BILDUNGSCOMPUTER

r o b o t r o n A 5 1 0 5

Beschreibung des Betriebssystems

VEB ROBOTRON-MESSELEKTRONIK >OTTO SCHÖN< DRESDEN
Lingnerallee 3, Postfach 211, Dresden, DDR-8012

Betriebssystembeschreibung

BILDUNGSCOMPUTER
r o b o t r o n A 5 1 0 5

I N H A L T

VEB ROBOTRON-MESSELEKTRONIK >OTTO SCHÖN< DRESDEN
Lingnerallee 3, Postfach 211, Dresden, DDR-8012

Elektronische Aufbereitung: Ulrich Zander <zander@felix.sax.de>

1. Einleitung

Die Systemsoftware des BICs umfaßt zwei selbständige Komponenten:
- ein diskettenorientiertes Betriebssystem (SCP)
- einen BASIC-Interpreter (RBASIC)

Bestandteil beider Komponenten ist ein gemeinsamer Kern, von Gerätetreibern, der die Arbeit beider Systeme mit der Hardware des BICs realisiert.
Sämtliche Bestandteile der Systemsoftware sind im ROM vorhanden. Während der Initialisierung beider Systeme werden jeweils einzelne Teile in den RAM kopiert und werden im RAM abgearbeitet. Andere Betriebssystemteile werden im ROM abgearbeitet. Bei der Nutzung von Teilen der Systemsoftware über die im SCP-Handbuch bzw. im Programmierhandbuch beschriebenen Nutzerschnittstellen (CCP, BDOS, BIOS bzw. RBASIC-Anweisungen) hinaus, z.B. durch Assemblerprogramme, werden Informationen über Speicheraufteilung, Einsprungtabellen, Schnittstellen der Unterprogramme benötigt.
Diese Informationen soll die vorliegende Beschreibung vermitteln.
Es sei jedoch darauf hingewiesen, daß neben der Kenntnis der Betriebssystemschnittstellen gute Kenntnisse in der Assemblerprogrammierung vorhanden sein müssen, um Bestandteile der Systemsoftware erfolgreich zu nutzen.

2. Erläuterungen zur Hardware

Die Speicherorganisation des BICs ermöglicht die Adressierung von maximal 256 KBytes. Davon sind 64 KBytes als RAM und 56 KBytes als ROM im BIC belegt sowie 24 KBytes wegen der Hardwaregegebenheiten nicht nutzbar. 112 KBytes können als Module ergänzt werden.
Der eingesetzte Rechnerbaustein, die CPU UA880D, kann aber nur auf 64 KBytes Hauptspeicher direkt zugreifen. Daher müssen mittels softwaregesteuerter Zusatzhardware jeweils die 64 KBytes des Gesamtspeichers aktiviert werden, die das laufende Programm benötigt.

2.2. Speicheraufbau

Der Hauptspeicher des BICs ist in 4 SLOTs eingeteilt, die jeweils 64 KBytes Speicher im Adreßbereich von 0H bis FFFFH umfassen. Der Adreßbereich von 64 KBytes wird in 4 PAGEs von je 16 KBytes eingeteilt(s. Bild 1).

	SLOT 0	SLOT 1	SLOT 2	SLOT 3
	ROM im CGG (fest eingebaut)	ROM in DSE (fest eingebaut) und Modulsteckplatz DSE	RAM im CGG (fest eingebaut)	Modulsteckplatz (oben) am CGG
PAGE 3 (C000H bis FFFFH	nicht nutzbar	frei für Modul	RAM	frei für Modul
PAGE 2 (8000H bis BFFFH	nicht nutzbar / ROM	frei für Modul	RAM	frei für Modul
PAGE 1 (4000H bis 7FFFH	ROM	ROM / ROM	RAM	frei für Modul
PAGE 0 (0000H bis 3FFFH	ROM	frei für Modul	RAM	frei für Modul

Bild 1: Speicheraufbau des BICs A5105

2.2. Hardware zur Speicherumschaltung

Für jede Page kann immer nur ein Slot aktiv sein (d.h., die CPU kann auf diesen Speicherbereich zugreifen), die anderen drei sind inaktiv. Das Aktivieren eines Slots hat demzufolge immer das Deaktivieren der anderen Slots zur Folge (Speicherumschaltung). Die aktiven Slots aller 4 Pages sind im Slotkontrollbyte enthalten, dessen Aufbau Bild 2 zeigt.

