditions
(HL+30)-(HL+37) : Pass-Vergleichspuffer

9.SERIELLE SCHNITTSTELLEN

<<< HARDWARE >>>

Für die seriellen Schnittstellen ist der UART-Baustein (Port-adressen 20-23) zuständig.

Port 20 : Ein- und Ausgabe

Port 22 : - Parameter setzen, wenn im Port 23 b7 und b6 gesetzt sind

z.B. Baudrate setzen:

OUT 23,C0

OUT 22,lo-Byte von (76800d/Baudrate)

OUT 23,C1

OUT 22,hl-Byte

- Status lesen

b2 : COM1 Ready

b1 : Zeichen e-pfangen

b0 : COM2 Ready

Port 23 : - Befehle und Registeradressen ausgeben (siehe Port 22)

OUT 23,B4 : COM1 anwählen

OUT 23,B5 : COM2 anwählen

- Handshaking-Signale setzen

b5 : RTS (Invertiert)

b3 : 1st bei SNDB8K gesetzt

b1 : DTR (invertiert)

- Status lesen (außer CI)

b4 : COM1 gewählt

b2 : DSR

b1 : CD

b0 : CTS (Invertiert)

Der Zustand des CI-Eingangs kann mit der
Timer-Routine SRINP (c13) gelesen werden >Areg:

b5 : C1 (Invertiert)

<<< ROUTINEN >>>

Aufruf mit CALL 01D8, Funktionscode im Creg. Dreg enthält den
Code der gewünschten Schnittstelle;

Dreg=00 : "COM: " (momentan mit Setdev oder mit c12 selektierte
Schnittstelle)

Dreg=01 : "COM1:"

Dreg=02 : "COM2:"

c00 CRESET
Reset, Parameter im Areg wie beim Timer

c01 CWCOR
Setcom (HL)
(HL/) : Zeiteinheit 76800d /Baudrate (2..600)
(HL+02) : b7 : SHIFT IN, sonst SHIFT OUT
 b6 : XON, sonst XOFF
 b5 : PARITY EVEN, sonst PARITY ODD
 b4 : Parity an (PO/PE), sonst NO PARITY
 b3 : 0 0 1 1
 b2 : 0 1 0 1
 Wortlänge 5 6 7 8 Bit
 b1 : Immer 0
 b0 : 2 Stopbits, sonst 1

c02 CRCOM
Com\$ (Umkehrung von Setcom), Startadr >Dereg

c03 CSNDA
Send Areg
>sc : Fehler, in Areg codiert: b7 : BRK gedrückt
 b1 : Timeout

c04 CRCVA
Empfange 1 Byte >Areg, wartet bei leerem Empfangspuffer
>sz, wenn 1A (=End of Text) empfangen

c05 CRCVERR
Liest Empfangsfehler >Areg und löscht ihn
Codierung >Areg wie in c03

c06 CWIDTH
Setze Pconsole Zellenlänge Areg

c07 CRXD
Rxd\$ >Areg
>sz, wenn keine Daten vorliegen

c08 CNUMRCV
Zahl der im Puffer empfangenen Zeichen >Dereg

c0A CNUMFRE
Zahl der freien Bytes in Puffer >Dereg

c0B CSETEOL
Setze Pconsole EOL-Code Areg

c0C CZONE
Setze Pzone Areg (08..FF)

c0D CSNDEOL
Send EOL
>sc bei Fehler, Codierung >Areg wie in c03

c0E CSETHS
RTS und DTR high, evtl. XON (=11) senden

c0F CRESHS
RTS und DTR low

c10 CWOUTS
Outstat Ereg
b7 : b1 und b0 gültig, sonst Auto-Handshake
b1 : RTS low
b0 : DTR low

c11 CRINS
Instat >Areg
b5 : CI low
b4 : DSX low
b3 : CD low
b2 : CTS low
b1 : RTS low
b0 : DTR low

c12 CSETDEV
Setdev Areg
b6 : "COM2:", sonst "COM1:"
b2 : PO
b0 : KI

c13 CRDEV
Liest Setdev-Einstellung >Areg
b7: Kanal offen, sonst geschlossen
andere Bits wie in c12

c14 CESND
Sndstat Ereg: Die Bits der für die Übertragung relevanten
Signale sind zurückzusetzen
b4 : DSR nicht relevant
b3 : CD ---, ---
b2 : CTS ---, ---
Breg: Timeout-Wert in 0.5-sek-Einheiten
(00=unendlich)

c15 CERCV
Rcvstat Ereg, Breg: Bedeutung wie c14

c16 CSBRK
Sndbrk Breg mal senden
>Areg=00: Fertig gesendet
>Areg=01: Hit BRK-Taste abgebrochen

c17 CSRCVB
Init Empfangspuffer-Größe HLreg (0000 oder 0050-3FFF)
HLreg=0000: Standardpuffer mit 40d Bytes
>Areg=00: ok, sonst: Fehler

c1B CCLRSB
Sendestart

c1C CCLRRB
Empfangsstart (Puffer und Fehlercode löschen)


```
c1D CREOL
    EOL-Code lesen >Areg
c1E CRZONE
    Pzone lesen >Areg
c1F CRHIDTH
    Pconsole Zeilenlänge lesen >Areg
```

10.CENTRONICS-INTERFACE

<<< KONZEPTION >>>

Mit den Parallel-Interface CE-1600E kann der PC-1600 mit einem Centronics-Drucker betrieben werden. Die BASIC-Befehle entsprechen dabei denen des Plotters in Text-Mode. Als Device Name ist "LPT2:" vorgesehen.

Die Diskettenstation kann wie üblich angeschlossen und verwendet werden. Ein Cassettenrecorder-Interface ist nicht vorhanden.

Die ROM-Module des Parallel-Interfaces befinden sich auf Bank 4, adr 6000-7FFF, die der Floppy auf Bank 5, adr 4000-5FFF. Zur Ein- und Ausgabe werden die Ports 80 bis 83 verwendet.

Die BASIC-Befehle besitzen Einsprungstellen für Eigenerweiterungen. Wenn b7 von FE78 gesetzt ist, wird vor jeden dieser Befehle ein selbstdefiniertes Unterprogramm aufgerufen, dessen Adresse in FE7A/ zu schreiben ist. Als Rücksprungsadresse befinden sich je nach Befehl folgende Werte in Stack:

```
6026   PCONSOLE
6029   PZONE
602C   INIT
602F   LF
6032   TAB
6035   LPRINT
6038   LLIST
```

<<< ROUTINEN >>>

Die Systemroutinen werden mit CALL 6008 auf Bank 4 aufgerufen, Funktionscode im Creg:

```
c00 PAOUT
    Sende (Steuercode)Creg

c01 PASTROUT
    Sende Breg Codes ab (DE)

c02 EXPRM
    Sende Impuls zum Hardware-Reset des Druckers

c03 PAPRT
    Drucke Breg Codes ab (DE) mit Berücksichtigung des Tabulator3
    und der PCONSOLE-Einstellungen

c04 PAXON
    Sende XON-Code (11) zur Kommunikationssteuerung
```

<<< ZEICHENSATZ >>>

Ist in FE74 der Wert 00 gespeichert, so wird der Standard-IBM-Zeichensatz verwendet. Die ASCII-Codes von 80-FF können jedoch bei Bedarf auf zwei Arten uncodiert werden:

- In FE75/ wird die Anfangsadresse einer 80 Bytes langen Tabelle abgelegt, in der der Reihe nach jeden Code ein neuer zugeordnet wird. Dazu muß im n1 von FE74 die Bank gespeichert und zusätzlich das b7 gesetzt sein.
- Im ROM sind 15d Codierungstabellen gespeichert, deren Nummer (01-0F) in FE74 angegeben werden kann.

11.PLOTTER**<<< SCHRITTMOTOR-STEUERUNG >>>**

Der Plotter CE-1600P besitzt drei Schrittmotoren, deren Magnete einzeln angesteuert werden können:

1. x-Motor (Stifthalterung links/rechts)
2. y-Motor (Walze vor/zurück)
3. Stiftmotor (Farben wechseln, Stift heben und senken)

Für die 4 Magneten eines Motors ist je eine Leitung (1 Bit) zuständig. Diese werden so beschaltet, daß sich eine Kreisbahn des Ankers ergibt.

0=Magnet aus, X=Magnet an, die Zahl in der Mitte des Kreises gibt den hexadezimalen Code an:

Ruhestellung:	0		bo	
	0 0 0		bl	b3
	0		b2	
Drehung:	0) X	1) 0	2) 0	3) 0
	0 9 X	0 8 X	0 C X	0 4 0
	0	0	X	X
	4) 0	5) 0	6) X	7) X
	X 6 0	X 2 0	X 3 0	0 1 0
	X	0	0	0

Jede Achteldrehung des Schrittmotors entspricht einer Bewegung des Papiers um 1/10 Millimeter.

		y-Motor	x-Motor
Momentaner Motorstand	(0,1,2,3,4,5,6,7):	(F9F1) Nibble1	n0
Wert in Port 83	(9,8,C,4,6,2,3,1):	(F9F2) n1	n0
Stabilisierungs-Counter	(6)	(F9F0) n1	n0
Bewegungsrichtung bei Erhöhung des Motorstand-Counters	:	zurück	rechts

Zwischen den Achteldrehungen, also den Herausschreiben der entsprechenden Werte in Port 83, sollte eine Zeit von ca.4700d Zyklen verstreichen, die durch folgende Schleife erreicht werden kann:

```

06 F9      LDB,F9
0A         LDA,(BC)
10 FD      DJNZ -03

```

Der Interrupt sollte gesperrt werden.

Um ein Nachdrehen zu verhindern, sollte an Schluß 6* der gleiche Wert zur Stabilisierung gesendet werden.

Danach sind die Magneten mit XOR A ; OUT (83),A abzuschalten, da sie viel Strom verbrauchen.

Entsprechendes gilt auch für den Stiftmotor, jedoch werden bei ihm nur 4 Stellungen angefahren:

Momentaner Motorstand	(0,1,2,3)	:(F9F3) n0
Wert in Fort 82	(9,C,6,3)	:Port82 n0
Stabilisierungs-Counter		:(F9F3) n1

Die elementare Routine zur Steuerung der x- und y-Motoren steht auf Bank 4, adr 50B1.

(Ermittlung der Startadr bei anderen ROM-Versionen:

Sprungzieladr des Sprungs bei 4008 +004D)

Die Motoren werden. einzeln oder auch gleichzeitig, um eine Achfeldrehung bewegt. Falls der Druckkopf außerhalb der erlaubten Grenzen gerät, wird die Bewegung nicht ausgeführt. Soll diese Prüfung umgangen werden, so kann auch ab 5292 gestartet werden.

Als Parameter wird in Creg der Code der Bewegung mitgegeben:

b0 Bewegung in x-Richtung
 b1=0 nach links
 b1=1 nach rechts
b4 Bewegung in y-Richtung
 b5=0 vorwärts
 b5=1 rückwärts

<<< ROUTINEN >>>

Alle Routinen liegen auf Bank 4.

Bei einem Fehler wird das Carry-Flag gesetzt; der Fehlercode in Areg ist derselbe wie in BASIC.

Nach dem Aufruf einer Routine sollten folgende Befehle ausgeführt werden:

CALL 4029 schaltet die Schrittmotor-Magneten aus
CALL 6777 schaltet die Interrupt-Tasten an und setzt die Interrupt-Maske in Port 35

4020 PCHEK
 Plotterstatus
 >Areg:
 b4 Hardware nicht zurückgesetzt
 b2 Interrupt-Tasten gesperrt
 b1 Stiftwechsel-Modus
 b0 Batterie leer
4023 POUT
 Drucke Zeichen Ereg
4029 POFF
 Schrittmotoren aus
402C PSTROUT
 Drucke Breg* Zeichen ab (HL)

Die übrigen Routinen werden mit CALL 4008 aufgerufen, Funktionscode im Creg. Es ist zu beachten, daß viele Funktionen (wie auch in BASIC) entweder nur in Text- oder in Graph-Modus erlaubt sind. Die Fehlercodes sind wie in BASIC.

```

c00 PINIT
    Reset Drucker-RAM
c01 PTEXT
    Text-Modus
c02 PGRAPH
    Graph-Modus

c03 PCSIZE
    Csize Areg
c04 PCOLOR
    Color Areg

c06 PWIDTH
    Pconsole Zeilenlänge Areg
c07 PLEFTM
    Pconsole Heftrand
Areg c08 PPITCH
    Pitch (HL): x (HL+01): y (04..FF, oder 00:Standardwert)
c09 PPAPER
    Areg=00: Paper C (Einzelblatt), sonst: Paper R
c0A PSCRL
    Paper (HL/): Limit 1, (HL+02/): Limit 2 (0000:Standard)
c0B PEOL
    Pconsole EOL-Code Areg
    00: CR 01: LF 02: CR+LF
c0C PZONE
    Pzone Areg

C0D PPENUP
    Areg=FF: Stift senken (vor), sonst: Stift heben
c0E PROTATE
    Rotate Areg
c0F PLTYPE
    Lline-Typ Areg

c11 PSORGN
    Sorgn
c12 PAMVUP
    Gcursor (HL/), (HL+02/)
    Graphik-Koordinaten: -2048d bis 2047d (Zweierkomplement)
    Breg; Anzahl der xy-Paare ab (HL) (01-06. normalerweise 01)
c13 PRMVUP
    Relativer Gcursor ab momentaner Stiftposition
    sonst wie c12
c14 PAMVDN
    Lline (HL/), (HL+02/)-(HL+04/), (HL+06/)-... Breg mal
c15 PRMVDK
    Rline, sonst wie c14
    
```

```

c16 PTEST
    Test
c17 PTAB
    Lcursor Areg
c18 ALLOFF
    Schrittmotoren aus
c19 PMYRIGHT
    Zahl der noch druckbaren Zeichen ab Stiftpostion bis zum
    rechten Rand >Areg
c1A PCUP
    LF-Areg                Falls das Kommando nicht voll-
c1B PCDOWN
    LF+Areg                ständig ausgeführt werden konnte:
c1C PCLEFT
    Areg* nach links       Zahl der restlichen Zeichen
c1D PCRIGHT
    Areg* nach rechts      >Breg

c1E PGUP
    HLreg* rückwärts (in 0.2mm-Schritt)
c1F PGDOWN
    HLreg* vorwärts
c20 PGLEFT
    HLreg* links
c21 PGRIGHT
    HLreg* rechts

c22 PCHGPEN
    Areg=00: Stiftwechsel an,sonst: Stiftwechsel Ende
c28 PRESET
    Totalreset Drucker-RAM

c29 PCR
    CR (Druckkopf nach links)
c2A PDIRC
    Schreibrichtung Areg
    00: nach rechts 01: unten 02: links 03: oben
c2B PCRLF
    CR+LF

c2C PMYFD
    Anzahl der möglichen LF- >HLreg
    (FFFF:Bei Rollenpapier unendlich)
c2D PPYFD
    Anzahl der möglichen LF+ >HLreg
c2E PGETMOD
    Check Mode >Areg
    00: Text 01: Graph

c2F PBOXA
    Boxline (HL/),(HL+02/)-(HL+04/),(HL+06/)
c30 PBOXR
    Relative Boxline,sonst wie c2F
    
```

c31 HARESET
Hardware-Reset (Druckkopf ganz links, Color 0)
c32 PRESULT
Falls Drucker-PRINTSchalter in P-Stellung:
Areg=00 Drucke Inhalt des BASIC-Puffers (ab FBB0)
Are&=01 Drucke letztes Ergebnis
c34 PPCHR
Drucke ein Zeichen, dessen Code ab (HL) abgelegt ist, Codelänge in Breg.

<<< ZEICHENCODIERUNG >>>

1.Code: Startposition des Zeichens ab der momentanen Stiftposition
nl: x n0: y links unten=(0,0)

Nächste Codes: Linien bis zu einen bestimmten x- (oder y-)Wert
b6-b4 geben die Richtung an:

nl=0: waagerecht,	bis x=n0
nl=1: senkrecht,	bis y=n0
nl=2: schräg nach unten	bis x=n0
nl=3: schräg nach oben	bis x=n0
nl=4: flach schräg nach oben,	bis x=n0
nl=6: steil schräg nach oben,	bis x=n0
nl v80: Nächster Code ist wieder ein xy-Wert,	der mit gehobenen Stift angefahren wird

Sondercodes: F0 Ende des Zeichens, Stift fährt zur Position des nächsten Zeichens

F3 Backspace

F8 Relativer Versatz nach links unten (Unterlängen)

Nächster Code: nl: Versatz in -x-Richtung

n0: Versatz in -y-Richtung

Die Größe des Ausdrucks ist von Czize abhängig.

Die Position des nächsten Zeichens ist von Rotate und der Schreibrichtung abhängig.

Die Codierungen der ASCII-Zeichen befinden sich fest im Plotter-ROM und können nicht geändert werden:

Die Routine 6474 ermittelt aus den ASC-Code im Areg die Adresse der internen Drucker Codierung >HLreg, Länge >Breg.

Die Codetabelle steht ab 4166 und ist in Blöcke von 0012 Bytes aufgeteilt, aus denen sich die Startadressen für je 16d ASCII-Zeichen ermitteln lassen:

ASC 20-FF

4166 F2 42 Startadr des Codes 20 (Space): 42F2
4168 01 Länge von 'Space': 01
Damit ergibt sich die Startadr des Codes 21 als 42F3.

4169 0A Länge von 0A

usw.

ASC 30-3F

4178 55 43 Startadr des Codes 30: (0) : 4355

417A 0A Länge von "0" : 0A

usw.

Japanische Zeichen. (ASC 80-FF):

Beginn der Blöcke bei 4262

<<< INTERRUPT-TASTEN >>>

Die Tasten werden beim ROM-Interrupt 0 aus dem Port 81 gelesen:

b2 Papier rückwärts

b1 Papier vorwärts

b0 Stiftwechsel

Durch CALL 0175 wird der Zustand der Shift-Taste gelesen.

In Port 80 können die Bit3, die den Druckertasten entsprechen, zurückgesetzt und damit einzelne Tasten maskiert werden.

Außerdem lassen sich die Tasteninterrupts durch Setzen von b5 der Adresse F9EE sperren.

Das leidige in-die-Mitte-fahren des Druckkopfes beim Drücken der Paperfeed-Taste kann durch Setzen von b5 der Adresse F9F8 unterdrückt werden.

12. CASSETTENRECORDER-INTERFACE**<<< HARDWARE >>>**

Das Remonte-Relais wird mit b4 des Ports 82 ein- und mit b5 ausgeschaltet. Bevor wieder in Ruhestellung (^CF) geschaltet wird, sollte folgende Warteschleife durchlaufen werden:

```

06 11    LD B,11
3D       DEC A
20 FD    JR NZ,-03
10 FB    DJNZ -05

```

Das Senden von Daten erfolgt über die Buzzer-Leitung (b7 von Port 18). Das so erzeugte Rechtecksignal wird in Interface, das im Plotter eingebaut ist, in ein analoges Signal umgewandelt.

Umgekehrt wird zum Empfang der Daten das Analog-Signal wieder in einen Rechteckimpuls zurückverwandelt. Dazu muß das Interface durch Setzen von b7 im Port 82 angeschaltet werden. Jetzt liegt der Impuls an b2 des Ports 1F an.

Alle folgenden Ausführungen gelten nur für den MODE 0, da ich die Emulation des PC-1500-Aufzeichnungsformates für uninteressant halte.

Die Bits werden frequenzmoduliert.

Die "0" wird als kurzer High-Low-Impuls (SHORT, Länge 0.327 msec), die "1" als langer High-Low-Impuls (LONG, 818 msec) übertragen.

Die Dauer, der Impulse kann geändert werden, da sie im System-RAM gespeichert sind:

adr	SHORT/LONG	Dauer in Millisekunden	Standardwert
F197	"0" high	(44d + 18d * X) / 3580d	1E
F198	"0" low	(172d + 18d * X) / 3580d	17
F199	"1" high	(44d + 18d * X) / 3580d	4F
F19A	"1" low	(187d + 18d * X) / 3580d	47

Beim Lesen wird die Impulslänge gemessen. Ist sie größer als in (F1A8) angegeben (Standard=16), so wird sie als LONG-Byte (= "1") interpretiert.

Wenn b7 von (F1A9) zurückgesetzt ist, wird dabei der High-Low-Übergang ausgewertet, ansonsten umgekehrt. Durch diese beiden Möglichkeiten ist die Wahrscheinlichkeit größer, die Daten korrekt lesen zu können (siehe BASIC-Handbuch Seite 6-24).

