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 Adressenan8aben verstehen sich, wenn nicht anders vermerkt, grundsätzlich bexadezimal.

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 oinen Port (mittels des OUT-Befehls) verschiedene Speicherbausteine, die die gleiche Adresse belegen, selektiert.
Beim PC-1600 können durch den Port 31 die Adroßbereiche 4000-7FFF und 8000-BFFF unabhängig voneinander in 8 Bänke unterteilt werden.
Die Funktion des Ports läßt sich am besten binär beschreibon:

```
     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)
```

```
Boispiel:
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 diesom 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 gößer a1s 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:R0M

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 O
4000-7FFF:Bank 3
8000-BFFF:Bank 6
CO00-FFFF:Bank O,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 Schnitt-stellen,Zeichen- und Tokentabelle

Die ROMs der Peripheriegeräte (adr 4000-7FFF) sind modular organisiert. Bem 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 folgeden 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 Tokontabelle (+0036)
4015 : JP XXXX Sprung für File-Bearbeitung
```

"ROM-Bit:"

In F0AE stehen,bitweise codiert,die vorhandenen ROM-Hodule 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 DEreg die Größe in Bytes und im Creg die Nummer des Moduls gesetzt werden.

Mögliche Werte für Creg:

```
0O     : Standard-Systemspeicher
01-07  : ROM Bank 1-7,Start 4000 (EXROMI-EXROM7)
08-0E  : Ron Bank 1-7,Start 6000 (EXROM8-EXROME)
0F     : COM-Puffer ( wie INIT “COM:”,Puffergröße)
10     : Filepuffer ( wie MAXFILES=n ; DEreg = 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) : FOOO Ende+l
```

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-FO1E : S1

F01F-F028 : S2

F029-F02C : SO

F050-F053 : Sl,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,AOOO oder B000 beginnt. Entsprechend der Initialisierung mit INIT "Sn:" ergibt sich folgendes Bild (hier am Beispiel dor Adrosse 8000):

```
 "H"          "P"     "F"                        "S" (System

                                software)


 8000 : FF  55               55                          55
 8001 : FF  Start hi^7F      80                          00
 8002 : FF  BASIC-           neg.Checksum(8000-8O1F) 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               8l wenn RAM,01 wenn WP    82
 8008 : RESERVE-Bereich      log. Format
                             (siehe RAM-Disk)
```

Normalerweise sind den Slots folgende Adressen zugeordnet:

```
           Bank 0      Bank l     Bank 2      Bank 3
800O-BFFF Slot Ia      Slot Ib    Slot IIa    Slot IIb
```

Sie können durch zwei Routinen umadressiert werden:

```
0196 SLOT1MAP
     Areg=00 :Normalzustand
     Areg=01 :Slot lb wird zusätzlich dem Adreßbereich
              4000-7FFF der Bank I zugeordnet
0199 SLOT2MAP
     Areg=00 :Normalzustand
     Areg=01 :S1ot lla wird dem Adreßbereich
              0000-3FFF der Bank l zugeordnet
     Areg=02 :Slot llb wird dcm Adreßbereich
              0000-3FFF,Bank 1 zugeortnet
              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 8O) im Dreg
     >Areg= 00:Fehler
            FO:"P"
            Fl:"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$(O)
b7 : WAKE$(I)=CI
```

<<< RESET DER PERIPHERIE >>>

* Reset der wichtigsten Systenparameter (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?” angezelgt (siehe CHECK-Heldungen)
* Test,ob ein POWER ON mögilch 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-Konnando
- Geräte- und Modulbestückung gleich geblieben
- Alle Geräte haben volle Batterieen (CALL 4002 Areg=02)
- Die Routine APOTEST Bank (F0CB),adr (F0CC/) 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 folgendor Reihenfolge gesucht und, falls vorhanden, ausgeführt:

* Peripherlegeräto (z.B.Flopppy) (CALL 4002,Areg=05)
falls gefunden (CALL 4002,Areg=06)
* Systemsoftware (EPROM)
* Systemsoftware (RAM)
* Sprung n&ch Bank (F0DC),adr (F0DD/):
  Hier steht nach einom Totalreset die Startadresse des BASIC interpreters

Der BASIC-Interpreter setzt je nach Reset-Ursache seine System adressen und löscht bei einen Totalreset den Internen und den mit INIT “Sn:”,”M” selektierten Programspeicher.
Falls keine Meldungen (z.B.CHECK-Meldung) auszugeben sind, durch sucht er mit dem Sprung 400E die Peripheriegeräte nach der BASIC Datei AUTOBUN.BAS, lädt sie ggf. In den Speicher und startet sie Autonatisch.
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 Breich FA21-FA24 mindestens 1 Bit gesetzt,so wird eine entsprechende CHECK-Heldung angezeigt.