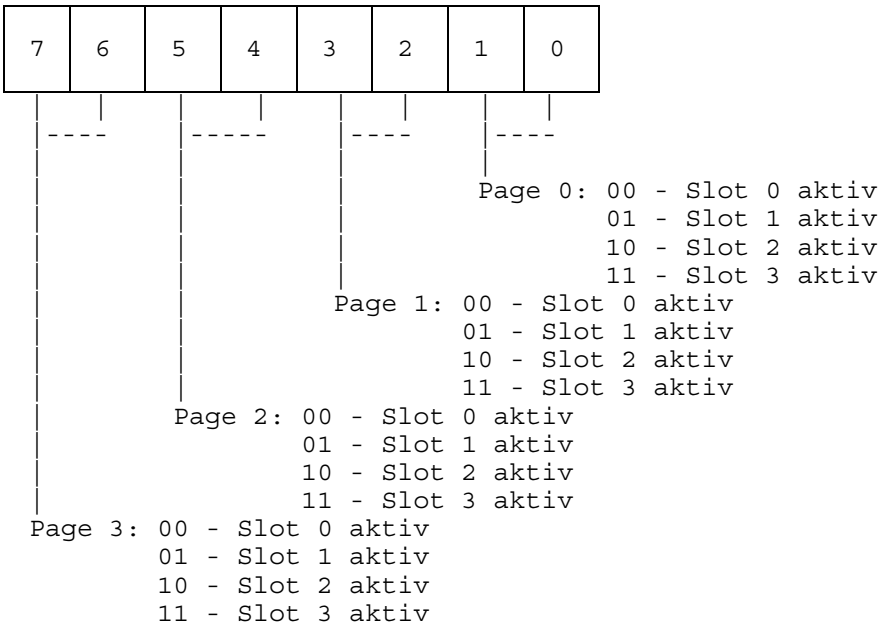


Bild 2:
Aufbau des Slotkontroll-Systems

Zur Speicherumschaltung dient der EA-Port mit der Adresse A8H im SVG-Schaltkreis U1520 FC007.
Bei Eingaben von diesem Port erhält man das aktuelle Slotkontrollbyte, welches die zu dieser Zeit aktiven Speicherbereiche enthält.
Wird ein Slotkontrollbyte auf diesen Port ausgegeben, so werden die im Slotkontrollbyte enthaltenen Slots aktiviert.

3. Software

3.1. RBASIC

3.1.1. Speicherkonfiguration

Bei der Initialisierung von RBASIC wird folgende Speicherkonfiguration eingeschaltet:

	SLOT 0	SLOT 1	SLOT 2	SLOT 3
PAGE 3 (C000H bis FFFFH)			RAM	
PAGE 2 (8000H bis BFFFH)			RAM	
PAGE 1 (4000H bis 7FFFH)	ROM			
PAGE 0 (0000H bis 3FFFH	ROM			

Bild 1: Speicheraufbau für RBASIC

Im ROM auf SLOT 0 befindet sich im Bereich 0...22FFH eine Sammlung von Unterprogrammen, die die Arbeit mit der Hardware realisieren, sowie Betriebssystemfunktionen ausführen.
Im Bereich 2300H...7FFFH liegt der RBASIC-Interpreter. Die erstgenannten Unterprogramme werden aus dem BASIC-Interpreter direkt aufgerufen. Dabei führt der Sprung direkt über eine Sprungtabelle, die im Adreßbereich 0...19AH liegt. Von einem geübten Assemblerprogrammierer können diese Unterprogramme in zusätzliche Nutzerprogramme einbezogen werden. **Eine Beschreibung der Funktion und der Schnittstellen der Unterprogramme enthält Anhang A.**

Die Nutzung eines solchen Unterprogramms über die Betriebssystemsprungleiste von einem RBASIC-Programm aus zeigt das folgende einfache Beispiel.

Auf der Adresse 5FH im SLOT 0 steht ein Sprung zur Routine 'Bildschirmmodus einstellen'. Vor deren Aufruf müssen in den Registern B und D der Modus und in den Registern C und E die Seitennummer eingetragen werden.

```
10   'Speicherplatz für Maschinenprogramm reserviert
20   '
30   CLEAR 200,&HD000
40   DEFUSR0=HD000
50   '
60   'Maschinenprogramm
70   '
80   DATA &h0E,&h0            :'LD C,0
90   DATA &h59                :'LD E,C    Seite 0
100  DATA &h6,&h8             :'LD B,8
110  DATA &h50                :'LD D,B    Modus 8
120  DATA &hCD,&h5F,&h0       :'CALL 5FH  einschalten
130  DATA &hC9                :'RET
140  '
150  'Maschinenprogramm laden
160  '
170  V%=&HD000
180  FOR I% = 0 TO 9
190  READ J%:POKE V% + I%,J%
200  NEXT I%
210  '
220  'Screen 0 einschalten, Zeichen ausgeben
230  '
240  SCREEN 0,0
250  CLS
260  PRINT STRING$ (120,"E")
270  PAUSE 100
280  '
290  'unter Nutzung der Routine SELSCR (5FH) auf Screen 8 umschalten
300  '
310  X = USR(0)
320  PAUSE 100
330  END
```