Reihenfolge der Bits in einem Byte:

Startbit (LONG="1"), b7, b6, b5, ... b0

<<< LOGISCHES AUFZEICHNUNGSFORMAT >>>

-Headerblock (Aufzeichnungskopf)

1. Relais an
2. Pause 1,Länge in Zehntelsekunden=50 (F19D)
3. Vorspann 1 (SHORT),Anzahl=2710 (F19F/)
4. Vorspann 2 (LONG) ,Anzahl=28 (F1A1)
5. Vorspann 3 (SHORT),Anzahl=28 (F1A2)
6. 1*LONG
7. Header (30 Bytes):
 - +00 Mode b3 Daten (PRINT#)
 - b2 ASCII
 - b1 BASIC-Programm (binär gespeichert)
 - b0 Maschinenprogramm
 - +01-0D : Name
 - +0E-10 : Extension
 - +11 : 0D
 - +12/ : Dateilänge mod 65536d (nicht bei ASCII-Dateien)
 - +14/ : Startadresse Maschinenprogramm
 - +16/ : Autostart-Adresse oder FFFF
 - +18 : Datenformat für MODE 1
 - +19 : Monat
 - +1A : Tag (BCD codiert)
 - +1B : Stunde (----,----)
 - +1C : Minute (----,----)
 - +1D Dateilänge\65536d
 - +1E Bank Startadresse
 - +1F Bank Autostart oder FF
 - +20-2F: 00
8. Checksumme aller Datenbits des Headers (2 Byte)
9. 1*LONG
- [*]
10. Pause 2,Länge in Zehntelsekunden=14(F19E)
11. Relais aus

- Datenblock

1. Relais an
2. Vorspann 1 (SHORT),Anzahl=2AF8 (F1A3/)
3. Vorspann 2 (LONG) ,Anzahl=14 (F1A5)
4. Vorspann 3 (SHORT),Anzahl=14 (F1A6)
5. 1*LONG
6. Daten
7. Checksumme aller Datenbits (2 Byte)
8. 1*LONG
- [*]
10. Pause Länge in Zehntelsekunden=28 (F1A7)
11. Relais aus

[*]: Falls b0 von FlAE gesetzt ist, werden die Nutzdaten doppelt abgespeichert (höhere Datensicherheit).

Format dieser redundanten Speicherung:

1. Vorspann 4 (SHORT), Anzahl=100 (FlAF) 00 entspricht 256d
2. Nochmal Header bzw. Daten
3. Checksumme der doppelten Nutzdaten (2 Byte)
4. 1*LONG

Bei ASCII-Dateien werden die Daten in Blöcke von 100 Bytes unterteilt. Das Ende der Datei ist durch 1A (=CTRL-Z) markiert.

<<< ROUTINEN >>>

Aufruf mit CALL 6008 der Bank 5, Funktionscode im Creg

>sc: Fehler, Code >Areg
Areg=00: BRK-Taste
Areg=01: Checksummenfehler
Areg=02: Verifikationsfehler
Areg=03: Lesefehler

c02 RMTON
Relais

an c03

RMTOFF
Relais aus

c04 HEADERSND
Sende Kopf in FB60-FB8F

c05 HEADERCV
Lade Kopf nach F21D-F24C

Bei den folgenden Routinen sind Bank, Startadr und Länge block-

weise ab FB98 anzugeben:

FB98 16d*Bank I

FB99/ Startadr I

FB9B/ Länge 1

FB9D 16d*Bank II, oder: 80=Ende

FB9E Startadr II

c06 DATASND
Sende Daten ab [FB98]

c07 DATARCV
Lade Daten ab [FB98]

c06 CVERIFY
Vergleiche Daten ab [FB98]

13.DISKETTENLAUFWERK

<<< ROUTINEN >>>

Aufruf der Routinen mit CALL 4008 der Bank 4,Funktionscode in Creg
Areg:Laufwerknnummer (01="X",02="Y")

>sc,wenn,Fehler: Code in Areg

b7 :Keine Diskette im Laufwerk
b6 :Batterie leer
b5 :Schreibschutz gesetzt (WP)
b4 :Spur nicht gefunden
b3 :Sektor nicht gefunden (nicht formatiert)
b2 :Dateiende überschritten
b1 :Schreib- oder Lesefehler
b0 :Hardware-Fehler
Areg=00: Verifikationsfehler

c80 DINIT

Disk-Initialisierung und Test

c81 CNCTDRV

Porttest,Zahl der angeschlossenen Laufwerke >Areg

c82 RESTORE

Lesekopf auf Spur 0,evtl "Set diskette for"-Meldung

c83 FORMAT

Formatieren

c84 DREAD

Lese absolute Sektoren >(HL)

Breg: Zahl der Sektoren =Bytezahl*200

Dreg: Nummer der ersten Spur (00..0F)

Ereg: Nummer des erstn Sektors (00..07)

c85 DWRITE

Schreibe auf absolute Sektoren ab der Speicheradr (HL)

sonst wie c84

c86 DVERIFY

Vergleiche ab (HL)

c87 GETDRVST

Dizkstatus lesen >Areg

b7 Diskettenmotor aus

b6 Kein WP (nur gültig,wenn Motor an und Disk in Laufwerk)

b3 Diskette im Laufwerk

b2 Diskette gewechselt (nur gültig,wenn b0 gesetzt)

b0 Diskettenoperation läuft

c88 HFREAD

Lies 100 Bytes (=halber Sektor) von Spur Dreg,Sektor Ereg
nach (HL)

c89 HFVERIFY

Vergleiche 100 Bytes ab (HL),sonst wie c88

Ein Leckerbissen:

DISKCOPY : CALL 05,5FF0

Wenn beide RAM-Module voll ausgebaut sind und also 64d KB RAM zur Verfügung stehen, kopiert dieses Programm komplette Diskettenseiten (mit komfortabler Benutzerführung) 1

Physikalisches Aufzeichnungsformat:

10 Spuren

08 Sektoren

200 Bytes/Sektor, ergibt 64d KB pro Seite (unformatiert)

14.DATEIEN

<<< STRUKTUR UND ORGANISATION >>>

Sowohl auf der Diskette als auch in der RAM-Diak werden alle Daten in Sektoren von 200 Bytes Größe aufgeteilt. Je 8 Sektoren bilden eine Spur (=Track). Je nach Disk-Kapazität werden ein oder mehrere Sektoren zu einer Verwaltungseinheit zusammengefaßt, die als Cluster bezeichnet wird.

Logische Struktur einer Disk:

Sektor 0: Bootsektor

x: File Allocation Table (FAT)

y: Directory (S2: ab 8600)

z: Daten

Der Bootsektor kann ein Programm enthalten, das beim Reset automatisch gestartet wird, wenn Byte Nr.3 (=viertes Byte) C3 ist.

- Diskette: Start ab Byte 20

- RAM-Diak: Start ab der Adresse, die in Byte 4 und 5 gegeben ist (siehe Speicherorganisation, Header einer RAM-Disk)

In FAT werden die Cluster der Dateien verwaltet, das erste Byte enthält Informationen über Organisation und Größe der Disk (=Media ID Code, wird weiter unten erklärt).

Die folgenden 254 Bytes geben an, ob ein Cluster des Datenbereichs (Zählung von 01 bis FE) zu einer Datei gehört und welche Nummer der nächste Cluster der Datei hat. Ist die Clusternummer =FF, so war dies der letzte Cluster der Datei.

Ein unbenutzter Cluster, der keiner Datei angehört, wird durch den Wert 00 dargestellt.

Im Directory (=Inhaltsverzeichnis der Disk) sind jeder Datei 20 Bytes zugewiesen, die folgende Informationen enthalten:

+00-07 : Name der Datei, E5:gekillt

+0B-0A : Extension

+0B : Set-Code (normalerweise 20)

b0 "P":Schreibschutz

b1 "I" : ?

b2 Wird bei FILES nicht gelistet

b5 Immer gesetzt

+0C-15 : 00

+16-19 : Zeit, b7 b6 -b5 b4 b3 b2 b1 b0

+16 : Minute (b2-b0) / 0 0 0 0 0

+17 : Stunde (b4-b0) / Minute

(b5-b3)

+18 : Monat (b2-b0) / Tag (b4-b0)

+19 : 0 0 0 0 0 0 0 Monat

(b3)

+1A/ : Erste Cluster-Nummer der Datei

+1C/ : Dateilänge mod 65536d

+1E/ : Dateilänge\65536d

Media ID:

Folge aus 13 Bytes (bei der RAM-Disk 18 Bytes), die folgende Informationen über die Aufteilung der Disk enthalten:

adr	Funktion	Standardwert bei 16KB 32KB 64KB 128KB (Diskettenlaufwerk: wie 64KB)			
+00	*Media ID Code	F0	F1	F2	F4
+01	:Sektorgröße (Bytes/Sektor)	0200	0200	0200	0200
+03	:Sektorgröße/20 -01	0F	0F	0F	0F
+04	zDirshift=LOG (Sektorgröße/20) zur Basis2	04	04	04	04
+05	:Zahl der Sektoren pro Cluster -01	00	00	00	00
+06	:Clustershift=LOG (Sekt. pro Cluster), Basis2	01	01	01	01
+07	:Clusternummer des ersten FAT	0001	0001	0001	0001
+09	:Zahl der FATs	01	01	02	02
+0A	:Max. Zahl der Dateien in Directory	20	30	30	80
+0B	:Clusternummer des ersten Datensektor	0004	0005	0006	0008
+0D	:Zahl der vorhandenen Datencluster	001D	003C	007B	00F6
+0F	:Zahl der Cluster pro FAT	01	01	01	01
+10	:Clusternummer des ersten Directory-Sektors	0002	0002	0003	0003
+12	:Zahl der Sektoren pro Spur	08	08	08	08

Bei der RAM-Disk stehen hier noch Boot-Informationen:

+13 :00

+14 :00

+15: evtl. Ladeadresse des Boot-Programms

+17 :00: Der Bootsektor wird an die Ladeadresse geladen

FF: Das Bootprogramm wird direkt in der RAM-Disk ausgeführt

Der Media ID steht beim Diskettenlaufwerk in ROM, die Startadresse (4242) steht in Disketten-Puffer und kann so geändert werden.

Bei der RAM-Disk befindet sich der Media ID und die Boot-Informationen hinter dem Header im Adreßbereich 8008-801F.

File Control Block (FCB):

Jeder geöffneten Datei wird ein FCB zugeordnet, in den alle wichtigen Informationen und ein Block von 100 Bytes zum sequentiellen Schreiben und Lesen vorhanden sind. Der Standard-FCB zum Laden und Speichern mit (B)LOAD und (B)SAVE steht in den Adressen F3C7 bis F4FF. Alle anderen FCBs müssen mit MAXFILES reserviert werden und stehen ab der Adresse, die in F04E/ gehalten wird, in Abstand von 139 Bytes hintereinander.

Die Adresse des gewünschten FCB ist bei den Routinen (siehe unten) im Dereg anzugeben.

Struktur des FCB:

+00/ : FCB-Nummer, 00=geschlossen
+01-04 : Device name (X,Y,SO,Sl,COK,COM1,COM2,CAS)
Ist der Name kürzer als 4 Buchstaben, so muß der Rest mit
00 gefüllt werden.
+05 : Mode
01 : Input
02 : Output
03 : Append
+06 : Schreibpointer
+07 : Lesepointer
+08 : Status: b0=EOF
+09-28 : Directory-Inhalt der Datei (Name, Größe usw., siehe oben)
+29/ : Momentane Cluster-Nummer bei sequentiellen Zugriff
+2B/ : Zahl der schon bearbeiteten Cluster
+2D : Clusternummer im FAT
+2E/ : Bytes pro logischer Block (0100)
+30-32 : Zähler der log. Blocks
+33-35 :
+36 : b0: FCB verändert
b1: Datei-Inhalt verändert
+37 : 00
+38 : 00
+39-138: Puffer für einen log. Block

Interne Struktur einer Datei:

ASCII-Format : Kein Datei-Header
ASCII-Codes, zwischen den Zeilen: CR+LF (0D bzw.
0A) Dateiende : EOF (1A)

Header bei BASIC- und Maschinenprogrammen:

+00-03 : FF 10 00 00
+04 : 10: Maschinenprogramm
21: BASIC-Programm (in Token-Format)
+05/ : Länge mod 65536d
+07 : Länge\65536d
+08-0D : Nur für Maschinenprogramme, sonst 00
+08/ : Ladeadresse
+0A : Bank
+0B : Autostart-Adresse, oder FFFF
+0D : Bank Autostart, oder FF
+0E-0F : 00 0F
ab +10 : Nutzdaten

<<< ROUTINEN >>>

Die Routinen werden mit CALL 01DE aufgerufen, wobei im Creg der Funktionscode und in DEreg die Adresse des FCB zu übergeben ist. Der Name des gewünschten Gerätes und der Datei muß in FCB abgelegt sein. >Areg=00: Kein Fehler aufgetreten,sonst:

- b7: Falsches Gerät
- b6: Gerätefehler
- b5: Keine Diskette eingelegt
- b4: Falscher Funktionscode
- b3: Hardware-Fehler, Sonstiges
- b2: Dateiende beim Lesen überschritten
- b1: Disk voll
- b0: Datei nicht gefunden, Directory voll

c0F OPEN FILE

Datei öffnen (zum Lesen oder Anfügen)

c10 CLOSE FILE

Datei schließen

C11-c13:Nicht für COM,CAS

C11 SEARCH FIRST

Hole das Directory einer Datei nach der Adresse, die durch c1A gesetzt wurde (FILES-Befehl, erste Datei)
In Dateinamen sind Wildcards ("?.") erlaubt, 8 bzw. 3 "?. " ersetzen ein

c12 SEARCH NEXT

Suche die nächste Datei mit den Bezeichner, der in c11 angegeben wurde

c13 DELETE FILE

Lösche Datei, Wildcards sind hier möglich

c14 SEQUENTIAL RD

Sequentielles Lesen eines logischen Blockes (100 Bytes), dessen Adresse in c1A gesetzt wurde

c15 SEQUENTIAL WR

Sequentielles Schreiben, sonst wie c14

c16 CREATE FILE

Datei erzeugen (zum Schreiben)

c17 RENAME FILE

Ersetze Name (DE+09-DE+13) durch (DE+29-DE+33), Wildcards sind erlaubt (nicht für COM,CAS)

c1A SET DMA

c1B-c23: nicht für COM,CAS

```
c1B GET ALLOC
    >DSKF in Bytes: BCreg*Ereg*HLreg
    >BCreg: Bytes pro Sektor
    >Ereg : Sektoren pro Cluster
    >HLreg: Zahl der freien Cluster c1E SET ATTRB
    Setze Dateiattribute (DE+14)
c23 GET LENGTH
    Log. Blockzahl der Datei
    >DEreg; hi
    >HLreg: lo
```

<<< RAM-DISK-ROUTINEN >>>

Aufruf mit CALL 4008 der Bank 3, Funktionscode im Creg.Slot-Nr.
in Areg (01: "S1:" , 02: "S2:")

```
c80-c87 Sinngemäß wie bei der Diskettenstation c88 SET ID
    Setze Media ID (HL+00-HL+12)
c89 SECTSET
    Bankselect log. Sektornummer DEreg adr>HL*reg
c8A LOGREAD
    Lies Sektoren (Anzahl in Breg) alt je 200 Bytes nach (HL)
    DEreg: Nummer des ersten log. Sektors (Nicht Spur/Sektor)
c8B LOGWRITE
    Schreibe Sektoren,sonst wie c8A
c8C GET DISKSIZ
    Größe der RAM-Disk in 16KB-Schritten >Creg c8D GET RAMSIZ
    Größe des RAMs in Slot Areg >Dreg in 16KB-Schritten
    >Ereg Rest-in 4KB-Schritten
c8E SEARCH BOOT
    Suche Bootprozedur,gefunden: >nc,Slot-Nummer nach FC16
c8F START BOOT
    Starte Bootprogramm c90 SET DISKSIZ
    Setze RAM-Disk-Größe in 2KB-Schritten nach F055 bzw. F05B
```

<<< DISKETTEN-PUFFER >>>

Der Disketten-Puffer wird beim Reset (CALL 4002,Areg=00/01) reserviert. Seine Startadresse wird in F038/ gehalten.

Aufbau:

```
+000-1FF: Zwischenspeicher für einen Daten- oder Directory-Sektor
+200      : 01: 1 Block von "X" im Puffer präsent
           : 02: 1 Block von "Y" im Puffer präsent
+201-20B: Name der präsenten Datei
+20C      : 00: Inhalt des Puffers noch nicht verändert
+20D/     : Log. Nummer des momentan präsenten Sektors
+20F-30F: FAT "Y"
+310-410: FAT "X"
```

+411-429:

In den Disketten-Routinen zeigt das IYreg auf diesen Bereich:

+00/	:	Startadr von Media	IDX
+02/	:	Startadr des FAT	X
+04	:	00	X
+05	:	Checksumme des FAT	X
+06/	:	Startadr von Media	ID y
+08/	:	Startadr des FAT	y
+0A	:	00	y
+0B	:	Checksumme des FAT	y
+0C	:	Zahl der angeschlossenen Laufwerke	
+0D	:	Zuletzt verwendetes Laufwerk (01= X 02= Y)	
+0E	:	00	
+0F	:	b2: Port 70-73	
		b1: Port 78-7B	
		b0: Bei Fehler abbrechen	
+10	:	Attribut-Maske für SET-Befehl (D8)	
+11	:	Anzahl der Versuche bei Schreib/Lesefehler	
+12/	:	Sprungadr bei ungültigem Funktionscode in Creg	
+14/	:	Stackpointer bei Abbruch durch Fehler	
+16	:	PortadresBe (70/713)	
+17	:	00	

15.EDITOR (ABLAUFBESCHREIBUNG)

Leider gibt es für den Editor des FC-1600 keine in den Sprungtabellen festgelegten Routinen. Deshalb muß ich mich auf eine allgemeine Ablaufbeschreibung beschränken.

Der Start des Editors kann aus der untersten Position in Stack (F5FD/) abgelesen werden. Hier werden alle Umschaltssymbole der Statusanzeige zurückgesetzt, der Stackpointer mit F5FF initialisiert und die Rücksprungadresse auf den Stapel gelegt. Der Editor ist nämlich in BASIC die unterste Ebene der Eingabenverarbeitung, die erst bei Betätigung der ENTER-Taste verlassen wird. Es ist also nur mit Tricks möglich, die Eingabe für ein Maschinenprogramm als Unterprogramm zu benutzen (siehe Programmbeispiele).

Ist nach dem Aufruf der Tastenroutine das Carry-Flag gesetzt, so wird die ALARM\$-Meldung ausgegeben, und die Eingabe beginnt von vorne. Ansonsten werden zunächst die Umschalttasten sowie die Control-Sequenzen ausgeführt, deren Codes und Sprungadressen nach Tabellen überprüft und ggf. angesprungen werden. Die Codes der übrigen Tasten unter 20 werden (mit Ausnahme der Funktionstasten) als Index zu einer Adreßtabelle verwendet und die entsprechenden Routinen angesprungen.

Die übrigen Tasten sind nur erlaubt, wenn das Error-Flag (b7 von F8B0, ist in Creg gespeichert) zurückgesetzt ist.

Es folgen die Verarbeitung der Funktionstasten (RUN/PRO-Mode bzw. RESERVE-Mode) und die DEF-Tastenkombinationen (Programmstart oder vordefinierte BASIC-Befehle).

Erst jetzt werden die Einfüge- und Löschoperationen ausgeführt, die eingegebenen Zeichen (und eventuell auch die vorherigen) angezeigt und der Cursor an die neue Position gefahren. Hierzu können Sie sich die Adressen dieser wichtigen (und z.T. extrem komplizierten) Routinen beschaffen, wenn Sie ab Adresse 6800 der Bank 0 die folgende Befehlssequenz suchen, deren Adressen ich hier für zwei ROM-Versionen angeben kann:

6C86 :EI	POP HL		6B18 :EI	POP HL
CB 55	BIT 2,L		CB 55	BIT 2,L
28 05	JR Z,+05 >90		28 05	JR Z,+05 >22
18 17	JR +17 >A4		18 17	JR +17 >36
CD B7 47	CALL 47B7	*1	CD EB 46	CALL 46EB
C3 8F 48	JP 488F	*2	C3 C3 47	JP 47C3
47	LD B,A		47	LD B,A
CD 91 76	CALL 7691	*3	CD AD 75	CALL 75AD
C3 90 6C	JP 6C90		C3 22 6B	JP 6B22
CD F7 7C	CALL 7CF7		CD 15 7C	CALL 7C15
C3 E3 6B	JP 6BE3		C3 75 6A	JP 6A75
CD 59 7B	CALL 7B59	*4	CD 77 7A	CALL 7A77
D0	RET NC		D0	RET NC

- *1: Zeige Inhalt des Eingabepuffers ab der Position Breg (Breg=Bytenummer im Puffer) und ab den xy-Koordinaten in Dereg an
- *2: Setze den Blinkstatus des Cursors in Abhängigkeit von der Cursorposition
- *3: Setze die xy-Koordinaten des Cursors in Abhängigkeit von seiner Position in Eingabepuffer, die in Breg gegeben ist
- *4: Lösche ab der Pufferposition Breg Areg* Zeichen und füge Creg* Zeichen ein, die in (DE) stehen. Der neue Pufferinhalt wird angezeigt und der Cursor gesetzt.