Dabei sind den Bits folgende Zeichen zugeordnet:

	b7	b6	b5	b4	b3	b2	b1	bO
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:

FFOO-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 a1so 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 aus-gelöst werden.

<<< INTERRUPT-URSACHEN >>>

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

b0 : Daten auf den Schnittstel1en empfangen Sie werden im Puffer abge1egt.

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 Kontro11e 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 Schnittste1len

- 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 Bi1d-schirm, 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 a1s Treiber für die PC-1500-Peripherie eingesetzt. Sein ROM entspricht weitgehend dem des PC-150O. 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-Reihenfo1ge ab a1s der Z80. Deshalb werden Adressen im Systemspeicher des reinen BASIC-Interpreters in der Reihenfolge hi-1o-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 g1eichen 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 getei1t:

	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	Zei1e 4	Zeile 4
y=0B..0F	Zeile 1	Zeile 1	Zeile 5	Zeile 5
y=10..17	Zei1e 2	Zei1e 2	Zei1e 6	Zei1e 6
y=18..1F	Zei1e 3	Zei1e 3	Zei1e 7	Zeile 7

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

Die Bytes mit x=FF (Block II,Zei1en 4,6,7) steuern die Status Symbole. Sie sind auch im System-RAM gespeichert:

	b7	b6	b5	b4	b3	b2	b1	b0	Zei1e
F3C7 :	KBI1				S	x	CTRL	BATT	4
F64F :		RUN	PRO	RESERVE		RAD	G	DE	6
F64E :	DEF	I	1I	I1I	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 jewei1s 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 : Zei1e 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 güItigen Daten;es kann erst ab dem zweiten Zugriff gelesen werden.

Beim Lesen bzw. Schreiben von Daten verden die X-Koardinaten automatisch inkrementiert, so daß, wenn die Koordinaten einmal angewählt sind, fortlaufend ge1esen 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 beispie1sweise stattdessen der Wert C8 ausgegeben, so scro1lt 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 Forme1 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.

Beispie1e:
- 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           0R 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 ma1 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-FIag 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 Zah1 vor der Adresse gibt die Bank an.

```
0,00E5 DSPCTR
       Areg v0l :Disp1ay an
       sonst    :DispIay aus

0,0112 CLS
       Cls,Status-Symbole bleiben
0,0145 ERS1LN
       CIear Zei1e Areg

0,0142 INS1LN
       Zei1e Areg einfügen
0,012D UPSCRL
       Scro1l vorwärts
0,0130 DWNSCRL
       Scro1l 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 1öschen, Areg v01   :40d-Zeichen-Mode
                                 Sonst :26d-Zeichen-Mode
```

```
0,0133 CGMODE
       Areg v0l :PC-1500-Zeichensatz
       sonst    :IBM-Zeichensatz
0,0136 CGSET80
       Zeichensatz ASC>80 setzen: Bank Areg,adr DEreg
0,020B CGNORM8O
       Zeichensatz ASC>8O wieder normal

0,020E STATSYMGET
       Lies Statusanzeige der Zeile Areg (4,6,7) >Areg
0,0211 STATSYSSET
       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 1öschen
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
       Ho1e mocentanen Printcursor >DEreg
0,0100 PRTANK
       Prine Zeichen Areg,Cursor (F05F/)
0,0103 PRTDBL
       Print Doppe1zeichen DEreg
0,0106 PRTASTR0
       Print String ab (DE) bis 0O, 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 ma1 SPACE printen
       sonst :Gcursor (DEreg,HLreg) Breg* SPACE printen
0,011B RVSCHR
       Cursor DEreg,Areg* Zeichen invertiere
```

0,0148 GCRSRPOS
Ho1e nom. Gprint-Cursor aus (F099/),(F09B/)>DEreg,BCreg
0,014B GCRSRSET
Gcursor (DEreg,BCreg)

0,014E PRTGCHR
Gprint Zeichen Areg auf mom. Gcursor
O,O151 PRTGSTR0
Gprint String (DE) bis O0
O,O0EE PRTGSTR
Gprint (DE)-Areg
6,809F CHARADR
Pixe1code 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/),(FO9B/) >Areg

O,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 SAVELCD
Save Bitmuster von Zeile Areg in (DE)-(DE+9B)
0,0160 LOADLCD
Load Bitmuster von (DE)-(DE+9B, nach Zei1e Areg
0,0157 CPY1500LCD
Kopiere unterste Zeile in den PC-15O0-kompatiblen LCD
Speicher F600-F64F,F700-F74F

Arbeitsspeicher:

F05C-F078
F08E-F09C

<<< ZEICHENGENERATOR >>>

Der PC-1600 besizt 3 Zeichentabellen, in denen die Daten punktweise in dem Format gespeichert sind, wie es auch beim BASIC-Befeh1 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 1iegt im RAM und kann geändert werden. Bank (F066),adr (F064/)

3. Die Zeichen ASC 00-1F sind nicht vorgesehen (b1ank),ihnen kann aber in (F061/),Bank (F063) eine eigene Tabelle zugeortnet werden, wenn in F05D das b3 gesetzt wird.