Funktion des Lreg:

Lreg=00: Der Cursor steht jetzt hinter
den

eingefügten Zeichen

Lreg=01: Der Cursor bleibt, wo er ist

Im Editor gibt es einige Stellen, die auf das System-RAM zwischen

FD00 und FDFF zugreifen und Routinen im Adreßbereich über 7000 auf der Bank 3 benutzen. Ebenso gibt es einige Printroutinen und Zeichenkonvertierungen mit scheinbar unsinniger Funktion. Diese werden für die japanische Version des PC-1600 benutzt.

C) BASIC - INTERPRETER

1. PROGRAMMSPEICHER UND VARIABLENBEREICH

Der von BASIC verwaltete Speicher ist in 5 logische Bänke unterteilt. Bank 1 bis 4 sind den RAM-Modulen der physikalischen Bänke 0 bis 3 zugeordnet, die log. Bank 5 stellt den internen Arbeitsspeicher dar.

Verwaltungsadressen der Slots: (hier: Bank=log. Bank)

	S0:	S1:	S2:
Start hi	F029	F015	F015
Start-Bank "P" (FE=Speicherweiterung)	F02A	F016	F020
Ende hi		F017	F021
End-Bank (nur bei "P")		F02B	F022
BASIC-Start	F865	F019/	F023/
Bank	F02B	F01B	F025
BASIC-Ende	F867	F01C/	F026/
Bank	F02C	F01E	F028

Um beispielsweise Adresse und Bank der ersten BASIC-Zelle in momentan mit TITLE selektierten Programm zu finden, ist folgendermaßen vorzugehen:

Die Slot-Nummer (0..2) steht in F1D5.

Die Adresse, in der der BASIC-Start und dessen log. Bank steht, kann jetzt, je nach den Inhalt von F1D5, aus der obigen Tabelle entnommen werden. (Achtung: Der Inhalt von F865/ und F867/ ist in der PC-1500-Notation gegeben !)

Zur Konvertierung der log. Bank in die physikalische Bankadresse ist die Nummer der log. Bank zu F1D5 zu addieren.

Inhalt des Bytes, das in der so gewonnenen Adresse steht:

b0-b1 Slot-Nummer (0..2)

b4-b6 Phys. Bankadresse (00-30)

b7 Diese log. Bank wird in einen Programm-Modul verwendet

Somit kann die Bank der gesuchten Adresse wie folgt selektiert werden:

OUT &31,PEEK (&F1D5+LOG_BANK) AND &70

Die Routine 02F4 setzt die BASIC-Start- und Endadressen mit den dazugehörigen log. Banks in FE3C-FE40 je nach dem aktuellen Slot.

Interne Codierung eines BASIC-Programms (Beispiel):

```
10: INPUT A
20: GOTO 10
```

```
+00/ 000A binäre Zeilennummer 10d
+02 04 Länge der Zeile incl. Endcode
+03/ F091 Token des Befehls INPUT
+05 41 ASCII-Code des Zeichens A
+06 0D Endcode CR
+07/ 0014 Zeilennummer 20d
+09 07 Zeilenlänge
+0A/ F192 Token von GOTO
+0C 1F Markierung für eine folgende binär codierte Zahl
      (wird bei Zeilennummer-Konstanten aus Geschwindig-
      keitsgründen der ASCII-darstellung vorgezogen)
+0D/ 000A Sprungziel:Zeile 10d
+0F 00 Ende der binären Zahl
+10 0D Zeilenend-Code
+11 FF Ende des Programms: Die nächste Zeilennummer
      ist >=FF00
```

Falls die nächste Zeilennummer =0000 ist, wurde das Ende der log. Bank erreicht, und es muß die nächste Bank selektiert werden.

Der RESERVE-Speicher liegt zwischen den Modul-Header und dem Programmbereich, also z.B. im SO: zwischen C008 und C0C4.

Aufbau:

```
C008-C021 String I
C022-C03B String II,Alarm$
C03C-C055 String III
C056-C0C4 Funktionstasten-Belegung:
      Die Belegung einer Taste besteht aus der Tasten-
      Nummer und den zugeordneten Zeichen (auch Token),
      deren ASCII-Codes >=20 sein müssen.
      Tasten-Nummer: F1-F6,I : 01-06
                     F1-F6,II : 11-16
                     F1-F6,III: 09-0E
```

Den Standardvariablen sind folgende Breiche zugeordnet:

```
A-Z   : je 08 Bytes F900-F9CF
As-D$ : je 10 Bytes F8C0-FBFF
E$-O$ : je 10 Bytes F650-F6FF
P$-Z$ : je 10 bytes F750-F7FF
```


Alle anderen Variablen werden unterhalb der ROM-Modul-Arbeitspeicher abgelegt und wachsen dabei auf den Programmspeicher von oben zu.

Die Startadresse dieser Variablen wird in F899/(PC-1500-Notation) gehalten; ihre Endadresse ist stets eine Page-Grenze (Vielfaches von 100 Bytes), deren hi-Byte in F864 (mit zurückgesetztem b7) steht.

Jede einzelne Variable hat einen eigenen Identifikationskopf, der folgende Informationen enthält:

+00	ASCII-Code des ersten Buchstabens
+01	ASCII-Code des zweiten Buchstabens
	b7: Array (=Feldvariable)
	b5: String-Variable
+02	Abstand ab +04 zur nächsten Variable, lo-Byte
+03	hi-Byte
+04	Zeilenindex bei Feldvariablen
+05	Spaltenindex bei zweidimensionalen Feldvariablen
+06	Bytezahl pro Array-Element
	88: Numerische Variable
ab +07	Daten

2.INTERNE DATENDARSTELLUNG UND STACKS

Neben den Variablen gibt es noch Register zur Zwischenspeicherung von Werten bei der Abarbeitung von Ausdrücken und als Argument-speicher für Funktionen:

FA00-FA07 XX
FA08-FA0F ZZ
FA10-FA17 YY
FA1B-FA1F UU
FA20-FA27 VV
FA28-FA2F WW
FA30-FA37 SS

In ihnen können sowohl numerische Daten als auch Strings gespeichert werden.

Fließkomma-Zahlen:

+00 : Exponent (in Zweierkomplement) von 9D=-99d bis 63=+99d
+01 : b7: Vorzeichen negativ
+02-06 : Mantisse in BCD-System
+07 : 00

Beispiel: -9.87E-04: FC 80 98 70 00 00 00 00

Integer-Zahlen

:
+04 : B2
+05 : hi
+06 : lo

Strings:

+04 : D0
+05 : Startadr des Strings hi,b7 invertiert
+06 : lo
+07 : String-Länge

Der Bereich von FB10 bis FB5F dient als Stringpuffer, auf den die String-Startadressen in den BASIC-Registern oder der Stringpuffer-Pointer (F894) zeigen.

Stacks: Zwischen FA38 und FAFF befindet sich der BASIC-Stack (wie beim PC-1500). In ihm werden die folgenden Daten zwischengespeichert:

1. For-Next-Schleifen (je 0C Bytes)
Pointer des lo-Byte (F890), hi-Byte=FA
(alle Werte hi vor lo)
+00/ Schleifenvariablen-Adresse
+02/ Endwert der Schleifenvariable
+04/ Step
+06/ Startadresse der Schleife
b15 zurückgesetzt, wenn die Schleife in der folgenden Zeile beginnt
+08/ Zeilennummer des Schleifenbeginns
+0A/ Programm-Startadresse, b15 invertiert
2. Zwischenergebnisse (je 08 Bytes)
Pointer (F892)
in den Adressen oberhalb der For-Next-Schleifen
3. Gosub-Stack (je 06 Bytes)
Pointer (F891), wächst ab FAFF abwärts in den Speicher
(alle Werte hi vor lo)
+00/ Adresse hinter GOSUB
b15 zurückgesetzt, wenn GOSUB letzter Befehl in der Zeile war
+02/ Rücksprung-Zeilennunae
+04/ Programm-Startadresse, b15 invertiert

Ein Bank-Stack ist beim PC-1600 zusätzlich nötig:

Im Bereich von F1DB-F21C werden die log.Banks (F1CE) und (F1C1) für die FOR-NEXT-Schleifen (aufwärts) und die Unterprogramme (abwärts) gespeichert. Als Stackpointer fungieren dabei (FE3A) und (FE3B). Sie enthalten das lo-Byte, das zur Adresse F1DB addiert wird.

3.BASIC-INTERRUPTS

Die Z80-Interrupt-Routine setzt, wenn die entsprechenden Bedingungen gegeben sind und der BASIC-Interrupt mit STOP oder ON aktiviert ist, die Interrupt-Request-Bits in den Adressen F1D3/.

F1CF/: Interrupt STOP oder ON

1.Byte:

b7	ADIN
b6	WAKE\$(0)
b2	COM2
b1	COM1
b0	PHONE

2.Byte:

b7	ALARM\$
b6	TIME\$
b5	KEY 6
b4	KEY 5
b3	KEY 4
b2	KEY 3
b1	KEY 2
b0	KEY 1

F1D1/: Interrupt ON

F1D3/: IRQ-Bits

Bei der Abarbeitung eines BASIC-Programms werden nach jedem Befehl die Interrupt-Masken abgefragt. Ein BASIC-Interrupt-Programm wird nur dann gestartet, wenn in allen drei Masken das Bit für einen bestimmten Interrupt gesetzt ist.

Vorgänge bei einem BASIC-Interrupt:

- IRQ-Bit in F1D3/ löschen
- Interrupt-Maske (F1D1/) auf den Masken-Stack, dann erneuten Interrupt während der Interrupt-Routine verbieten
Der Interrupt-Masken-Stack steht in F31D-F32C. F335 enthält als Stackpointer das lo-Byte, das zur Adresse F31D addiert wird, um dort den Inhalt von F1D2 und F1D1 abzulegen.
- Rücksprungadresse wie bei GOSUB auf den BASIC-Stack

- Aufruf der vorher definierten BASIC-Interrupt-Routine:
Jeder definierten Interrupt-Routine ist im Bereich F32D-F334 ein Code zugeordnet :

00 : PHONE
01 : COM1
02 : COM2
07 : ADIN
08-0D : KEY 1-6
0E : TIME\$
FF : frei

Zu jeden dieser Codes stehen in Bereich F337-F35E je 5 Bytes mit der Adresse der Interrupt-Routine:

- 1 Byte log. Bank
- 2 Bytes TOP,
- 2 Bytes Adr
- Beim Befehl RETI:
- Ursprüngl. Interrupt-Maske wieder setzen
- Rücksprungadresse von BASIC-Stack
- Hauptprogramm fortsetzen

4.TOKEN

Alle BASIC-Befehle werden im Programm nicht in ASCII-Zeichen, sondern in speziellen Codes gespeichert, die als Token bezeichnet werden. Beim PC-1600 bestehen diese aus 2-Byte-Kombinationen, deren erster Wert >=E0 ist. Das zweite Byte gibt Aufschluß über den Befehls-Typ:

50-51: Operator (steht zwischen zwei Argumenten,z.B.MOD)
52-5F: Systemvariable (z.B.MEM)
60-7F: Funktionen (mit nachfolgenden Parametern,z.B.PEEK)
80-FF: Anweisung (z.B.GOTO)

Bei der Abarbeitung eines BASIC-Befehls wird aus diesen Codes mit Hilfe von Tokentabellen die Bank und die Adresse der dazugehörigen Maschinensprache-Routine, die zur Verarbeitung des jeweiligen Befehls dient, ermittelt und angesprungen. Dazu dient u.A. die Routine 02BC.

Die Standard-Tokentabelle steht auf der Bank 6. Die Adresse läßt sich wie folgt ermitteln:

Suchen Sie ab dem Sprungziel des Bankjumps bei 02BC folgende

Befehle:

```
DB 31      IN A,(31)
E6 8F      AND 8F
F6 60      OR 60
D3 31      OUT (31),A
21 xx xx   LD HI,Adresse
```

Für zwei mir bekannte ROM-Versionen lautet dies Adresse B6DE.

Die Startadressen der Tokentabellen in den ROM-Modulen stehen in (4013/) bzw. (6013/). Ist diese Adresse 0000, so besitzt das Modul keine eigenen BASIC-Befehle.

Der tatsächliche Tabellen-Beginn liegt 0036 Bytes hinter dieser Adresse.

Format der Tokentabellen:

```
+00   : Wert für Bankselekt Port 31
      80: Adresse des Befehls in einer, anderen Tabelle suchen
+01/   : Sprungadresse
+03/   : Token 0000=Tabellenende
```

+05 : Code der Befehlsart

Bei Anweisungen:

b6: Befehl kann in Zusammenhang mit ON verwendet werden
(z.B. ON PHONE GOSUB)

b5: In RUN-Mode erlaubt

b4: Im RESERVE-Mode erlaubt

b3: Programmierbarer Befehl

b2: Als direkter Befehl möglich

b0: Im PRO-Mode erlaubt

Bei Funktionen, Operatoren,

Systemvariablen:

b2: Parameterzahl variabel (z.B.POINT)

b1-b0: Zahl der Parameter

+06 : Länge der ASCII-Zeichen, Abstand von +07 zum nächsten
Befehl

+07- : ASCII-Zeichen des Befehlsnamen

Ein Ausdruck der Tokentabellen für zwei ROM-Versionen befindet
sich im Anhang.

<<< DIREKTE BEFEHLE >>>

Wird nach der Eingabe eines Befehls in den Eingabepuffer F21D-F31C (mit Hilfe des Editors) die ENTER-Taste betätigt, so werden die ASCII-Zeichen der Schlüsselwörter in Token verwandelt und im BASIC-Puffer FBBO-FBFF in gleichen Format wie in einer BASIC Zeile gespeichert.

Der Token des ersten Befehls in Puffer wird im DEreg abgelegt, und im HLreg befindet sich die Verarbeitungs-Adresse (z.B.FBB2). Die Routine 02E2 ermittelt die Adresse des Befehls und springt sie an, wobei HLreg wieder die Verarbeitungsadresse enthält.

Die Verarbeitungsroutine kann so ab (HL) eventuelle Argumente des Befehls auslesen und den Befehl verarbeiten. Tritt dabei ein Fehler auf, so ist beim Rücksprung des Carry-Flag gesetzt, und im Areg wird der Fehlercode übergeben. In diesen Fall wird die Fehlermeldung ausgegeben und die Tastensperre gesetzt, bevor die Kontrolle wieder zum Editor übergeht.

Ansonsten wird vor der Rückkehr in den Editor noch überprüft, ob der Befehl vollständig abgearbeitet wurde (ob also der zurückgegebene Verarbeitungspointer (HL) den Zeilenendcode 0D enthält), um andernfalls einen Syntax Error zu erzeugen.

<<< PROGRAMM-VERARBEITUNG >>>

Der Unterschied der Programm-Verarbeitung zur Behandlung eines direkten Befehls liegt darin, dass sie eine Schleife darstellt, in der bis zu einer Programmunterbrechung ein Befehl nach dem anderen abgearbeitet wird. Als Startpunkt der Schleife kann die Adresse, die 0016 Bytes vor dem Sprungziel der Routine 02E5 liegt, angesehen werden. (Bei wir bekannten ROM-Versionen sind dies die Adressen 3C04 bzw. 3B5F.) Hier wird aus (HL) der Code des nächsten Befehls bzw. ein Variablen-Name ausgelesen und eine Rücksprungsadresse auf den Stack gelegt, die nach der bearbeitung des Befehls angesprungen wird. Hier werden Bank und Stackpointer initialisiert und bei einem Fehler die Fehlerbehandlungsroutine im BASIC-Programm (bei ON ERROR GOTO) verarbeitet oder mit einer Fehlermeldung abgebrochen.

Falls inzwischen das Ende einer BASIC-Zeile erreicht wurde, wird die Adresse der nächsten Zeile ermittelt und in F89C gespeichert. Es folgen die Abfrage der BRK-Taste und der ALARM\$-Funktion. Nach der Verarbeitung von eventuellen TRACE-Funktionen und BASIC-Interrupts werden, wenn nötig, eine neue logische Bank selektiert und Labels vor- dem nächsten Befehl überlesen. Damit ist die Schleife durchlaufen und beginnt wieder von vorn.

<<< FUNKTIONEN >>>

Bei allen Funktionen mit einem Parameter werden sowohl Parameter als auch Returnwert in XX-Register übergeben. In diesem Fall steht im Argument-Counter F88C der Wert 01, bevor die Funktion mit dem Token in Dereg mit CALL 0202 aufgerufen wird.

Die Operatoren besitzen ihre eigenen Sprungadressen
(01F3-01F9

bzw. 021A-0223) und erwarten ihre Parameter in den Registern XX und YY, wobei in F88C der Wert 02 stehen muss. Der Returnwert wird im XX abgeliefert.

Die numerischen Vergleichsoperatoren werden mit CALL 01FC aufgerufen. Sie sind durch das Dereg charakterisiert:

<> 8000
< 8001
> 8002
= 8004
<= 8005
>= 8006

Ansonsten werden sie wie die anderen Operatoren behandelt.

Die String-Vergleichsoperatoren sind über CALL 01FF zu erreichen und werden wie folgt in Dereg codiert:

<> 8000
< 8001
> 8002
= 8004

D) P R O G R A M M B E I S P I E L E

1. ALLGEMEINE TIPS ZUR MASCHINENPROGRAMMIERUNG

Maschinenprogramme sind in der Regel nur schwer durchschaubar, und Fehler, die meist fatale Folgen haben, sind oft nur mit großem Aufwand zu finden und zu beseitigen. Deshalb sollte man sich von Anfang an einen sauberen Programmierstil angewöhnen, der Fehler erst gar nicht aufkommen läßt. Bei der Entwicklung von neuen Programmen haben sich folgende Schritte bewährt:

- Stellen Sie eine Liste der Aufgaben zusammen, die das Programm durchführen soll, und teilen sie diese in Unterprogramme ein.
- Entwerfen Sie die notwendigen Datenstrukturen und überlegen Sie, welche Variablen benötigt werden.
- Zeichnen Sie ein Ablaufdiagramm.
- Schreiben Sie jedes größere Unterprogramm getrennt und testen Sie es separat mit allen möglichen Parametern und Returnwerten. Damit ist das Programm noch überschaubar, und Fehler lassen sich viel leichter lokalisieren.
- Eine genaue Dokumentation mit allen Adressen von Routinen, Daten und Variablen ist für spätere Änderungen und Erweiterungen eine große Erleichterung.

Falls Sie keine Programmierhilfen zur Verfügung haben, geben Sie das Programm am besten wie folgt ein:

- Schreiben Sie das Programm auf Papier in der Assemblersprache (mit Mnemonics).
- Geben Sie die Codes in POKE-Zeilen ein, und zwar alle Zahlen im Hexadezimalsystem, immer einheitlich 16d Bytes in einer Zeile. Beispiel:
10 POKE &C100,&00,&FF,&12,&34, (16d
Werte)
20 POKE
&C110,&FE,&DC,
- Speichern Sie das POKE-Programm (auch BASIC-Lader genannt) für spätere Änderungen ab.
- Starten Sie den BASIC-Lader und testen Sie das Maschinenprogramm, indem Sie es mit CALL aufrufen.
- Fehler sind am leichtesten in BASIC-Lader zu korrigieren, weil Sie dort die Codes vor sich sehen und einfacher editieren können.
- Erst wenn alles einwandfrei läuft, können Sie das Maschinenprogramm direkt mit BSAVE abspeichern und den BASIC-Lader löschen.

Mit einem Assembler (siehe auch Referenzliste) können Sie sich die Arbeit um einiges erleichtern. Oft stehen dann auch ein Breakpoint-Monitor (zum Test-Abbruch an kritischen Stellen) und ein Single-Step (zum schrittweisen Testen des Programms) zur Verfügung.

2.VOM BASIC-PRINT ZUM MASCHINENPROGRAMM

Wir wollen nun folgendes kleine Programm in Maschinensprache umsetzen:

```
10 FOR I=0 TO 2
20 CURSOR I*8,2
30 PRINT STR$(I+1);".Hallo"
40 NEXT I
50 END
```

Aufgabe:

Es soll in der Zeile 2 dreimal das Wort "Hallo" mit der entsprechenden Nummer ausgegeben werden.

Daten:

Wir benötigen den String "x.Hallo".

Variablen:

Als Schleifenzähler für 3 Durchläufe benutzen wird das Creg.

Ablaufdiagramm:

Schleifenzähler mit 0 initialisieren

Schleifenbeginn

Cursor in Dereg berechnen und Routine aufrufen

Hallo-String mit der Nummer korrigieren

Endcode im Areg und Adresse in Dereg setzen

Printroutine aufrufen

Schleifenzähler inkrementieren

Wenn noch nicht größer als 2:Neuer Durchgang

Rücksprung nach BASIC

Assembler-Listing:

E200;	0E 00	LD C,00	Zähler initialisieren
E202;	51	LD D,C	mom. Zähler-Wert nach Dreg kopieren
E203;	CB 22	SLA D	mit 8 multiplizieren
E205;	CB 22	SLA D	=dreimal nach links schieben
E207;	CB 22	SLA D	ergibt x-Wert
E209;	IE 02	LD E,02	y-Wert
E20B;	CD 15 01	CALL 0115	Cursor setzen
E20E;	79	LD A,C	Hallo-Nummer ins Areg
E20F;	C6 31	ADD A,31	in ASCII-Zeichen verwandeln
E211;	11 21 E2	LD DE,E221	Startadresse des Stririgs
E214;	12	LD (DE),A	Hallo-Nummer als 1.Zeichen speichern
E215;	3E 0D	LD A,0D	Endcc-de ins Areg
E217;	CD F1 00	CALL 00F1	Printroutine
E21A;	0C	INC C	Schleifenzähler +1

```
E21B: 79      LD A,C      nach Areg
E21C: FE 03   CP 03      Test auf 3 Durchgänge
E21E: 38 E2   JR C,-1E   Falls nicht, Sprung nach E202
E220: C9      RET        Ende
E221: 00      Platz für Hallo-Nummer
E222: 2E 48 61 6C 6C 6F ".Hallo"
E228: 0D      Endcode des Strings
```

Die Startadresse des Programms wurde willkürlich gewählt; sie kann beliebig verändert werden, wenn auch die Adresse im Befehl LD DE,E221 entsprechend geändert wird.

BASIC-Lader:

```
10 POKE &E200,&0E,&00,&51,&CB,&22,&CB,&22,&CB,&22,&1E,&02,&CD,
   &15,&01,&79,&C6
20 POKE &E210,&31,&11,&21,&E2,&12,&3E,&0D,&CD,&F1,&00,&0C,&79,
   &FE,&03,&38,&E2
30 POKE &E220,&C9,&00,&2E,&48,&61,&6C,&6C,&6F,&0D,&00,&00,&00,
   &00,&00,&00,&00
40 END
```

Nach dem Start des BASIC-Laders mit RUN kann dieser gelöscht und das Programm mit CALL &E200 aufgerufen werden.

Das Maschinenprogramm ist nur ca. 6mal schneller als das BASIC-Programm (bei 256d Durchläufen ca. 6.4 sek). Das liegt an der langsamen Standard-Printroutine. Würde mit einer speziellen Printroutine direkt auf die Hardware der Anzeige zugegriffen, so wäre mit einer weiteren Geschwindigkeitssteigerung etwa um den Faktor 8 zu rechnen.

3.POINT IN MASCHINENSPRACHE

Als Demonstration des Zugriffs auf die Anzeigen-Hardware wollen wir nun die Wirkung des BASIC-Befehls POINT nachvollziehen.

Dieses BASIC-Programm gibt einen senkrechten Strich aus und liest seinen Punktcode 10000d mal wieder ein:

```
10 GCUSOR 100,16:GPRINT 126;
20 FOR I=0TO 9999:A=POINT 100:NEXT
I 30 PRINT A
40 END
```

Die Schleife dauert ca.125d Sekunden.

Ein Maschinenprogramm mit der gleichen Funktion, das die Standard-POINT-Routine (015A) verwendet, wäre nur geringfügig schneller. Eine selbst programmierte Routine, die speziell auf die gewünschten Bildschirm-Koordinaten zugreift, kann erheblich schneller sein:

E200:	21 10 27	LD HL,2710	Schleifenzähler für 10000d Durchläufe
E203:	CD OC E2	CALL E20C	POINT-Routine
E206:	2B	DEC HL	Schleifenzähler -1
E207:	7C	LD A,H	Test auf 0000
E208:	B5	OR L	
E209:	20 F8	JR NZ,-08	Falls nicht, Sprung auf E203
E20B:	C9	RET	Ende
E20C:	F3	DI	Interrupt verbieten
E20D:	3A 5C F0	LD A,(F05C)	Nummer der mon. obersten Zeile >Areg
E210:	C6 02	ADD A,02	Zeile 2 anwählen (für y-Wert 16d)
E212:	E6 07	AND 07	und im Port 54 (weil der x-Wert
E214:	F6 B8	OR B8	zwischen 40 und 7F liegt) heraus-
E216:	D3 54	OUT (54),A	schreiben
E218:	E3	EX (SP),HL	warten
E219:	E3 EX	(SP),HL	
E21A:	18 00	JR +00	
E21C:	3E 64	LD A,64	Spalte ((x-Wert -3F) v40) aus-
E21E:	D3 54	OUT (54),A	geben
E220:	E3	EX (SP),HL	warten
E221:	E3	EX (SP),HL	
E222:	18 00	JR +00	
E224:	DB 57	IN A,(57)	1.(ungültigen) Wert lesen
E226:	E3	EX (SP),HL	warten
E227:	E3	EX (SP),HL	
E228:	18 00	JR +00	
E22A:	DB 57	IN A (57)	2.Wert (POINT-Wert) lesen
E22C:	32 31 E2	LD (E231),A	nach E231 abspeichern
E22F:	FB	EI	Interrupt wieder erlauben
E230:	C9	RET	Ende der POINT-Routine

Das BASIC-Programm zum Testen sieht wie folgt aus:

```
100 GCURSOR 100,16:GPRINT
126;
110 CALL &E200
120 PRINT PEEK &E231
130 END
```

Nach dem Laden des Maschinenprogramms kann der Test mit GOTO 100 gestartet werden. Diese Routine dauert nur noch ca. 1.2 Sekunden und ist damit über hundertmal schneller!

4.UMLEITUNG VON SYSTEMROUTINEN

Fast alle Display- und Tastaturroutinen besitzen Einsprungstellen für eigene Erweiterungen. Dazu stehen am Anfang der Routinen Unterprogrammaufrufe an den Anfang des ROMs (CALL 0070-CALL 00D9, je nach Routine). Von hier aus erfolgen CALLs nach vier verschiedenen Adressen im RAM, in denen normalerweise INC SP, INC SP, RET steht. In diese Adressen kann ein Jump geschrieben werden. Im Stack befindet sich die Rücksprungadresse auf die CALLs am ROM-Anfang (also 0073-00DC), über die die aufrufende Routine identifiziert werden kann:

Einsprung F09D:

0073: PRTANK
0076: PRTDBL
0079: PRTASTR
007C: CLS
007F: RVSCHR
0082: CRSRITAT
0085: LINE
0088: BOX
008B: DOTSET
008E: DOTREAD
0091: UPSCRL
0094: DWNSCRL
0097: SBLREAD
009A: SMBLSET
009D: ERSSTR
00A0: INSILN
00A3: ERSILN
00A6: PRTGCHR
00A9: PRTGSTR
00AC: PRTGPTN
00AF: GPTNREAD
00B2: nicht benutzt
00B5: nicht benutzt
00B8: PRTBSTR
00BB: BSPCTR

Einsprung F0A2:

00BE: PRTANK
00C1: PRTDBL

Einsprung F0A7:

00C4: KEYGET

00C7: KEYGETR

00CA: BREAKCHK

00CD: CURUDCHK

00D0: KEYDIRECT

00D3: OFFCHK

00D6: KEYGETND

00D9: BREAKRESET

Einsprung F3C1:

00DC: COM-Routine CRCVA

Als Beispiel soll die Standard-Printroutine PRTBSTR (00F1) so verändert werden, daß Groß- und Kleinbuchstaben vertauscht werden (siehe besonders beim Listen von BASIC-Programmen ungewohnt aus): In die Einsprungstelle F09D wird JP E200 geschrieben.

Hier muß zuerst getestet werden, ob es sich tatsächlich um die Printroutine handelt, die den Einsprung aufgerufen hat. Andernfalls erfolgt ein Rücksprung.

E200: E3 EX (SP).HL HLreg sichern, Rücksprungadr

>HLreg

E201: F5 PUSH AF AFreg sichern

E202: 7D LD A,L lo-Byte der Rücksprungadresse

E203: FE B8 CP B8 von Print-Routine ?

E205: 28 03 JR Z,+03 wenn ja, Sprung nach E20A

E207: F1 POP AF sonst AFreg wieder herstellen

E208: E1 POP HL HLreg vom Stapel nehmen und mit um

E209: C9 RET 2 erhöhten Stackpointer zurück

Jetzt beginnt die neue Printroutine (die nicht kommentierten Quelle wurden größtenteils von der Original-Routine übernommen):

E20A: 3E 66 LD A,66 Bank 6/3 selektieren

E20C: D3 31 OUT (31),A

E20E: F1 POP AF AFreg wieder zurückholen, HLreg bleibt

E20F: C5 PUSH BC

E210: 47 LD B.A

E211: EB EX DE.HL

E212: 3A 67 F0 LD A,(F067)

E215: F5 PUSH AF

E216: CD 9C 80 CALL 809C

E219: 7E LD A,(HL)

E21A: B8 CP B

E21B: 28 25 JR Z,+25

E21D: FE 41 CP 41 Kleiner als "A" ?

E21F: 38 0E JR C,+0E Ja, dann keine Änderung. Sprung

E22F

E221: FE 5B CP 5B Zwischen "A" und "Z" ?

E223: 38 08 JR C,+08 Ja, dann Änderung, Sprung nach E22D

E225: FE 61 CP 61 Zwischen "[" und "'" ?

E227: 38 06 JR C,+06 Ja, dann Sprung nach E22F

E229: FE 7B CP 7B Größer als "z" ?

E22B: 30 02 JR NC,+02 Ja, dann Sprung nach E22F

E22D: EE 20 XOR 20 Änderung: b5 invertieren

E22F: 23 INC HL


```
E230: ED 5B 5F F0  LD DE,(F05F)
E234: CD 4E 40      CALL 404E      Das Zeichen im Areg ausgeben
E237: ED 53 5F F0  LD (F05F),DE
E23B: 30 DC         JR NC,-24
E23D: D1           POP DE
E23E: 3E 01        LD A,01
E240: 18 05        JR +05
E242: F1           POP AF
E243: CD A2 80     CALL 80A2
E246: AF           XOR A
E247: EB           EX DE,HL
E248: C1           POP BC
E249: E1           POP HL      HLreg zurückholen,SP ist um 2
erhöht
E24A: 33           INC SP      Rücksprungadresse in die Original-
E24B: 33           INC SP      Printroutine überspringen
E24C: C9           RET
```

Ist das Programm geladen, so muß zur Aktivierung noch POKE &F09D,&C3,&00,&E2 eingegeben werden. Zur Beendigung der Umleitungs-Funktion bitte den Computer abschalten oder POKE &F09D,&33,&33,&C9 eintippen.

5.SOUNDGENERATOR

Hier soll ein kleiner Soundgenerator vorgestellt werden, den man z.B. als Gag in Spielen einsetzen könnte. Er soll als Anregung dienen, was man mit dem Buzzer alles anstellen kann. Es wird ein immer schneller auf- und abschwellenden Ton erzeugt, der immer höher wird und schließlich abbricht. Experimentieren Sie mit den 2 Konstanten im Programm! Tonhöhe und Geschwindigkeit lassen sich in weiten Grenzen variieren.

E200:	F3	DI	Der Interrupt stört den Ton
E201:	1E 50	LD E,50	Wert für Dauer und Wiederholung
E203:	53	LD D,E	
E204:	4A	LD C,D	
E205:	DB 18	IN A,(18)	Einen Knackiapuls erzeugen
E207:	EE 60	XOR 80	(Buzzer an-
E209:	D3 18	OUT (18),A	und abschalten)
E20B:	06 18	LD B,18	Wert für Tonhöhe und Geschwindigkeit
E20D:	10 FE	DJNZ -02	konstante Verzögerung
E20F:	0D	DEC C	
E210:	20 F9	JR NZ,-07	variable Verzögerung
E212:	15	DEC D	
E213:	20 EF	JR NZ,-11	Wert der Tonhöhensteigerung
E215:	1D	DEC E	
E216:	20 EB	JR NZ,-15	Zahl der Tonhöhensteigerungen
E218:	F6 80	OR 80	
E21A:	D3 18	OUT (18),A	Buzzer auf jeden Fall abschalten
E21C:	FB	EI	
E21D:	C9	RET	

Änderungen können mit POKE &E202,X und POKE &E20C,Y durchgeführt werden.

10.BENUTZUNG DES EDITORS

Für eigene Maschinenprogramme ist oft eine Eingabemöglichkeit über einen Editor wünschenswert. Selbst einen Editor zu programmieren, ist sehr aufwendig, so daß man am besten auf den relativ komfortablen BASIC-Editor zurückgreift. Dies ist allerdings nur mit ein paar Tricks möglich, die ich hier aufzeigen will:

Da die Eingabe ein Unterprogramm sein soll und der Editor den Stackpointer auf F5FF setzt, muß das Hauptprogramm einen anderen Stackbereich besitzen (z.B. ab F57F abwärts). Der Stackpointer muß also vor dem Aufruf der Editor-Routine zwischengespeichert werden.

In den Eingabepuffer F21D-F31C können eine Eingabe-Meldung und eine Voreingabe (sehr komfortabel, wenn viel mit Default-Eingaben gearbeitet wird) gespeichert werden. Der Rest des Puffers wird mit 0D-Codes gefüllt.

Drei Pufferpointer sind zu setzen:

FE13: Startposition der Default-Eingabe
(hinter der Input-Meldung)

FE12: Endposition der Default-Eingabe (0D-Position)

FE11: Mom. Position des Cursors im Puffer

Koordinaten und Blinkstatus des Cursors werden mit der CRSRSET (0115)- bzw. CRSRSTAT (011E)- Routine gesetzt.

Sonstige Einstellungen:

F1BC: b1 (BRK-Sperre) setzen

F880: 40 (Cursor an)

F88A: 20 (Input)

FE10: 00 (linker Rand)

FE18: 1A (Zeilenlänge)

FE42: b2 (2-Zeilen-Modus) zurücksetzen

FE60: 00 (Pufferverschiebung)

Busy-Symbol löschen

Die Tastenroutine KEYGET (0166) muß mit dem Einsprung F0A7 angezapft werden: Die CA-, MODE- und DEF- Tasten werden abgefangen, und die ENTER-Taste dient zur Beendigung der Eingabe, indem der alte Stackpointer wieder hergestellt wird und ein Rücksprung in das aufrufende Hauptprogramm erfolgt.

Test-Hauptprogramm:

E200: 31 60 F5	LD SP,F580	Anderen Stackbereich verwenden
E203: 3E 02	LD A,02	Zeile 2
E205: 06 08	LD B,06	Input-Meldung :Länge 8
E207: 0E 05	LD C,05	Voreingabe :Länge 5
E209: 11 01 09	LD DE,0901	Cursor x:9 y:1
E20C: 21 23 E2	LD HL,E223	Startadr der, Meldung
E20F: CD 30 E2	CALL E230	Eingaberoutine
E212: 01 10 00	LD BC,0010	16d bytes aus dem Eingabe-

```

E215: 11 C0 F8      LD DE,F8C0      puffer ab Position 8 in
E218: 21 25 F2      LD HL,F225      die Standardvariable A$
E21B: ED B0         LDIR            kopieren
E21D: CD 12 03      CALL 0312       Rücksprung ins
E220: C3 53 02      JP 0253         BASIC (wie nach Reset)
E223: 45 69 6E 67 61 62 65 3A      "Eingabe:"
E22B: 31 32 33 41 42      "123AB"

```

Eingabe-Routine:

Parameter:

Areg :Cursor-Blinkstatus (wie in der CRSRSTAT-Routine)

Breg :Länge der Input-Meldung

Creg :La'nge der Voreingabe

Dereg:Koordinaten des Cursors

HLreg:Startadresse der Input-Meldung und Voreingabe

```

E230: F5           PUSH AF
E231: 78           LD A,B           Länge der Input-Meldung
E232: 32 13 FE     LD (FE13),A
E235: 81           ADD A,C           Länge der Voreingabe addieren
E236: 4F           LD C,A           Gesamtlänge nach BCREg
E237: 06 00        LD B,00
E239: 32 12 FE     LD (FE12),A
E21C: 7A           LD A,D           Cursorposition x
E23D: 32 11 FE     LD (FE11),A
E240: F5           PUSH AF
E241: 50           LD D,B           Dreg=00,Ereg=Cursorkoordinate y
E242: CD 15 01     CALL 0115        Setze Koordinaten für Print
E245: 11 1D F2     LD DE,F21D       Startadr Puffer
E248: D5           PUSH DE
E249: AF           XOR A           Zahl der restlichen bytes im
E24A: 91           SUB A,C           Puffer ermitteln
E24B: 28 02        JR Z,+02         Falls 256d,keine Meldung kopieren
E24D: ED B0        LDIR            Meldung+Voreingabe in den Puffer
E24F: 47           LD B,A           Rest mit 0D füllen
E250: 3E 0D        LD A,0D
E252: 12           LD (DE),A
E253: 13           INC DE
E254: 10 FC        DJNZ - 04
E256: D1           POP DE           Dereg=F21D
E257: CD F1 00     CALL 00F1        Pufferitihalt printen (bis 0D)
E25A: F1           POP AF           Cursorpositon x
E25B: 32 60 F0     LD (F060),A      setzen
E25E: F1           POP AF           Blinkstatus
E25F: CD 1E 01     CALL 011E        setzen
E262: 21 BC F1     LD HL,F1BC       sonstige Systemadressen setzen
E265: CB CE        SET 1,(HL)
E267: 3E 40        LD A,40
E269: 32 80 F8     LD (F880),A
E26C: 0F           RRCA
E26D: 32 8A F8     LD (F88A),A
E270: AF           XOR A
E271: 21 10 FE     LD HL, FE10
E274: 77           LD (HL),A
E275: 2E 60        LD L,60

```

Programmbeispiele; 6. Benutzung des Editors 1

E277:	77	LD (HL),A	
E278:	2E 18	LD L,18	
E27A:	36 1A	LD (HL),1A	
E27C:	2E 42	LD L,42	
E27E:	CB 96	RES 2.(HL)	
E280:	CD 39 01	CALL 0139	Busy-Symbol löschen (Breg=00)
E283:	E6 FE	AND FE	
E285:	CD 3C 01	CALL 013C	
E288:	21 FA E2	LD HL,E2FA	Einsprung nach F0A7 kopieren
E28B:	11 A7 F0	LD DE,F0A7	
E28E:	0E 03 00	LD C,03	Breg immer noch 00
E290:	ED B0	LDIR	
E292:	DB 31	IN A,(31)	Bank sichern
E294:	F5	PUSH AF	
E295:	E6 70	AND 70	Bank 0 (für Page 1) anwählen
E297:	D3 31	OUT (31),A	
E299:	ED 73 FE FF	LD (FFFE),SP	Stackpointer sichern
E29D:	C3 B3 69	JP 69B3/6844	Sprung in den BASIC-Editor
E2A0:	E3	EX (SP),HL	Tasteneinsprung
E2A1:	FS	PUSH AF	
E2A2:	7D	LD A,L	
E2A3:	FE C4	CP C4	Testet Einsprungadr von KEYGET
E2A5:	28 03	JR Z,+03	Ja,dann E2AA
E2A7:	F1	POP AF	
E2A8:	E1	POP HL	
E2A9:	C9	RET	
E2AA:	21 0B 00	LD HL,000B	Rücksprungadr von KEYGET
E2AD:	39	ADD HL,SP	nach FFFC/ speichern
E2AE:	7E	LD A,(HL)	
E2AF:	32 FD FF	LD (FFFD),A	
E2B2:	36 E2	LD (HL),E2	Mit der Adresse E2BD vertauschen
E2B4:	2B	DEC HL	
E2B5:	7E	LD A.(HL)	
E2B6:	32 FC FF	LD (FFFC),A	
E2B9:	36 BD	LD (HL),BD	
E2BB:	18 EA	JR -16	Tasteneinsprung normal beeriden
Nach der Tastenroutine Sprung hierher:			
E2BD:	FE 1A	CP 1A	CA-Taste ?
E2BF:	28 04	JR Z,+04	Ja,dann E2C5
E2C1:	FE 1F	CP 1F	MODE-Taste ?
E2C3:	20 09	JR NZ,+09	Nein,da'nn E2CE
E2C5:	3E 18	LD A,18	Durch CL-Taste ersetzen
E2C7:	E5	FUSH HL	Rücksprung in den Editor
E2C8:	2A FC FF	LD HL,(FFFC)	auf die aus deui Stack
E2CB:	E3	EX (SP),HL	gesicherte Adresse
E2CC:	A7	AND A	Carry-Flag zurücksetzen
E2CD:	C9	RET	
E2CE:	FE 1B	CP 1B	DEF-Taste ?
E2D0:	20 04	JR NZ,+04	
E2D2:	3E 01	LD A,01	Wenn ja,dann durch SHIFT ersetzen
E2D4:	18 F1	JR -OF	und in den Editor zurück
E2D6:	FE 0D	CP 0D	ENTER-Taste ?
E2D8:	20 ED	JR NZ,-13	Wenn nein,zurück zum Editor
E2DA:	ED 7B FE FF	LD SP,(FFFE)	Stackpointer wieder laden
E2DE:	F1	POP AF	Bank wieder herstellen

Programmbeispiele; 6. Benutzung des Editors 3

E2DF:	D3 31	OUT (31),A	
E2E1:	21 FD E2	LD HL,E2FD	Einsprung beseitigen
E2E4:	11 A7 F0	LD DE,F0A7	
E2E7:	01 03 00	LD BC,0003	
E2EA:	ED B0	LDIR	
E2EC:	CD 39 01	CALL 0139	Busy-Symbol an (Bre=00)
E2EF:	F6 01	OR 01	
E2F1:	CD 3C 01	CALL 013C	
E2F4:	21 BC F1	LD HL,F1BC	Break an
E2F7:	CB 8E	RES 1,(HL)	
E2F9:	C9	RET	Rücksprung ins Hauptprogramm
E2FA:	C3 AO E2	JP E2A0	Wert für die Einsprungstelle
E2FD:	33	INC SP	ursprünglicher Wert
E2FE:	33	INC SP	
E2FF:	C9	RET	

Die Sprungadresse in E29D kann mit HEX\$ PEEK
&F5FE+HEX\$ PEEK &F5FD gelesen werden.