6.TASTATUR

<<< TASTATURMATRIX UND TASTATURPUFPER >>>

Die Tasten sind in einem 8*8-Matrix verschaltet. Dabei sind den 9 Spa1ten 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 a1s 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/lF:							
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. Desha1b ist entweder DI zu verwenden oder b0 von F079 zu setzen.

Die BRK-Taste hat eine Sonderste1lung:
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 zyk1ischen 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 ge1esen werden. Der Puffer steht in F0DF-F11E. Die Nunuer des ersten gültigen Wertes steht in F080 (Lesepointer). Sie wird zu F0DF dazugezäh1t und ergibt so die Adresse des ersten zu 1esenden Bytes. Der Schreibpointer steht in F07F.
Sind beide Pointer gleich, dann ist entweder der Puffcr leer, oder, fa11s b7 von F07F gesetzt ist,ist er mit 64d Bytes gefü1lt. 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 P1atz 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 1ese 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 Wiederho1funktion unt CIick,Code in Areg
b2:Repeat für a11e 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 g1eichzeitig gedrückt: >sc
0,0175 KEYDIRECT
Wie vor,aber außer der Matrix werden auch die durch Keystat se1ektierten 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 ,fa11s kein Wert vorhanden
0,0187 KEYGETND
Wie vor,jedoch bleibt der Inhalt des Puffers erhalten
6,80B7 KEYSCAN
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 Stantard-Tastentabellen
3,4054 KEYCODCNV
Konvertiert den Tastencode in Areg je nach der Stel1ung
der SHIFT-,SML-,DEF-,CTRL- und KBll-Umschalter >Areg

Arbeitsbereich:

F079-F08C
F0DF-F121

<<< TASTENTABELLEN >>>

Die Tabelle zur Umrechnung ter 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 a1s Index zur Startadresse addiert. Der Inhalt dieser neuen Adresse ist der neue, durch die Umscha1ttasten erzeugte, ASCII-Code. Die Tabellen ge1ten 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 BCreg (wie in BASIC)
>sc wenn die BRK gedrückt wird
Frequenz in Hz: 1300000d /(22d * Areg+ 166d)
Dauer in sek : BCreg/Frequenz
01B4 BOUT
DEreg * 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 BCreg (in l/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.

```
cO0 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$(0)
cO7 SRAIT
    On Time$, holen   >wie Wake$(0)
c08 SHA2T
    Alarm$ setzen, wie Wake$(0)
c09 SRA2T
    AlarmS holen      >wie WakeS(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 Inter
rupts müssen gesetzt sein:
   b7 : Wake$(0)
   b6 : On Tine$
   b5 : Alaram$
   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 mögilch 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
  bl : 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 1äßt 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 dle 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 1ch 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-Verglelchspuffer
```

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,CI
OUT 22,hl-Byte
- Status lesen
b2 : COM1 Ready
bl : Zeichen e~pfangen
b0 : COM2 Ready
Port 23 : - Befehle und Registeradre~sen ausgeben (slehe 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 : COMI gewählt
b2 : DSR
bl : CD
b0 : CTS (lnvertiert)
Der Zustand des CI-Eingangs kann mit der Timer-Routine SRINP (c13) gelesen werden >Areg:
b5 : Cl (lnvertiert)

<<< ROUTINEN >>>

Aufruf mit CALL 01D8, Funktitonscode 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 CWCOM
     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 : Partty 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 Enpfangsfehler >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
    bl : RTS low
    b0 : DTR low
cll CRINS
    Instat >Areg
    b5 : CI low
    b4 : DSX low
    b3 : CD low
    b2 : CTS low
    bl : 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 relovant
    b3 : CD ---,---
    b2 : CTS ---,---
         Breg: Timeout-Wert in 0.5-sek-Einheiten
               (00=unendllch)
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
cIB CCLRSB
    Sendestart
clC CCLRRB
    Empfangsstart (Puffer und Fehlercode löschen)
```

```
clD CREOL
    EOL-Code lesen >Areg
clE CRZONE
    Pzone lesen >Areg
clF CRHIDTH
    Pconsole Zeilenlänge lesen >Areg
```