Zur Anpassung an einen anderen Adreabereich sind die im Disassembler-Listing mit einem * gekennzeichneten Befehle entsprechend zu ändern.

Bei der Eingabe sollte keine Funktionstaste gedrückt werden, deren String mit dem @-Zeichen zur automatischen Ausführung endet. Dies kann zu einem Absturz des Rechners führen.

7.DER NEW-RETTTER

Dieses Programm ist weniger für den notorischen NEW-Eintipper gedacht, es soll vielmehr die Speicherstruktur eines BASIC-Programmes verdeutlichen.

Bei einem einfachen NEW (ohne Adreßparameter) wird der Pointer der BASIC-Startadresse auf den Pointer der Endadresse kopiert. Ebenso die Bank-Nummer. Außerdem wird das erste Byte des Programms (hi - byte der ersten Zellennummer) mit dem Endcode FF überschrieben.

Der NEW-Retter dagegen setzt dieses Byte wieder auf 00 (weil die erste Zeilennummer meist kleiner als 256d ist) und sucht ab der Programm-Startadresse Zeile für Zeile nach dem Programm-Endcode. Die Fundadresse und die logische Bank werden in den BASIC-Endpointer zurückgeschrieben.

Das Programm ist voll relokatable, kann also ohne Änderung an eine andere Adresse als E200 geladen werden. Allerdings darf es nicht unter der Adresse C000 stehen, sondern muß in dem von allen Bänken erreichbaren internen RAM stehen.

Im Breg (hier Adresse E201) zu(1 die Slot-Nummer, die evtl. beim NEW-Befehl angegeben wurde, oder der momentan mit TITLE selektierte Slot gePOKEt werden.

Wurde die Struktur des BASIC-Programms zerstört, so hängt sich der NEW-Retter auf.

E200:	06 00	LD B,00	Breg:Slot-Nummer
E202:	21 19 F0	LD HL,F019	Adr des BASIC-Start-Pointers für S1
E205:	05	DEC B	S1 ?
E206:	28 12	JR Z,+12	ja,dann E21A
E208:	2E 23	LD L,23	Adresse Pointer S2
E20A:	05	DEC B	S2 ?
E20B:	28 0D	JR Z,+0D	ja,dann E21A
E20D:	2A 65 F8	LD HL,(F865)	Startadresse des Programms S0
E210:	5C	LD E,H	nach DEreg in Z80-Notation kopieren
E211:	55	LD D,L	
E212:	CB FA	SET 7,D	
E214:	3A 2B F0	LD A,(F02B)	log. Bank S0
E217:	4F	LD C,A	>Cres
E218:	18 06	JR +06	Sprung nach E220
E21A:	5E	LD E,(HL)	Startadresse und Bank für S1/S2
E21B:	23	INC HL	
E21C:	56	LD D,(HL)	
E21D:	23	INC HL	
E21E:	4E	LD C,(HL)	
E21F:	23	INC HL	
E220:	E5	PUSH HL	Pointer für S1/S2 sichern
E221:	CD D4 02	CALL 02D4	log. Bank Creg in Wert für Port 31
E224:	D3 31	OUT (31),A	konvertieren und ausgeben

Programmbeispiel ; 7. Der NEW-Retter 1

E226:	EB	EX DE,HL	HLreg:Startadresse
E227:	36 00	LD (HL),00	mit 00 überschreiben
E229:	7E	LD A,(HL)	Zeilennummer hi-Byte
E22A:	FE FF	CP FF	Endcode erreicht
E22C:	28 0C	JR Z,+0C	ja,dann E23A
E22E:	23	INC HL	Pointer auf Zeilennummer lo
E22F:	B6	OR (HL)	beide 00 ?
E230:	28 27	JR Z,+27	ja,dann Ende der log. Bank, >E259
E232:	23	INC HL	Zeiger auf Zeilenlänge
E233:	5E	LD E,(HL)	nach Ereg
E234:	16 00	LD D,00	für Addition hi-Byte 00
E236:	1C	INC E	Adresse der nächsten
E237:	19	ADD HL,DE	Zeile berechnen
E238:	18 EF	JR -11	und weitersuchen >E229
E23A:	22 3F FE	LD (FE3F),HL	Endadresse
E23D:	79	LD A,C	log. Bank
E23E:	32 41 FE	LD (FE41),A	
E241:	EB	EX DE,HL	Endadresse nach Dereg
E242:	E1	POP HL	Pointer zurückholen
E243:	04	INC B	S1/S2 ?
E244:	04	INC B	
E245:	20 0C	JR NZ,+0C	ja,dann E253
E247:	63	LD H,E	Endadresse des Programms in LH-5803-
E248:	6A	LD L,D	Notation nach Pointer für SO
E249:	CB BD	RES 7,L	
E24B:	22 67 F8	LD (F867),HL	
E24E:	79	LD A,C	log. Bank
E24F:	32 2C F0	LD (F02C),A	
E252:	C9	RET	Ende S0
E253:	73	LD (HL),E	Endadresse und log. Bank S1/S2
E254:	23	INC HL	nach F01Cff bzw. F026ff
E255:	72	LD (HL),D	
E256:	23	INC HL	
E257:	71	LD (HL),C	
E258:	C9	RET	Ende S1/S2
E259:	0C	INC C	nächste log. Bank
E25A:	11 00 80	LD DE,8000	nächste Suchadresse
E25D:	CD D4 02	CALL 02D4	physikal. Bank
E260:	D3 31	OUT (31),A	selektieren
E262:	EB	EX DE,HL	Adresse >HLreg
E263:	18 C4	JR -3C	und weitersuchen >E229

8.EIN BOOT.PROGRAMM

Hier möchte ich Ihnen ein Boot-Programm für die Diskette vorstellen, das bei einem Totalreset individuelle Voreinstellungen (Defaults) vornimmt.

Als Beispiel wurden folgende Defaults gewählt:

Plotter-Reset abschalten

Plotterstift fährt bei Paperfeed-Taste nicht mehr in die Mitte
Tastenclick- und Wiederholfunktion an

Stundensignal an

NEW "SO:", &839 ausführen

Hexmon mit BLOAD "X:HEXMON.BIN", #0, &C0C5 laden

TIME abfragen

Realisierung:

Das Initialisierungsprogramm, das hier ab E1F0 steht, kopiert den Inhalt, von E200 bis E3FF in den Bootsektor (00). Von dort wird es beim Reset in den Diskettenpuffer ab der Adresse, die in F038 steht, geladen und ab Byte 0020 gestartet, weil im Byte 0003 der Wert C3 steht. Liegt kein Reset oder Totalreset vor, endet das Boot-Programm sofort. Andernfalls wird ein Programmteil in den ungenutzten Bereich von FEE0 bis FEF8 geladen und in die Einsprungstelle des BASIC-Interpreters integriert. Er dient zur Initialisierung des Plotters beim Reset und Anschalten, weil der Hardware-Reset des Druckers verhindert wurde. Bei einem Totalreset werden zusätzlich die o.g. Defaults gesetzt. Diese Defaults können Sie natürlich Ihren eigenen Bedürfnissen durch Umschreiben anpassen. Bitte beachten Sie, daß sich dabei die Startadresse und die Länge des Keybuffer-Strings ändern und entsprechend korrigiert werden müssen.

Initialisierungs-Programm:

E1F0:	3E 01	LD A,01	Diskette "X:"
E1F2:	01 85 01	LD BC,0185	1 Sektor,Funktionscode 85 (DWRITE)
E1F5:	11 00 00	LD DE,0000	Spur/Sektor 00
E1F8:	21 00 E2	LD HL,E200	ab Adresse E200
E1FB:	E7 05 08 40	CALL 05,4008	Disk-Systemroutinen aufrufen
E1FF:	C9	RET	Ende

Bootsektor-Inhalt:

E200:	00 00 00	unbenutzt (leer)	
E203:	C3	Bootprogramm vorhanden	
E204-E21F:	00	leer	
E220:	AF	XOR A	kein weiteres Boot-Programm
E221:	32 18 FA	LD (FA18),A	
E224:	3A 1B FA	LD A,(FA1B)	Reset-Ursache
E227:	FE 08	CP 08	Ende,wenn Power on
E229:	D0	RET NC	
E22A:	2A DD F0	LD HL, (F0DD)	Startadr BASIC-Interpreter als

E22D:	22	F7	FE	LD (FEF7),HL	Adresse für den Sprung benutzen
E230:	21	E0	FE	LD HL,FEE0	und durch die Adresse des Pro-
E233:	22	DD	F0	LD (F0DD),HL	grammteils ersetzen
E236:	EB			EX DE,HL	Adresse nach Dereg
E237:	2A	38	F0	LD HL,(F038)	Originaladres-
E23A:	01	63	00	LD BC,0063	se zum Kopie-
E23D:	09			ADD HL,BC	ren ermitteln
E23E:	0E	17		LD C,17	BCre8=Länge=0017
E240:	ED	B0		LDIR	nach FEE0ff kopieren
E242:	FE	02		CP 02	Totalreset ?
E244:	DO			RET NC	nein.dann Rücksprung
E245:	3E	28		LD A,28	Plotter-Reset und Mitten-
E247:	32	F8	F9	LD (F9F8),A	Funktion abschalten
E24A:	3E	03		LD A,03	Tastenclick und
E24C:	CD	7E	01	CALL 017E	Wiederholung einschalten
E24F:	3E	08		LD A,08	Stundenbeep an:
E251:	0E	14		LD C,14	SWPON wählen und
E253:	CD	D5	01	CALL 01D5	Timer aufrufen
E256:	3E	32		* LD A,32	Stringlänge für Tastaturpuffer
E258:	2A	38	F0	LD HL,(F038)	Startadresse berechnen:
E25B:	01	7C	00	* LD BC,007C	Offset ab E200
E25E:	09			ADD HL,BC	>E27C
E25F:	EB			EX DE,HL	nach Dereg
E260:	C3	6C	01	JP 016C	KBUFSET anspringen,Ende
Programmteil für Plotter-Reset:					
E263:	21	23	FA	LD HL,FA23	Checkmeldung des
E266:	CB	5E		BIT 3,(HL)	Plotters ?
E268:	20	0F		JR NZ,+0F	ja,dann FEF6
E26A:	21	AF	F0	LD HL,F0AF	Plotter angeschlossen ? (wird am
E26D:	CB	56		BIT 2,(HL)	Cassetteninterface-Modul
getestet)					
E26F:	28	08		JR Z,+08	nein,dann FEF6
E271:	2E	B2		LD L,B2	ROM-Interrupt für
E273:	CB	DE		SET 3,(HL)	Plotter anschalten
E275:	3E	07		LD A,07	Plottertasten an-
E277:	D3	80		OUT (80),A	schalten
E279:	C3	00	00	JP 0000	Sprung zur Startadr des BASIC-In-
terpreters,der hier (in FEF7/)ein-					
geschrieben wurde					
E27C:	1F				MODE-Taste:In PRO-Mode schalten
E27D:	4E	45	57 22 53 30 3A 22		NEW"SO:"
E285:	2C	26	38 33 39		,&839
E28A:	0D				ENTER-Taste
E28B:	42	4C	4F 41 44 22 58 3A		BLOAD»X:
E293:	46	45	58 4D 4F 4E 2E 42		HEXMON.B
E29B:	49	4E	22 2C 23 30 2C 26		IN",#0,&
E2A3:	43	30	43 35		C0C5
E2A7:	0D				ENTER-Taste
E2A8:	1F				MODE-Taste:Zurück in RUN-Mode
E2A9:	54	49	4D 45 3D		TIME=
E2AE-E3FF:00					leer

9. HEX- UND DISKMONITOR

Mit diesem Monitor können Sie den gesamten Speicher sowie den Inhalt der Diskettensektoren ansehen und editieren. Außerdem gibt es die Möglichkeit, ein Maschinenprogramm zu starten, wobei die Register davor und danach angesehen bzw. geändert werden können.

Das Progeamm paßt sich selbst an jede Adresse an und kann somit an jede beliebige Adresse (oberhalb von C000) geladen werden. Dazu muß es von BASIC aus mit CALL gestartet werden, damit es die Aufrufadresse in HL'reg und die Editoradresse in der untersten Stackposition übermittelt bekommt. So kann es eine Sprungtabelle für die absoluten Verzweigungen ermitteln und im Bereich von FF6E bis FFD9 eine Tabelle ablegen.

Nach dem Start wird zunächst die gewünschte Editieradresse abgefragt.

bei der Eingabe der Adresse ist folgendes Format einzuhalten:

1. Adresse als 4stellige Hexadezimalzahl
2. Ein Komma
3. Bank
 - 0-7:Bank 0-7
 - 3B :verstecktes
 - BASIC-ROM 4B :Japan-ROM
 - 20-27,30-37:Extra-Bankumschaltung (0..7,normal:0) für große Speichermodule im Slot 2
 - X :Diskette "X:"
 - Y :Diskette "Y:"Bei der Bank sind auch andere Kombinationen (z.B. XB oder 07) möglich, aber sinnlos.
4. Zeilenend-Code CR

Die Diskette wird als virtueller Speicher auf Bank X/Y mit dem Adreßbereich 0000-FFFF angesehen und blockweise im Diskettenpuffer editiert. Zusammenhang zwischen virtueller Adresse und der Spur/Sektor-Adressierung:

Adresse = (Spur)*1000 + (Sektor)*200 + Bytenummer im Sektor

Tastenfunktionen im Hexdump-Modus:

Pf-0 : Eine Zelle aufwärts
Pf-U : Eine Zeile abwärts
Pf-R : Ein Zeichen nach rechts
PF-L : Ein Zeichen nach links
DEF C: Editiermodus-Umschaltung von HEX auf ASCII und umgekehrt
DEF S: Neue Adresse
DEF G: In den Test-Modus schalten
MODE : Monitor verlassen
OFF : Rechner abschalten
SHIFT: Dauerumschaltung
SML : ---,---
KbII : ---,---
DEF : ---,--- ,bei den nicht schon oben erwähnten
Tasten

wird zum ASCII-Code der Wert A0 addiert, so daß im ASCII-Mode auch Graphik-Sonderzeichen direkt erreichbar sind.

Eine weitere Taste können Sie selbst definieren: 1.Zur Startadresse des Monitors 00C8 addieren

2.In die errechnete Adresse (und in die zwei folgenden) poken:
-Tastencode
-relative Sprungadresse (zur Monitor-Startadresse)
lo -rel. Adresse hi

Als Voreinstellung steht hier der Tastencode 00 (keine Taste) und die relative Adresse 0773, die auf ein RET am Programmende zeigt.

Der Test-Modus:

Mit den Cursortasten können Sie die Stelle anwählen, an der ein Registerinhalt geändert werden soll. Geben Sie dazu einfach hinter dem entsprechenden Registernamen auf dem Bildschirm die neuen Hex-Werte ein.

Die Angaben am rechten Bildschirmrand zeigen den momentanen stand des Carry- und Zero- Flags, das I-Register und die selektierte Speicherbank.

Mit der ENTER-Taste wird das Programm ab der Adresse im PC als Unterprogramm ausgeführt.

Die MODE-Taste beendet den Test-Modus, und nach der Abfrage der Startadresse befinden Sie sich wieder im Hexdump-Modus.

Bei einer Fehleingabe (z.B. ein Nicht-Hex-Zeichen im HEX-Modus) ertönt ein, zweistimmiges Signal, und die Eingabe wird nicht angenommen. Der gleiche Ton wird bei einem Diskettenfehler, erzeugt. In diesem Fall startet der Editor neu.

Zur Realisierung:

Auf ein komplettes Listing des 0774 Bytes langen Programms möchte ich verzichten.

Daten bereich :

FF40-FFS9 : Zwischenspeicher für eine Zeile
FF5A-FF66 : Eingabemeldung
FF67-FF6B : Voreingabe Adresse
FF6C-FF6D : Voreingabe Bank

Variablen-Bereich:

FFE0/ : Stackpointer-Zwischenspeicher
FFE2/ : Rücksprungadresse für Tasteneinsprung
FFE4 : Ireg für Test-Modus
FFE5 : Status
 b0: ASCII-Mode
 b1: Disk
FFE6 : log. Cursor x
FFE7 : log. Sektornummer
FFE8-FFFF : Register-Zwischenspeicher für Test-Modus
IYreg : mom. Editieradresse

Tabelle für Sprünge und Unterprogramme:

FF6E : Startadresse des Programms
FF71 : Hauptprogramm
FF74 : Anzeigenroutine Zeile Areg,ab (DE)
FF77 : Zeile,Spalte auf der Anzeige anwählen
FF7A : Ein Zeichen ausgeben
FF7D : Areg >ASCII >(HL)
FF80 : ASCII (HL) >Areg
FF83 : Eine Taste,Umschaltung
FF86 : Error Beep
FF69 : Adresse abfragen
FF8C : Einen Block abspeichern
FFBF : log. Block Nr. Areg lesen/schl-eiben
FF92 : Zeile Areg aufbauen und anzeigen
FF95 : log. Spalte >physikal. Adresse >HLreg

Tastentabelle Hexdump-Mode (Code,Sprungadresse):

FF98 : Pf-R
FF9B : Pf-U
FF9E : Pf-L
FFA1 : Pf-O
FFA4 : MODE
FFA7 : OFF
FFAA : DEF C
FFAD : DEF S
FFB0 : DEF G
FFB3 : selbstdefinierte Taste

FFB6 : Einsprung für F0A7
FFB9 : Einsprung für Tastenroutine
FFBC : Register anzeigen

Tastentabelle Test-Mode:

FFBF : ENTER
FFC2 : Pf-R
FFC5 : Pf-U
FFC8 : Pf-L
FFCB : Pf-0

Absolute Adressen:

FFCF/ : Hexdump-Tastenrücksprung
FFD2/ : Test-Tastenrücksprung
FFD5/ : Pointer für Registernamen
FFD8/ : Rücksprungadresse in den Monitor

FFDA : Sprung in den BASIC-Editor
FFDD : ursprüngl. Wert für Einsprungstelle (33,33,C9)

E) A N H A N G

1.REFERENZLISTE

<<< Literaturverweise >>>

- Rodney Zaks:

Programmierung des Z80

bei: Sybex, Düsseldorf

- Sharp

Pocket Computer

PC-1600

Technical Reference Manual

bei: Sharp Corporation, Osaka (Japan)

<<< Assembler >>>

- DOLMAS

Label-Macro-Assembler-Debugger

Autor: David von Oheimb

Vertrieb durch: Klaus Ditze, Weilerswist

3. VOLLSTÄNDIGE TASTEN- UND ASCII- TABELLE

Diese Tabelle können Sie auch zur Umrechnung von Hexadezimal nach Dezimal und umgekehrt verwenden.

16X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	016	032	048	064	080	096	112	128	144	160	176	192	208	224	240	
Null Repeat			0	@	P	'	p	C	E	ä	·	·	·	·	·	·
001	017	033	049	065	081	097	113	129	145	161	177	193	209	225	241	
1 SHIFT F1		1	A	Q	a	q	ü	æ	í	·	·	·	·	·	·	·
002	018	034	050	066	082	098	114	130	146	162	178	194	210	226	242	
2 SML F2		"	2	B	R	b	r	é	æ	ó	·	·	·	·	·	·
003	019	035	051	067	083	099	115	131	147	163	179	195	211	227	243	
3 CTRL F3		#	3	C	S	c	s	â	ô	ú	·	·	·	·	·	·
004	020	036	052	068	084	100	116	132	148	164	180	196	212	228	244	
4 KBIT F4		\$	4	D	T	d	t	ä	ö	ñ	-	-	-	-	-	-
005	021	037	053	069	085	101	117	133	149	165	181	197	213	229	245	
5 BS F5		%	5	E	U	e	u	à	ò	N	=	+	F	σ	J	
006	022	038	054	070	086	102	118	134	150	166	182	198	214	230	246	
6 F6		&	6	F	V	f	v	â	û	·	·	·	·	·	·	·
007	023	039	055	071	087	103	119	135	151	167	183	199	215	231	247	
7		'	7	G	W	g	w	ç	ù	·	·	·	·	·	·	·
008	024	040	056	072	088	104	120	136	152	168	184	200	216	232	248	
3 CL		<	8	H	X	h	x	ê	ÿ	·	·	·	·	·	·	·
009	025	041	057	073	089	105	121	137	153	169	185	201	217	233	249	
9 RCL		)	9	I	Y	i	y	ë	Ö	-	-	-	-	-	-	-
010	026	042	058	074	090	106	122	138	154	170	186	202	218	234	250	
1 CA		*	:	J	Z	j	z	è	Ü	-	-	-	-	-	-	-
011	027	043	059	075	091	107	123	139	155	171	187	203	219	235	251	
2 DEF		+	;	K	L	k	l	ï	Ç	·	·	·	·	·	·	·
012	028	044	060	076	092	108	124	140	156	172	188	204	220	236	252	
3 INS		,	<	L	\	l	l	î	£	·	·	·	·	·	·	·
013	029	045	061	077	093	109	125	141	157	173	189	205	221	237	253	
ENTER DEL		-	=	M	J	m	j	ï	¥	ö	·	·	·	·	·	·
014	030	046	062	078	094	110	126	142	158	174	190	206	222	238	254	
ON RESERU		.	>	N	n	n	Ä	R	<	=	·	·	·	·	·	·
015	031	047	063	079	095	111	127	143	159	175	191	207	223	239	255	
OFF MODE		/	?	O	o	·	·	·	·	·	·	·	·	·	·	·

5.TOKENTABELLE

Hier die Tokentabellen von den zwei mir bekannten ROM-Versionen, alphabetisch und nach Tokencodes geordnet.

In der ersten Tabelle sind für jeden Befehl der Token, der Code, der den Befehlstyp angibt, und der Befehlsname aufgelistet.

Dahinter stehen die Sprungadressen mit der dazugehörigen Bank.

(1 oder 2 Adressen je nach Befehl, für beide ROM-Versionen).

F120	21	ABS	0:0202	0:0202
F262	01	ACNV\$	0:0000	6:80FF
F174	21	ACS	0:0202	0:0202
F280	6D	ADIN	3:402D	3:402D
F25A	20	AIN	3:405D	3:405D
F25C	00	ALARM\$	3:4057	3:4057
F150	22	AND	0:01F6	0:01F6
F2BC	09	AOFF	0:0000	0:0000
F2BF	00	APPEND	0:24E8	0:2413
F180	28	AREAD		0:1CAB0:1B06
F181	28	ARUN	0:1CAB	0:1B06
F2B0	00	AS	0:24E8	0:2413
F160	01	ASC	6:8105	6:8105
F173	21	ASN	0:0202	0:0202
F125	21	ATN	0:0202	0:0202
F2B6	05	AUTO	3:4003	3:4003
F182	2D	BEEP	3:4054	3:4054
F290	2D	BLOAD	3:40B4	3:40B4
F0B3	AD	BREAK	0:6868	0:66F9
F291	2D	BSAVE	3:40B1	3:40B1
F282	AD	CALL	0:0047	0:0047
F0B2	28	CHAIN	5:68FF	5:68FF
F163	21	CHR\$	6:80E4	6:80E4
F187	2D	CLEAR	0:5623	0:555B
F089	35	CLOAD	5:6974	5:6974
F292	2D	CLOSE	3:4090	3:4090
F088	2D	CLS	0:5767	0:5700
F0B5	2D	COLOR	4:70D7	4:70D7
F2A3	6D	COM	3:4027	3:4027
E858	05	COM\$	3:4060	3:4060
F081	00	CONSOLE	0:0000	0:0000
F183	24	CONT	3:4000	3:4000
F293	2D	COPY	3:40A8	3:40A8
F17E	21	COS	0:0202	0:0202
F095	30	CSAVE	5:669C	5:6699
E680	2D	CSIZE	4:7295	4:7295
F084	2D	CURSOR	0:5652	0:558A
F18D	28	DATA	0:1CAB	0:1806
F257	80	DATE\$	3:404E	3:404E
F165	21	DEG	0:0202	0:0202
F18C	30	DEGREE		0:01AE0:01A4
F289	05	DELETE	0:4D9F	0:4CD0
E857	20	DEV\$	0:0202	0:0202
F18B	2D	DIM	0:5C9B	0:5C3A
F166	21	DMS	0:0202	0:0202
F274	01	DSKF	0:00FD	0:00FD
E884	00	DTE	0:0000	0:0000
F283	2A	ELSE	0:1C8C	0:1BB7
F18E	28	END	0:6368	0:6315

F221 21	EOF	0:00FA	0:00FA
F2B7 AD	ERASE	0:5DE2	0:5D81
F053 00	ERL	6:8111	6:8111
F052 00	ERN	6:810E	6:810E
F1B4 6C	ERROR	3:403F	3:403F
F178 21	EXP	0:0202	0:0202
F0B0 00	FEED	0:0000	0:0000
F098 2D	FILES	3:40A5	3:40A5
F1A5 28	FOR	0:5965	0:5904
F093 2D	GCURSOR	3:40D5	3:40D5
E682 2D	GLCURSOR	4:735B	4:735B
F194 2A	GOSUB	0:58ED	0:588C
F192 2E	GOTO	0:586A	0:580D
F09F 2D	GPRINT	3:40D2	3:40D2
F186 3D	GRAD	0:01AE	0:01AE
E681 2D	GRAPH	4:713E	4:713E
F265 A1	HEX\$	6:80E7	6:80E7
F196 28	IF	0:57BD	0:5756
F294 2D	INIT	3:4087 5:5C04	3:4087 5:5C04
F15C A5	INKEY\$	0:6731	0:65C2
F266 21	INP	6:80D8	6:80D8
F091 2D	INPUT	3:7E02 0:38BF	3:7E02
E859 01	INSTAT	3:4066	3:4066
F267 07	INSTR	6:80F0	6:80F0
F171 21	INT	0:0202	0:0202
F268 81	JS\$	0:0000	6:80EA
F284 AD	KBUFF\$	0:677E	0:660F
F263 01	KCNV\$	0:0000	6:8102
F269 22	KEFT\$	0:0000	6:80F6
F26A 01	KEN	0:0000	6:80F3
F285 6D	KEY	3:4021	3:4021
F286 2D	KEYSTAT	0:67B2	0:6643
F26B 23	KID\$	0:0000	6:80F9
F26C 22	KIGHT\$	0:0000	6:80FC
F287 2D	KILL	3:4099	3:4099
F264 01	KN\$	0:0000	6:B0ED
F0A5 2D	LCURSOR	4:7344	4:7344
F17A 22	LEFT\$	0:0202	0:0202
F164 01	LEN	0:0202	0:0202
F198 2C	LET	0:60EC	0:608B
F0B6 2D	LF	4:72CD	4:72CD
F0A0 2D	LFILES	3:40A2	3:40A2
F099 2D	LINE	0:6615	3:40C9
F090 05	LIST	3:4006	3:4006
F0B7 2D	LLINE	4:7486	4:7486
F0B8 2D	LLIST	3:7E06 4:78DE	3:7E06 4:78DE
F126 21	LN	0:0202	0:0202
F295 2D	LOAD	3:40AE	3:40AE
F272 21	LOC	0:00F4	0:00F4

F1B5	2D	LOCK	0:685D		0:66EE
F223	21	LOF	0:00F7		0:00F7
F177	21	LOG	0:0202		0:0202
F0B9	2D	LPRINT	3:7E04	4:75EA	3:7E04 4:75EA
F288	2D	MAXFILES	3:408A		3:408A
F158	20	MEM	0:0202		0:0202
F08F	25	MERGE	5:6969		5:6969
F17B	23	MID\$	0:0202		0:0202
F250	A2	MOD	6:80C9		6:80C9
F2B3	25	MODE	0:56FD		0:5696
F297	2D	NAME	3:409C		3:409C
F19B	95	NEW	3:4009		3:4009
F19A	28	NEXT	0:5A57		0:59F6
F16D	21	NOT	0:0202		0:0202
F19E	00	OFF	0:24E8		0:2413
F19C	28	ON	3:401E		3:401E
F296	2D	OPEN	3:408D		3:408D
F19D	2D	OPN	3:4084		3:4084
F151	22	OR	0:01F9		0:01F9
F28A	AD	OUT	0:004A		0:004A
F2BE	00	OUTPUT	0:24E8		0:2413
E880	2D	OUTSTAT	3:4069		3:4069
E381	2D	PAPER	4:73BF		4:73BF
F2B8	24	PASS	3:400C		3:400C
F1A2	2D	PAUSE	0:6337		0:62E4
F2B1	2D	PCONSOLE	3:406C	4:71B8	3:406C 4:7188
F26D	21	PEEK	6:80DB		6:80DB
F26E	22	PEEK#	6:80DE		6:80DE
F2A0	ED	PHONE	3:4039		3:4039
F15D	20	PI	0:34A4		0:33DF
F0A4	2D	PITCH	4:7412		4:7412
F168	26	POINT	6:80E1		6:80E1
F28C	2D	POKE	0:004D		0:004D
F28B	2D	POWER	0:005C		0:005C
F09A	2D	PRESET	3:40CF		3:40CF
F097	2D	PRINT	3:7E00	0:38B9	3:7E00
F09B	2D	PSET	3:40CC		3:40CC
F2B4	2D	PZONE	3:4063	4:7150	3:4063 4:7150
F1AA	3D	RADIAN	0:01AE		0:01AE
F1A8	3D	RANDOM	0:01AE		0:01AE
F2A4	2D	RCVSTAT	3:4075		3:4075
F1A6	28	READ	0:6118		0:60D1
F1AB	AB	REM	0:1C8C		0:1BB7
F2B5	05	RENUM	3:401B		3:401B
F1A2	2A	RESTORE	0:562C		0:5564
F28D	2A	RESUME	0:5809		0:5878
F28E	28	RETI	0:5BAE		0:5B4D
F199	28	RETURN	0:5937		0:58D6
F172	22	RIGHT\$	0:0202		0:0202

EB5A	20	RINKEY\$	0:0202	0:0202
F0BA	2D	RLINE	4:748A	4:748A
E2A9	2D	RMT	5:7007	5:7007
F12C	21	RND	0:0202	0:0202
E685	2D	ROTATE	4:70E2	4:70E2
F1A4	26	RUN	0:585A	0:5801
F256	24	RXD\$	6:8114	6:8114
F299	2D	SAVE	3:40AB	3:40AB
F298	2D	SET	3:409F	3:409F
E882	2D	SETCOM	3:406F	3:406F
E886	2D	SETDEV	3:4072	3:4072
F129	21	SGN	0:0202	0:0202
F17D	21	SIN	0:0202	0:0202
F2A1	2D	SNDBRK	3:407B	3:407B
F2A2	2D	SNDSTAT	3:4078	3:4078
E684	2D	SORGN	4:714A	4:714A
F061	21	SPACE\$	0:0202	0:0202
F16B	21	SQR	0:0202	0:0202
F167	21	STATUS	0:0202	0:0202
F1AD	00	STEP	0:24E8	0:2413
F1AC	28	STOP	0:635E	0:630B
F161	21	STR\$	0:0202	0:0202
F0BB	2D	TAB	4:7344	4:7344
F17F	21	TAN	0:0202	0:0202
E883	00	TERMINAL	0:0000	0:0000
F0BC	2D	TEST	4:7144	4:7144
E686	2D	TEXT	4:7133	4:7133
F1AE	2A	THEN	0:24E8	0:2413
F15B	20	TIME	3:4048	3:4048
F258	60	TIMES	3:4033	3:4033
F2BA	A5	TITLE	3:4018	3:4018
F1B1	00	TO	0:24E8	0:2413
E885	00	TRANSMIT	0:0000	0:0000
F1B0	2D	TROFF	3:400F	3:400F
F1AF	2D	TRON	3:4012	3:4012
F1B6	AD	UNLOCK	0:6860	0:66F1
F085	20	USING	0:654F	0:64FC
F162	81	VAL	0:0202	0:0202
F1B3	2D	WAIT	0:62E6	0:6293
F261	A1	WAKE\$	0:0041	0:0041
F087	2D	WDTH	0:0000	0:5635
F18A	3D	XCALL	0:01AE	0:01AE
F251	00	XOR	0:24E8	0:2413
F16F	21	XPEEK	0:0202	0:0202
F16E	21	XPEEK#	0:0202	0:0202
F1A1	3D	XPOKE	0:01AE	0:01AE
F1A0	3D	XPOKE#	0:01AE	0:01AE
F0B4	00	ZONE	0:0000	0:0000

E381 PAPER	F0B9 LPRINT	F18E END	F26C KIGHT\$
E680 CSIZE	F0BA RLINE	F192 GOTO	F26D PEEK
E681 GRAPH	F0BB TAB	F194 GOSUB	F26E PEEK#
E682 GLCURSOR	F0BC TEST	F196 IF	F221 EOF
E684 SORGN	F150 AND	F198 LET	F222 LOC
E685 ROTATE	F151 OR	F199 RETURN	F223 LOF
E686 TEXT	F158 MEM	F19A NEXT	F224 DSKF
E7A9 RMT	F15B TIME	F19B NEW	F280 ADIN
E857 DEV\$	F15C INKEY\$	F19C ON	F282 CALL
E858 COM\$	F15D PI	F19D OPN	F283 ELSE
E859 INSTAT	F160 ASC	F19E OFF	F284 KBUFF\$
E85A RINKEY\$	F161 STR\$	F1A0 XPOKE#	F285 KEY
E880 OUTSTAT	F162 VAL	F1A1 XPOKE	F286 KEYSTAT
E882 SETCOM	F163 CHR\$	F1A2 PAUSE	F287 KILL
E883 TERMINAL	F164 LEN	F1A4 RUN	F288 MAXFILES
E884 DTE	F165 DEG	F1A5 FOR	F28A OUT
E885 TRANSMIT	F166 OMS	F1A6 READ	F28B POWER
E886 SETDEV	F167 STATUS	F1A7 RESTORE	F28C POKE
F052 ERN	F168 POINT	F1A8 RANDOM	F28D RESUME
F053 ERL	F16B SQ	F1AA RADIAN	F28E RETI
F061 SPACES	F16D NOT	F1AB REM	F290 BLOAD
F084 CURSOR	F16E XPEEK#	F1AC STOP	F291 BSAVE
F085 USING	F16F XPEEK	F1AD STEP	F292 CLOSE
F087 WDT	F170 ABS	F1AE THEN	F293 COPY
F088 CLS	F171 INT	F1AF TRON	F294 INIT
F089 CLOAD	F172 RIGHT\$	F1B0 TROFF	F295 LOAD
F08F MERGE	F173 ASN	F1B1 TO	F296 OPEN
F090 LIST	F174 ACS	F1B3 WAIT	F292 NAME
F091 INPUT	F175 ATN	F1B4 ERROR	F298 SET
F093 GCURSOR	F176 LN	F1B5 LOCK	F299 SAUE
F095 CSAVE	F172 LOG	F1B6 UNLOCK	F2A0 PHONE
F092 PRINT	F178 EXP	F250 MOD	F2A1 SNOBRK
F098 FILES	F179 SGN	F251 XOR	F2A2 SNOSTAT
F099 LINE	F17A LEFT\$	F256 RXD\$	F2A3 COM
F09A PRESET	F17B MID\$	F257 DATE\$	F2A4 RCVSTAT
F09B PSET	F17C RND	F258 TIME\$	F2B1 PCONSOLE
F09F GPRINT	F17D SIN	F25A AIN	F2B3 MODE
F0A0 LFILES	F17E COS	F25C ALARM\$	F2B4 PZONE
F0A4 PITCH	F17F TAN	F261 WAKE\$	F2B5 RENUM
F0A5 LCURSOR	F180 AREAD	F262 ACNV\$	F2B6 AUTO
F0B0 FEED	F181 ARUN	F263 KCNV\$	F2B7 ERASE
F0B1 CONSOLE	F182 BEEP	F264 KN\$	F2B8 PASS
F0B2 CHAIN	F183 CONT	F265 HEX\$	F2B9 DELETE
F0B3 BREAK	F186 GRAD	F266 INP	F26A TITLE
F0B4 ZONE	F187 CLEAR	F267 INSTR	F28C AOFF
F0B5 COLOR	F18A XCALL	F268 JS\$	F2BD AS
F0B6 LF	F18B DIM	F269 KEFT\$	F2BE OUTPUT
F0B7 LLINE	F18C DEGREE	F26A KEN	F29F APPEND
F0B8 LLIST	F18D DATA	F26B KID\$	

6.I/O-ADRESSEN (PORTS)

Die Ports, deren Bedeutung schon in den entsprechenden Kapiteln abgehandelt wurde, sind hier nicht näher beschrieben (z.B. die LCD-Ports 50-5B).

Abkürzungen:

w=Write (OUT)

r=Read (IN)

00-0F:gesperrt

10-1F:kompatibel zum PC-1500-Portbaustein LH-5811 (#FF00-#FF0F)

14w :Teiler-Reset

15r :U-Register (serieller Empfang)

16w :L-Register (serieller Ausgang)

17w :F-Register (Modulation des seriellen Ausgangs)

18 :OPC-Register (Puffer für PC-Port)
 b7:Buzzerleitung (aktiv=0)
 b6:Buzzer an

19 :G-Register (Wait-Kontrolle,Baudrate)

1Aw :MSK-Register (Interrupt-Maske)
 b3:Empfängerflag RD
 b2:Senderflag TD
 b1:PB7 (BREAK-Taste)
 b0:IRQ (PC-1500-Peripherie)

1Ar :Lesen von
 -Interruptanforderungen:
 b7:RD
 b6:TD
 b5: PB7
 b4:IRQ
 -Interrupt-Maske MSK
 b3-1-0:wie 1Aw

1B :IF (Interrupt-Flipflop,bleibt bis zum Löschen gespeichert)
 b3-b0:wie 1Aw

1C :DDA-Register (übertragungsrichtung für PA-Port)
 Ein gesetztes Bit bedeutet OUTPUT

1D :DDB-Register (übertragungsrichtung für PB-Port)

1E :OPA-Register (PA-Port)
 wird für den Tastaturstrobe benutzt

1F :OPB-Register (PB-Port)
 b7:BREAK-Taste (PB7)
 b6:Tastenstrobe Nr.8 (PB6)
 b5:1/64-sek-Impuls (PB5)
 b2:Cassetten-Empfang (PB2)

20-23:UART (serielle/parallele Schnittstelle,COM1,COM2)
20 :serielle Ein- und Ausgabe
21 :parallele Ausgabe
(als Befehlsport für den Timer genutzt)
22w :Parameter (z.B. Baudrate) setzen,Registeradresse Port 23
22r :Status der seriellen Schnittstelle
23w :Befehl und Registeradresse
23r :Status der parallelen Schnittstelle

24-27:wie 20-23 (unvollständig decodiert)

28-2F:Slot 2
28w :Bank für S2 (0..7)

30-3F:interne Kontrollregister
31 :Baikswitching
32r :Interrupt-Ursache
33r :Eingang vom Timer
34 :Interrupt-Maske für LH-5803
35 :Interrupt-Maske für Z80
37w :b4:Clock für LCD-Prozessor an
37r :Eingang von der Tastaturmatrix
38w :Kontrolle zwischen den CPUs umschalten
39w :lo-Byte für indirekte Interrupt-Adresse (IM2)
3Aw :unbenutzt
3Bw :unbenutzt
3Cw :SLOTMAP (Umadressierungen)
In F08D zum Lesen gespeichert
3D :bankselect BASIC-ROM und JAPAN-ROM
b2:4000-7FFF Bank 3 normales-ROM-Modul (sonst BASIC-ROM)
b1:8000-BFFF Bank 4 JAPAN-ROM
In F07D zum Lesen gespeichert
3E-3F :unbenutzt

40-4F :reserviert

50-5F :Anzeige
50w :Befehl I+II
51 :unbenutzt
52w :Daten I+II
53 :unbenutzt
54w :Befehl II
55r :Status II
56w :Daten II
57r :Daten II
58w :Befehl I
59r :Status I
5Aw :Daten I
5Br :Daten I

5C-5F :unbenutzt

60-6F :Slot 2

70-77:Floppy II,sonst wie 78-7F

78-7F:Floppy I

78w :Befehl (40=lesen,60=schreiben,A0=formatieren)

78r :Motor- und Diskettenstatus
b7:Motor noch nicht angelaufen
b6:kein Schreibschutz
b3:Diskette im Laufwerk

79w :Sektor

7Aw :Motor
b7:Motor an

7Ar :Status
b6:Disk gewechselt
b1:Ready
b0:Fehler (invertiert)

7B :Daten lesen/schreiben

7C-7F:unbenutzt

80-83:Plotter/Centronics-Interface

Plotter:

80 :Interrupt-Maske
b5:wenn Kopf am linken Anschlag
b4:Printschalter
b3:Disk
b2:Reverse Paperfeed-Taste
b1:Paperfeed-Taste
b0:Farbwechsel-Taste
81w :Disk-Reset,wenn b0 zurückgesetzt
81r :Interrupt-Ursache,sonst wie 80
b7:Cassettenrecorder-Empfangsleitung
82 :b7:Cassettentrecorder-Interface an
b5:RMT OFF
b4:RMT ON
b3-b0:Stiftmotor
83 :b7-b4:y-Motor
b3-b0:x-Motor

Centronics-Interface:

80 :Interrupt-Maske
b3:Disk
81w :Disk-Reset,wenn b0 zurückgesetzt
81r :b3:Disk-Interrupt
b0:Busy
82 :Handshake-Leitungen
b5:EXPRM
b4:STROBE
83 :parallele-Daten

83-FF :reserviert

7.SYSTEM-RAM (MEMORY-MAP)

Hier, soweit bekannt, der Inhalt der Systemadressen.
**=PC-1500-Notation (hi vor lo,b15 invertiert)

<<< F000 >>>

F000
:

F001-F012:LH-5803
F001 :
F002 :Mode
 b4:Parameterübergabe
F003 :
F004 :Flags
 b4:H b3:V b2:Z b1:IE b0:C
F005 :A-A (Errorcode)
F006/:X-HL
F008/:Y-DE
F00A/:U-BC
F00C/:PC
F00E :b0:PV
F00F/:?-IX
F011/:?-IY

F013
F014

F015-F01E:"S1:"
F015 :Start hi
F016 :log. Bank Prg.-Modul (FE=Speichererweiterung,FF=unbenutzt)
F017 :Ende hi
F018 :log. Bank
F019/:BASIC-Start
F01B :log. Bank
F01C/:BASIC-Ende
F01E :log. Bank

F01F-F02b:"S2:"
F01F :Start hi
F020 :log. Bank Pi-g.-Modul (FE=Speichererweiterung,FF=unbenutzt)
F021 :Ende hi
F022 :log. Bank
F023/:BASIC-Start
F025 :log. Bank
F026/:BASIC-Ende
F028 :log. Bank

F029- F02C : "S0:"
F029 :Start hi
F02A :log. Bank
F02B :log. Bank BASIC-Start
F02C :log. Bank BASIC-Ende

F02D :Maxfiles

F02E-F04F:ROM-Module Arbeitsbereich-Adressen
F02E/:Standard-System-RAM
F030/:EXROM 1
F032/:EXROM2
F034/:CE-1F01A
F036/:CE-16000P/CE-1600E
F038/:CE-1600F
F03A/:EXROM6
F03C/:EXROM7
F03E/:EXROM8
F040/:EXROM9
F042/:EXROMA
F044/:EXROMB
F046/:CE-1600P (Cassette)
F048/:EXROMD
F04A/:EXROME
F04C/:COM-Puffer
F04E/:FBCs (Mit MAXFILES reserviert)

F050-F053:ROM-Programm-Modul S1
F050 :Start hi F051
:Länge hi
F052/:BASIC-Start

F054 :Start hi RAM S1
F055 :Größe in 2KB

F056-F059:ROM-Programm-Modul S2
F056 :Start hi
F057 :Länge hi
F058/:BASIC-Start **

F05A :Start hi RAM S2
F05B :Größe in 2KB

F05C-F078:Anzeige I
F05C :Scrollpointer
F05D :Status 1 schnell blinken (INSERT-Mode)
b3:Zeichensatz ASC<20 gültig>
b2:PC-1500-Zeichensatz (MODE 1)
F05E :Status 2
b1:LCD Zugriff
b0:Cursor im Moment sichtbar

F05F :Cursorpostion y
F060 :Cursorpostion x
F061/:Zeichentabelle ASC<20 Start
F063 :Bank
F064/:Zeichentabelle ASC>=80 Start
F066 :B.ank
F067 :Cursorstatus
 00:aus
 01:Unterstrich
 02:Rechteck
 03:Space
F068 :Blinkcounter
F069-F078:vom Cursor verdeckter Anzeigen-Bereich

F079-F08C:Tastatur I
F079 :Status 1 .
 b7:SHIFT,SML aus
 b6:Repeat schnell
 b5:
 b4:erster Repeat
 b3:alle Tasten Repeat
 b2:Repeat an
 b1:Click an
 b0:Tasteninterrupt-Sperre
F07A :Status 2
 b4:ALARM\$-Interrupt beim Warten auf Taste
 b3:Tastaturpuffer-Zugriff
 b2:Warten auf Taste
 b1:Tastencode >=80
 b0:Nicht die gleiche Taste
F07B :Status 3
 b3:Keystat 2
 b2:Power Off Off
 b1:Power Aoff 1
 b1:Keystat 1
F07C :Tasten von Keystat 1/2
 b7:Pf-U
 b6:Pf-0
 b0:OFF

F07D :Inhalt von Port 3D
F07E :Maske für Timer-Interrupt

F07F :Schreibpointer für Tastaturpuffer (0..3F)
 b7:Puffer voll
F080 :Lesepointer
F081 :letzte gedrückte Taste
F082 :Repeat-Counter
F083 :letzte Taste incl. Umschaltfunktion (für Repeat)
F084/:SHIFT-Tastencode-Tabelle Start
F086 :Bank
F087/: KBII-Tabelle Start
F089 :Bank
F08A/:SHIFT-KBII-Tabelle Start
F08C :Bank

F08D :Inhalt von Port 3C

F08E-F09C:Anzeige II (Line,Pset,Gprint)

F08E/:x1

F090/:y1

F092/:x2

F094/:y2

F096 :Set-Code

Wert	Line,Pset	Gprint
00	s	Set
01	R	Or
02	x	Xor

F097/:Line Punktcode

F099/:Gcursor x

F09B/:Gcursor y

F09D-F0A1:Anzeigenroutinen-Einsprung

F0A2-F0A6:Printroutinen-Einsprung

F0A7-F0AB:Tastaturroutinen-Einsprung

F0AC/:Auto Power Off-Counter

F0AE-F0BA:"ROM-Bit"

F0AE/:angeschlossene PC-1600-Peripherie

F0B0/:PC-1500-Peripherie

F0B2/:ROM-Interruptmaske 0

F0B4/:ROM-Interruptmaske 1

 b7 von F0B5:User-Interrupt

F0B6/:ROM-Interruptmaske 2

 b7 von F0B7:User-Interrupt

F0B8 :Counter für Batterietest

F0B9/:Leere Batterien der Peripheriegeräte

 b7 von F0B9:interne Batterie

F0BB :01:keine OFF-Taste bei Print möglich

F0BC :Bank User-Interrupt 2,vC0

F0BD/:Adr

F0BF :Bank User-Interrupt 1,vC0

F0C0/:Adr

F0C2 :Bank ON TIME\$-Interrupt,vC0

F0C3/:Adr

F0C5 :Bank ALARM\$-Interrupt,vC0

F0C6/:Adr

F0C8 :Bank WAKE\$(0)-Interrupt,vC0

F0C9/:Adr

F0CB :Bank APOTEST-Routine

F0CC/:Adr

F0CE-F0D0:RST 08

F0D1-F0D3:RST 28

F0D4-F0D6:RST 30

F0D7-F0D9:RST 38

F0DA/:Stackpointer bei Auto Power Off
F0DC :Bank BASIC-Interpreter-Start
F0DD/:Adr

F0DF-F121:Tastatur II

F0DF-F11E:Tastaturpuffer

<<< F100 >>>

F11F :Bank Tastencode-Tabelle (Normaltasten)
F120/:Adr

F122 :JAPAN-Mode
 b1:ROM Bank 4,8000 (OUT 3D,2)
 b0:16*8-Matrix o.k.

F123-F126:Codierte RAM-Modul-Bestückung (für Reset)
F123 :S1
 b7-b4:Startadr b15-b12
 b3-b0: 0:"P"
 1:"F"
 2:"S"
 F:"M"

F124 :Header-Checksumme S1
F125-F126:S2,sonst wie F123-F124

F127 :Timer-Interruptanforderung für BASIC
 b7:WAKE\$(0)
 b6:ON TIME\$
 b5:ALARM\$

F128
F129 :

F12A-F12E:Timer
F12A :gesetzte Interrupts
 b7:WAKE\$(0)
 b3:ON TIME\$
 b5:ALARM\$
 b1:Keystat 1

F12B :Signale
 b3:Stundensignal
 b2:Wake-Beep
 b1:Wake
 b0:WAKE\$(1)

F12C :b0:ON ADIN-Interrupt gesetzt
F12D :ON ADIN Unterergrenze
F12E :ON ADIN Obergrenze

```

F12F-F17F:serielle Schnittstellen
F12F-F131:Setcom COM1
F132-F134:Setcom COM2
F135-F137:INIT-String I
F138-F13A:INIT-String II
F13B :Pconsole Zeilenlänge    COM1
F13C :                          COM2
F13D :Pzone COM1
F13E      COM2
F140/:Startadr Puffer
F142/;Endadr Puffer
F144/:Lesepointer
F146/:Schreibpointer
F148 :Sndstat COM1
F149 :Sende-Timeout COM1
F14A :                          COM2
F14B :Rcvstat COM1
F14C :Empfangs-Tineout COM1
F14D :                          COM2
F14E :b6:Setdev COM2
      b2:PO
      b0:KI
F14F :b5:RTS an
      b3:SNDBRK
      b1:DTR an
F150 : Empfangsfehler
      b7:BRK
      b2:Puffer voll
      b1:Timeout
F151 :b3:XOFF empfangen
      b0:Wortlänge 7 + SHIFT IN
F152 :b7:XON gesendet
      b6:SHIFT OUT empfangen
      b3:XOFF gesendet
F156 :Timeout-Counter,zählt aufwärts im 0.5-sek-Takt
F158-F17F:Standard-Empfangspuffer
F180/:
F182-F196:Plotter I
F1b2 :Pitch x
F183 :Pitch y
F184 :b4:Lline Typ 20d
      b3-b0:Color
F185 :Pconsole Zeilenlänge
F186 :Lcursor/Tab
F187 :Pconsole EOL-Code
      b7:LF
      b6:kein CR
F188/:Papierposition (in 0.1mm) y-Richtung
F18A-F18C:Sprung für Systemroutinen
F18B/:Lline x-Differenz
F18D/:Lline y-Differenz
F18F/:max. x-Position
F191/:min. x-Position
F193 :Wert in Port 80
    
```

F194 :rel. Position in der Pzone
F195/:Zwischenspeicher x-Position für Stiftwechsel

F197-F1B1:Cassetterecoder-Interface
F197 :Länge 0-high
F198 :Länge 0-low
F199 :Länge 1-high
F19A :Länge 1-low
F19B/:Checksunmme der Bits
F19D :Länge Pause 1
F19E :Länge Pause 2
F19F/:Länge Header-Vorspann 1
F1A1 :Länge Header-Vorspann 2
F1A2 :Länge Header-Vorspann 3
F1A3/:Länge PRG-Vorspann 1
F1A5 :Länge PRG-Vorspann 2
F1A6 :Länge PRG-Vorspann 3
F1A7 :Länge Pause 3
F1A8 :Grenze 0/1-Interpretation beim Empfang
F1A9 :b7:high-low-übergang auswerten
F1AA/:Mindestlänge Vorspann 1 beim Empfang
F1AC :
F1AD :Wert im Point 35
F1AE :b7:Daten doppelt senden
F1AF :Länge Vorspann 4
F1B0 :Status
 b4:keine ASCII-Datei
 b3:schreiben
 b2:Ende beim Lesen erreicht
 b1:erster Block
 b0:geöffnet
F1B1 :

F1B2-F21C:BASIC 1
F1B2/:Verarbeitungspointer ROM-Modul-Befehle
F1B4 :
F1B5/:AUTO tom. Zeile
F1B7/:AUTO Inkrement
F1B9 :
F1BA :
F1BB :
F1BC :Mode
 b7:gesetzt,wenn b3 von Port 1F gesetzt ist (Normalfall)
 b6:Mode 1
 b1:break Off
 b0:Fehlerbehandlungs-Routine läuft
F1BD :Slot beim Zeilensuchen
F1BE :Bank des gefundenen Peripiterie-Befehls
F1BF : "ROM-Bit" für Peripherie-Token-tabelle
 b7 von F1C0:PC-1500-Token-tabelle

F1C1-F1CE:logische Banks

F1C1 :CURRENT
F1C2 :SEARCH START
F1C3 :SEARCH FOUND
F1C4 :MERGED
F1C5 :PREVIOUS I
F1C6 :PREVIOUS II
F1C7 :BREAK I
F1CB :BREAK II
F1C9 :ERROR I
F1CA :ERROR II
F1CB :ON ERROR I
F1CC :ON ERROR II
F1CD :RESTORE
F1CE :INTERPRET

F1CF-F1D4:BASIC-Interrupts

F1CF/:STOP oder ON
F1D1/:ON
F1D3/:Anforderung gespeichert

F1D5 :Title

F1D6-F1DA:Information für jede log. Bank
 b7:Progi-aeim-Modul
 b5-b4:Phys. Portadresse (Wert für Port 31)
 b1:Slot 2
 b0:Slot 1

F1DB-F21C :BASIC-Stack II

<<< F200 >>>

F21D-F31C:Eingabepuffer

<<< F300 >>>

F31D-F3C6:BASIC 2

F31D-F35E:Stacks für BASIC-Interrupts
F31D-F32C:Stack für Interrupt-Masken
F32D-F334:Max. 8 aktivierte Interrupts
F335 ;Stackpointer für Interrupt-Masken
F336 :
F337-F35E;pro aktiviertem Interrupt 5 Byte

F35F/;befehlstoken

F361-F3B5:

F3B6 :

F3B7-F2BE:

F3BF/:letztes ROM-Bit (für Reset)

F3C1-F3C3: COM-Einsprung

F3C4-F3C5:

F3C6 :Statusanzeige 2

b7:KBII

b3:S

b2:

b1:CTRL

b0:BATT

<<< F400 >>>

F3C7-F4FF:DEFAULT FCB

F3C8-F446:List-Puffer

<<< F500 >>>

F500-F5FE:Z80-Stackbereich F5FF :

<<< F600 >>>

F600-F64D:PC-1500 LCD 1

F64E :Statusanzeige 0

b7:DEF

b6:I

b5:II

b4.III

b3:SML

b2:

b1:SHIFT

b0:BUSY

F64F :Statusanzeige 1

b6:RUN

b5:PRO

b4:RESERVE

b2:RAD

b1:G

b0:DE

F650-F6FF:Standardvariablen E\$-O\$

F650 :E\$

F660 :F\$

F670 :G\$

F680 :H\$

F690 :I\$

F6A0 :J\$

F6B0 :K\$

F6C0 :L\$

F6D0 :M\$

F6E0 :N\$

F6F0 :O\$

<<< F700 >>>

F700-F74F:PC-1500-LCD II

F750-F7FF:Standardvariablen P\$-Z\$ F750

:P\$

F760 :Q\$

F770 :R\$

F780 :S\$

F790 :T\$

F7A0 :U\$

F7B0 :V\$

F7C0 :W\$

F7D0 :X\$

F7E0 :Y\$

F7F0 :Z\$

<<< F800 >>>

F800-FB4F:LH-5083 Stackbereich

F850-F85F:

F860-F8BF:BASIC 3

F860-F863:PC-1500-Programmmodul

F860 :Start hi.b7 invertiert

F861/:BASIC-Start **

F863 :Memory-Start *.hi,b7 invertiert

F864 :Ende Feldvariablen hi,b7 invertiert

F665/:BASIC-Start S0

F867/:BASIC-Ende S0

F869/:MERGED Start

F86B :b7:RMT OFF

B0:BEEP OFF

F86C-F870:

F871 :Wait

00:WAIT

02:WAIT x

03:WAIT 0

80:WAIT x,S

F872/:Wait Operand

F874 :

F875 :PRINT-Cursor x/GCURSOR x

F876-F87F:

F880 :Editor-Status
 b7:Error-Sperre
 b6:Cursor an
 b5:Ergebnis wird angezeigt
 b4:List
 b3:Reserve-Tastenstring
 b2:Cursor als blinkender Doppelpunkt
 b1 :
 b0:Systemmeldung
F881 :
F882 :
F883/:Variablenpointer für CALL
F885 :Variablenlänge
F886/:Variablenpointer für INPUT,hi vor lo
F888 :Variablenlänge
F869 :
F88A :Programmstatus
 b7:Stop,Break
 b6:Print
 b5:Input
 b3:Auto
F88B :
F88C :Argumentcounter für Funktionen
F88D :02:TRON
F88E :00:TROFF FF:TRON 03:-Trace
F88F :Ausgabepuffer-Pointer
F890 :FOR-Stackpointer
F891 :GOSUB-Stackpointer
F892 :DATA-Stackpointer
F893 :Operatoren-Stackpointer
F894 :Stringpointer
F895-F896:USING
F895 :b7: ^
 b5: +
 b4: ,
F896 :Vorkomma
F837 :String
F898 :Nachkomma+1
F899/:Feldvariablen-Start
F89B :Error Code

F89C/;CURRENT (mom. Verarbeitung) Zeile
F89E/:Top **
F8A0/:PREVIOUS (aktuelles Programm) Adr
F8A2/:Zeile
F8A4/;Top **
F8A6/:SEARCH Adr
F8A8/:Zeile
F8AA/:Top **
F8AC/:BREAK Adr
F8AE/:Zeile
F8B0/:Top **
F8B2/;EkR(JR Adr
F8B4/:Zeile **
F8B6/:TOP **

```
F8B8/:ON ERROR Adr **  
      B7 von F8B8:ON ERROR GOTO 0  
F8BA/:Zeile  
F8BC/:Top **  
F8BE/:READ-Pointer (DATA)
```

```
F8C0-F8FF:Standardvariablen A$-D$  
F8C0 :A$  
F8D0 :B$  
F8E0 :C$  
F8F0 :D$
```

```
<<< F900 >>>
```

```
F900-F9CF:Standardvariablen A-Z
```

```
F900 :A  
F908 :B  
F910 :C  
F918 :D  
F920 :E  
F928 :F  
F930 :G  
F938 :H  
F940 :I  
F948 :J  
F950 :K  
F958 :L  
F960 :M  
F968 :N  
F970 :O  
F978 :P  
F980 :Q  
F988 :R  
F990 :S  
F998 :T  
F9A0 :U  
F9A8 :V  
F9B0 :W  
F9B8 :X  
F9C0 :Y  
F9C8 :Z
```

```
F9D0 :  
F9D1 :OPN-Code  
F9D2-F9DF:
```

F9E0-F9F9:Plotter 2
F9E0/:Stiftposition x (in 0.1mm)
F9E2/:x-überschreitungen
F9E4/:y-überschreitungen
F9E6/:Paper Limit 2
F9E8/:Paper Limit 1
F9EA/:Graph Position x
F9EC/:Graph Position y
F9EE :Stift-Status
 b7:Stift gesenkt
 b5:Tastensperre
 b3:Lline:Stift heben
 b2:y-überschreitung
 b1:x-überschreitung
 b0:Stift senken
F9EF :Status 1
 b7:Hardware zurückgesetzt
 b6:Stiftwechsel
 b5:Batterie leer
 b3:BRK-Taste verboten
 b1:Paper R
 b0:Graph
F9F0 :Nachzähler
 b7-b4:y
 b3-b0:x
F9F1 :Motorstand
 b7-b4:x
 b3-b0:y
F9F2 :Wert für Port 83
F9F3 :Stiftmotor
 b7-b4:Nachzähler
 b3-b9:Motorstand
F9F4 :b7-b4:Rotate
 b3-b0:Schreibrichtung
F9F5 :b3-b0:Csize
F9F6 :Lline b7-b4:Counter
 b3-b0:Typ
F9F7 :Pzone
F9F8 :Status 2
 b6:Papier uneingeschränkt vorwärts
 b5:Kein in-die-Mitte-fahren des Stifts bei Paperfeed-Taste
 b3:Kein Hardware-Reset nach Power Off
 b2:Wert im Point 35 bleibt
 b1:Lline Typ 20d
 b0:Csize bleibt
F9F9 :Wert im Port 35

F9FA-F9FE:
F9FF :00:LOCK FF:UNLOCK

<<< FA00 >>>

FA00 :BASIC-Register XX
FA08 :ZZ
FA10 :YY
FA18 :UU
FA20 :VV
FA28 :WW
FA30 :SS
FA38-FAFF:BASIC-Stack

<<< FB00 >>>

FB00-FB07:RANDOM-Zahl
FB08-FB0F:
FB10-FB5F:Stringpuffer

FB60-FBAF:Cassetteninterface-Puffer
FB60-FB8F:Header-Puffer
FB90/:Startadresse
FB92 :Bank
FB93/:Endadresse
FB95 :Bank

FB96/:Pointer für Adressen-Längen-Block
FB98-FBAF:Adressen-Längen-Block
 (Wert für Port 31 b7=letzter Block
 Startadresse,Länge)

FBB0-FBFF:BASIC-Puffer für Tokenizer

<<< FC00 >>>

FC00-FCAF:Dateien-Arbeitsspeicher
FC00-FC07:RAM-Disk S1
FC00/: LOGFORM-Adr
FC02/:FAT-Adr
FC04 :FAT-Checksumme
FC05 :Media ID
FC06 :
FC07 :Zahl der offenen Dateien

FC08-FC0F:RAM-Disk S2
FC08/ :LOGFORM-Adr
FC0A/ :FAT-Adr
FC0C :FAT-Checksumme
FC0D :Media ID
FC0E :
FC0F/ :Zahl der offenen Dateien

FC10/:Bytezahl von alten Sektor
FC12/:Zahl der neuen sektoren
FC14/:bytezähl vom neuen Sektor
FC16 :Device Name
 01:X 02:Y 00:S1 01:S2 04:COM 05:COM1 06:COM2
FC17-FC21:Filename+Extension
FC22 :Clustercounter beim Suchen
FC23 :erster freier Cluster
FC24/:Disk:Puffercontrol-Adr RAM:Dirblock-Adr
FC26 :
FC27 :Dirblock-Nummer
FC28/:log. Blockzahl pro R/W-Vorgang
FC2A/:log. Sektorzahl LOW
FC2C : HIGH
FC2E/:Bytezahl LOW
FC30/: HIGH
FC32/:phys. Sektorenzahl
FC34/: Restbytes
FC36/:mom. Clusternummer
FC38 :Restsektoren
FC39/:mom. Schreib-Lese-Adr
FC3B :RAM-Disk Error Code
FC3C/:Zahl der bisherigen Cluster
FC3E/: FCB-Adr
FC40/:X/Y Control Adr
FC42/:LOGFORM-Adr
FC44/:FAT-Adr
FC46/: Schreib-Lese-FCB-Adr
FC48/:
FC4A :
FC4B :00:Lesen 01:Schreiben 02:Vergleichen
FC4C/:
FC4E/:mom. Sektorenzahl der Datei
FC50 :b7:neuer Sektor nötig
FC51 :Disk voll
FC52 :Sektor fertig
FC53-FC55:
FC56 :Wert in Port 31
FC57 :FAT-Schreibzähler
FC58 :Device Name für Filesuche
FC59 :
FC5A/:mom. Dirblock bei Filesuche
FC5C-FC99;
FC9A-FCA4:NAME neuer Name
FCA5-FCAF:alter Name

FCB0-FCCF:Für 16*8-Matrix

FCD0-FCD7:Für BASIC-Befehl LINE
FCD0/:x1
FCD2/:y1
FCD4/:x2
FCD6/:y2
FCD8-FCFF:

<<< FD00 >>>

FD00-FDFF:Für JAPAN-Mode

<<< FE00 >>>

FE00/:CURRENT Adr (BASIC-Verarbeitungspointer)

FE02-FE04:

FE05 :Länge der BASIC-Eingabe

FE06/:letzte eingegebene Zeile (Nummer)

FE08/:

FE0A/:Error-Adr bei direkten Befehlen

FE0C :01:-Trace

FE0D :log. Bank Error

FE0E-FE18:Editor

FE0E :

FE0F :

FE10 :linker Rand (Spaltenzahl-1)

FE11 :Cursorpos

FE12 :max. Cursorpos (0D-Position)

FE13 :min. Cursorpos (hinter Eingabemeldung)

FE14/:Startkoord. der mom. gelisteten Zeile

FE16 :Endpos. y

FE17 :

FE18 :Zahl der Zeichen bis zum rechten Rand

FE19-FE21:Zeilensuche

FE19/:3.Zeile vor der gefundenen Zeile (adr hi vor lo)

FE1B :log. Bank

FE1C/:2.Zeile

FE1E :log. Bank

FE1F/:Adr der Zeile vor der gefundenen Zeile

FE21 : log. Bank

FE22-FE31:Universal

FE32/:GCURSOR x

FE34/:GCURSOR y

FE36-FE39:Parameter-Ausdruck-Verarbeitung

FE3A :FOR-Stackpointer II

FE3B :GOSUB-Stackpointer II

FE3C-FE41:mit TITLE selektiertes BASIC-Programm

FE3C/:Start

FE3E :log. Bank

FE3F/:Ende

FE41 :log. Bank

FE42 :b7:Llist*
 b2:2-Zeilen-Modus
 b0:1.AUTO-Zeile
FE43 :b6:ON..
FE44/:NEW-Basisadr
FE46 :b2:CLS nach dir. Befehl
 b1:Gcursor-Wert in F875
FE47-FE55:Variablen-Verarbeitung
FE4C :1.Buchst. Variablenname
FE4D :2.Buchst.
FE50/:Zeilenindex DIM
FE52/:Spaltertindex

FE56/:Rücksprungadr nach INPUT/Stackpointer-Zwischenspeicher
FE56/:
FE5A :Cursor y
FE5B :Länge des ERROR-Strings

FE60 :Verschieben der Cursor-x-Position bei INPUT (00..19)
FE61 :Wert im Port 31

FE63 :b0:Zeile nach FB10-FBAB gesichert
FE64 :Restore S0-S2

FE65-FE67:LRINT COM
FE65 :Zahl der gesendeten Zeichen
FE66 :
FE67 :b0:Lprint,

FE68 :b5:INPUT Stringvariable
FE69 :Nummer der nach FB10-FBAB gesicherten Zeile

FE6A-FE7D:Centronics-Interface
FE6A :Pconsole
FE6B :Pzone
FE6C :
FE6D-FE6F:INIT-String I
FE70-FE72:INIT-String II
FE73 :Mom. Tab-Position
FE74 :Tabellennummer/Bank für Tabelle (FE75/) v80
FE75/:Startadr Tabelle
FE77 : b7:XOFF gesendet
 b1:EOL-Code CR+LF
 b0:EOL-Code nur LF, sonst nur CR
FE78 :b7:Einsprungadr gültig
FE79 :b1:Beim Listen von langen Zeilen kein Einrücken
 b0:EOL-Code bei Zeilenüberschreitung senden
FE7A/:Sprungadresse für BASIC-Befehle
FE7C/:Verarbeitungspointer für LPRINT

FE7E-FEFF :

<<< FF00 >>>

FF00-FF1F:WAKE\$(0)

FF20-FF3F:WAKE\$(1)

FF40-FFFF:

8.SPRUNGTABELLE

Hier noch einmal zur Übersicht die Routinen, deren Sprungadressen im ROM festgelegt sind. Genauere Informationen über Aufgabe und Parameter der Funktionen sind in den entsprechenden Kapiteln nachzulesen. XX=BASIC-Register X (siehe interne Datendarstellung)

```
0000 RESET
      RST 00:Warmstart
0005 APO
      Auto Power off
0008 RAM F0CE
      RST 08:Sprung nach F0CE
0015 EXROMCAL
      ROM-Bit Call, Bitmuster in DReg
0018 BANKJP
      RST 18:Bankjump
001B JPHLA
      Sprunge nach Adr HLreg, Bank Areg
0020 BANKCAL
      RST 20:Bankcall
0023 CHKMSG
      Für Check-Meldungen
0028 RAM F0D1
      RST 28:Sprung nach F0D1
002B APOTST
      Test, ob Auto Power on möglich ist
0030 RAM F0D4
      RST 30:Sprung nach F0D4 0035 POCLOS
      Alle offenen Dateien schließen+Power off
0038 RAM F0D7
      RST 38:Sprung nach F0D7 (z.B.für IMI)
003B SSLOTMP
      Setze Systemadr für Memory-und Programm-Module
003E SETPR SLOT
      Setze Systemadr F050-F05B
0041 WAKES$
0044 WAKES$=
0047 CALL
004A OUT
004D POKE
0050 JAPAN
      Nur für Japan-Modus
0053 JAPAN
      Nur für Japan-Modus
0056 EXROMAWK
      RAM-Zuteilung für ROM Bank3, adr 6000
0059 ungenutzt
005C POWER
```

```

005F USGCNT
    Konvertiere Zahl in XX >FA10-FA37 mit USING-Format
0062 TOKLEN
    Länge der ASCII-Zeichen des Tokens DReg >Areg
0066 NMI
    Nicht maskierbarer Interrupt (PC-1500-LCD Emulation)
006D SUSING2
    Setze USING-Parameter
0070-00B8 RAM-Calls der Anzeigenroutinen
00BB-00BE RAM-Calls der Printroutinen
00C1-00D6 RAM-Calls der Tastenroutinen
00D9 RAM-Call der COM-Routine CRCVI
00E5 BSPCTR
    Display an/aus
00E8 SLOTST
    Headertest
00EB PRTASTR
    Print -Areg, nur norm. Matrix
00EE PRTGSTR
    Gprint Zeichen (DE)-Areg
00F1 PRTBSTR
    Print -Areg
00F4 LOC
00F7 LOF
00FA EOF
00FD DSKF
0100 PRTANK
    Print Zeichen Areg
0103 PRTDBL
    Print Doppelzeichen DReg
0106 PRTASTRO
    Print -00, nur norm. Matrix
0109 SETANK
    Home+Cursor löschen
010C DBLMATR
    evtl. 16*8-Matrix
010F LWIDTH
    26d/40d-Zeichen-Modus
0112 CLS
    Bildschirm löschen
0115 CRSRSET
    Setze xy-Koordinaten
0118 CRSRPOS
    Hole Cursor-Koordinaten
011B RVCHR
    Invertiere Zeichen
011E CRSRSTAT
    Setze Cursor-Blinkstatus
0121 LINE
    Linie zeichnen
0124 BOX
    Fläche zeichnen
0127 DOTSET
    Punkt setzen
    
```

David von Oheimb: Das Systemhandbuch für den PC-1600

012A DOTREAD
1-Bit-Point

012D UPSCRL
Vorwärts scrollen

0130 DOWNSCRL
Rückwärts scrollen

0133 CGMODE
IBM/PC-1500-Zeichensatz

0136 CGSET80
Setze Zeichensatz ASC>=80

0139 SMBLREAD
Lies Statussymbole

013C SMBLSET
Setze Statussymbole

013F EKSSSTR
Space printen

0142 INSILN
Zeile einfügen

0145 ERSILN
Zeile löschen,nicht herausnehmen

0148 GCRSRPOS
Hole Gcursor

014B GCRSRSET
Setze Gcursor

014E PRTGCHR
Gprint 1 Zeichen

0151 PKTGSTRO
Gprint Zeichen -00

0154 PRTGPTN
Gprint

0157 CPY1500LCD
PC-1500 LCD Emulation

015A GPTNREAD
8-Bit-Point

015D SAVELCD
Kopiere Zeile ins RAM

0160 LOADLCD
Kopiere vom RAM in Zeile

0163 PRTBSTRO
Print -00

0166 KEYGET
Wartet auf Taste

0169 KEYGETR
Inkey

016C KBUFSET
Tastaturpuffer setzen

016F BREAKCHK
BRK-Abfrage

0172 CURUDCHK
Senkrechte Cursortasten

0175 KEYDIRECT
Mom. gedrückte Taste

0178 KEYSTRB
Tastenmatrix abfragen

017B KEYAUX
Eingabe-herkunft setzen

Anhang; 8. Sprungtabelle 3

017E KEYSTATSET
Click+Repeat
0181 KEYSTATREAD
Hole Keystat
0184 OFFCHK
OFF-Taste abfragen
0187 KEYGETND
Inkey,Puffer bleibt
018A BREAKRESET
Lösche Tastaturpuffer
018D MEMORYCHK
Test,ob Speicher vorhanden
0190 BANKSET
Bankselect
0193 BANKREAD
Lies selektierte Bank
0196 SLOTIMAP
Slot umadressieren 1
0199 SLOT2MAP
Slot unadreasieren 2
019C BANKJUMP
Sprung auf andere Bank
019F BANKCALL
Unterprogramm auf anderer Bank
01A2 RSLT150
Unterprogramm zum Ergebnis-Ausdruck auf den CE-150
01A5 FOTOK1500
Suche in PC-1500-Tokentabelle Befehle mit Token FO..
01A8 TOK1500
wie vor,aber sonstige Befehle.
01AB EXCOMM1500
Unterprograam für 01AE
01AE COMM1500
PC-1500-Befehle (z.B.XCALL)
01B1 BANKCPY
OUT 31,breg +LDIR
01B4 BOUT
Beep
01B7 SUUT
beep 1mal
01BA SWAIT
Wait
01BD BONOFF
beep on/off
01C0 BON
Beep on
01C3 BUZON
buzzer an
01C6 CALLH
Zweitprozessor aktivieren
01C9 P(A)OFF
Für Power Off/Aoff
01CC INT150
Interrupt-routine für PC-1500-Peripherie
01CF NMI

Anhang; 8. Sprungtabelle 4

```

01D2 USCNVHL
    Konvertiere Zahl im XX mit USING-Format,adr >HLreg
01D5 TIMER
    Suprozessor-Routinen
01D8 COM
    Ser. Schnittstellen-Routinen
01DB JAPAN
    Nur für Japan-Modus
01DE FILE
    Dateien-Routinen
01E1 WPCHK
    RAM-Disk Areg >Breg v80:Schreibschutz
01E4 SETMDSKSIZ
    Setze RAM-Disk-Größe F055/5B
01E7 INITMDSK
    Init Slot Areg,"F"
01EA POFF
    Für Power Off
01ED PAOFF
    Für Power Aoff
01F0 JAPAN
    Nur für Japan-Mode
01F3 ^(Potenz)
01F6 AND
01F9 OR
01FC CMPNUM
    Numerische Vergleiche
01FF CMPSTR
    String-Vergleiche
0202 FUNC
    BASIC-Funktionen
0205 & (Hex-Zahlen)
0208 CRSRVS
    Cusor evtl. invertieren (bei Interrupt)
020B CGNORM80
    Normaler Zeichensatz ASC>=80
020E STATSYMGET
    Lies Statussybole
0211 STATSYMSET
    Setze Statussymbole
0214 KEYSCAN
    Tastenabfrage bei Interrupt
0217 KEYNORM
    Normale Tastentabelle
021A +
021D -
0220 *
0223 /
0226 MOD
0229 \
022C unbenutzt
022F unbenutzt
0232 unbenutzt
0235 BANKPEEK
    Peek# (HL)

```

David von Oheimb: Das Systemhandbuch für den PC-1600

```

0238 BANKPOKE
    Poke#(HL)
023B SYSRESET
    Totalreset System-RAM
023E ASCBIN
    Integerzahl ASC (HL) >DReg
0241 BCBINS
    BCD XX >Zweierkomplement DReg
0244 GETRESULT
    Hole letztes Ergebnis (XX) als String >(DE)
0247 BCDBIN
    BCD XX >Integerzahl DReg
024A BI2BCD
    Integerzahl DReg >XX (BCD)
024D SCONT
    Setze CONT-Adresse
0250 SCA
    Für CA-Taste
0253 RESCA
    CA nach Reset
0256 TOKENIZE
    Verarbeitung des BASIC-Puffers
0259 ARRAYIND
    Berechne Index-Adresse für Feldvariable
025C EOCHK2
    Befehlsend-Code testen (CR,":",ELSE)
025F LINLIST
    Liest Zeile,die bei (HL) beginnt,mit der Cursor-Adresse BReg
    >Eingabepuffer F21D-F31C
0262 DATSKP
    überspringe ab Verarbeitungspointer HLReg den Rest der Zeile
0265 EOCHK3
    Teste Befehlsende
0268 BUFCLR
    Lösche Eingabepuffer,Cursorposition 00
026B EOCHK2
    Wie 025C
026E EOSCHK
    Teste Attribut von 027D auf Befehlsend-Code
0271 SERROR
    Setze Error-Adresse
0274 EXPRESS
    Numerischer oder String-Parameter >XX
0277 EXPREX
    Wie 0274,aber auch ein Teilausdruck wird ausgewertet
027A ALLCLOSE
    Alle geöffneten Dateien schliessen
027D GETCDI
    Lies Attribut (HL) >breg: b0:Einzelner ASCII-Code >Areg
    b1:BASIC-Token >DReg
    b2:Binär codierte Zeilennummer
>DReg
    b3:Doppel-ASCII >DReg
    b4:Einfüge-Zeichen
0280 BASTACKRES
    Reset Basic-Stack

```


0283 STRCNV
Integerzahl HLreg >String (DE)

0286 SLIST
Setze List-Start

0289 VARPAR
Variablen-Parameter (HL) >adr DReg

028C LINSRH
Suche Zeile DReg oder folgende >F8A6-F8A9

028F AUTORUN
Für Arun

0292 VARNAME
Variablen-Name (HL) >Codierung im DReg

0295 RUNSET
Setze Systemadressen für Run

0298 SUSING
Using (HL)

029B CMPTOK
Vergleiche DReg mit (HL)

029E Wie 005F

02A1 VARLET
Wert in XX in Variable kopieren

02A4 VARGET
Variable (HL) nach BASIC-Register (DE) kopieren

02A7 VARPEEK
Peek log, adr (HL) >Areg

02AA POPDAT
Hole Daten vom BASIC-Stack >(DE)

02AD VARLET2
Kopiere BASIC-Register (HL) nach Variable (DE)

02B0 VARPOKE
Poke log, adr (HL),Areg

02B3 VARADR
Adresse der Variable DReg >HLreg

02B6 TOKSRH
Token des Befehls ASC (HL) >DReg, Code >Creg, Länge >Areg

02B9 TOK2ASC
Token DReg >ASC (HL)

02BC COMMADR
Token DReg >Sprungadr HLreg,Bank-Out >Creg, Code >Areg

02BF EXROMCOM
Wie 02BC,aber nur ROM-Bit (F1BF/)

02C2 EXROMBANK
bankselekt des ROM-Moduls Creg

02C5 LINSRH2
Suche Zeile BCreg ab Adresse DReg

02C8 LINSRHREL
Suche Zeile BCreg relativ ab adr (F89C/) z.B. für Goto

02CB EXCOMMEXE
Führe Befehl Token (F35F/) eines ROM-Moduls aus

02CE LABSCH
Suche Label (HL) , Länge BCreg in alle Slots

02D1 LBSCH2
Wie 02CE aber, nur im mit Title selektieren Slot

02D4 TADCNV
bankselekt log. Bank Creg

Anhang; 8. Sprungtabelle 7

02D7 LOGADR
Korivertiere log. adr DReg >phs. adr BReg,Bank-Out >Areg

02DA LOGEND
Log. adr des BASIC-Program-Ende >DReg

02DD EXROM3WK
Teile ROM-Modul 3 DReg Bytes zu (für Eigenanwendung)

02DF EXROMWK
Teile ROM-Modul Creg DReg Bytes RAM zu

02E2 DIRCOMEXE
Führe direkten Befehl Token DReg aus,Verarbeitungsptr.HLreg

02E5 PRGEXE
Ausführung des Programms ab (FE00/),nach Aufruf von 02BC

02E8 INPUT1
Für Input

02EB INPUT1
Für Print 02EE NXTADR
>HLreg Startadresse der Zeile,die hinter der momentan durch
HLreg gegebenen Zeile steht,log. Bank >Areg

02F1 PASWPTST
Teste Password und Schreibschutz des mit Title sel.Slots >sc

02F4 PRGADR
Setze BASIC-Start- und Endadresse FE3C-FE41

02F7 EXROMCOMC
Suche den gleichen Token in anderem ROM-Modul

02FA RAMODSET
Setze Systemadressen der RAM-Module,Areg=MODE (00/01)

02FD ALLOFF2
Plottermotoren aus

0300 LPRTASTR2
Lprint Breg* (HL)

0303 BINBCD
DE >BCD,BASIC-Register (HL)

0306 EOF2
Für EOF

0309 JAPAN
Nur- für,
Japan-Mode

030C EDRESET
Editor-Reset

030F PRTWAITK
Print (DE)-0D und warte auf Taste

0312 BASPARES
BASIC-Parameter nach Reset setzen

Bei manchen ROM-Versionen folgen noch 4 weitere Sprünge, die nur für die Reset-Routine verwendet werden.