Das gerufene Unterprogramm vereinbart die Parameter Modus 8 und Seite 0. Danach ruft es die Routine zum Einschalten des vereinbarten Modus. Es erzielt die gleiche Wirkung wie die RBASIC-Anweisung SCREEN 8,0.

Weitere Bestandteile der Systemsoftware, die von RBASIC genutzt werden, befinden sich im ROM auf SLOT 0 PAGE 2 (Musikausgabe) und SLOT 1 PAGE 1 (Diskettenanschluß). Zur Nutzung dieser Teile muß eine andere Speicherkonfiguration eingestellt werden. (Bild 4)
Bei der Abarbeitung entsprechender RBASIC-Anweisungen wird diese Speicherkonfiguration automatisch eingestellt.

	SLOT 0	SLOT 1	SLOT 2	SLOT 3
PAGE 3 (C000H bis FFFFH)			RAM	
PAGE 2 (8000H bis BFFFH)	ROM (BASIC-Erw.)			
PAGE 1 (4000H bis 7FFFH)		ROM (Diskette)		
PAGE 0 (0000H bis 3FFFH	ROM			

Bild 4: Speicherkonfiguration für RBASIC-Erweiterungen

Wenn diese Erweiterungen von speziellen Anwenderprogrammen aus genutzt werden sollen, muß der Programmierer die Einstellung der entsprechenden Speicherkonfiguration selbst vornehmen.
Gleichfalls ist es möglich, die 32 KBytes RAM (PAGE 0 und 1 auf SLOT 2), welche vom BASIC-Interpreter nicht verwendet werden, für Maschinenprogramme, die im RBASIC eingebunden werden, nutzbar zu machen.
Dabei sind folgende Grundsätze zu beachten:

Genauso wie man den Ast, auf dem man selber sitzt, nicht absägen sollte, kann man den Speicherbereich, in dem das Programm zur Speicherumschaltung läuft, nicht deaktivieren. Man spricht in diesem Zusammenhang vom sogenannten Wurzelsegment (PAGE 3 auf SLOT 2), das immer aktiv bleibt.

Völlig sicher funktioniert die Speicherumschaltung erst dann, wenn auch
- der Stack
- die Interrupttabellen und die Interruptbehandlungsprogramme
- sowie die Betriebssystemvariablen

im Wurzelsegment liegen. Das ist für RBASIC gesichert, muß aber bei eigenen Maschinencodeprogrammen berücksichtigt werden.

Zur Realisierung der Speicherumschaltung gibt es die folgenden Möglichkeiten.

3.1.2. Komfortable Unterprogramme

Die folgenden Unterprogramme unterstützen die Arbeit mit mehreren Slots. Sie sind in der PAGE 0 auf SLOT 0 enthalten, der im RBASIC immer aktiv ist. Die Adressen der Unterprogramme sind von der RBASIC-Version unabhängig.
Einige Unterprogramme enden mit Interruptverbot (DI). Der Aufrufer muß selbst entscheiden, wann wieder EI gegeben werden kann.
Alle Unterprogramme sind reentrant, d.h., sie können sich selber aufrufen und auch in Unterprogrammen verwendet werden.

Lesen eines Bytes von einem nichtaktiven Slot

Adresse: 000CH

Input: A = SLOT (0...3)
HL = Speicheradresse (< C000H)

Output: A = gelesenes Byte
HL = unverändert

verwendete Register: AF, BC, DE

Funktion: - Lesen des Slotkontrollbytes
- Modifikation des Slotkontrollbytes entsprechend A und HL

- Speicher umschalten
- Byte lesen
- Speicher zurückschalten
- RET

Schreiben eines Bytes in einen nichtaktiven Slot

Adresse: 0014H

Input: A = SLOT (0...3)
HL = Speicheradresse (< C000H)
E = Byte

Output: HL = unverändert
E = unverändert

verwendete Register: AF, BC, D

Funktion: - Lesen des Slotkontrollbytes
- Modifikation des Slotkontrollbytes entsprechend A und HL
- Speicher umschalten
- Byte schreiben
- Speicher zurückschalten
- RET

Aufruf eines Unterprogrammes (mit CALL) bzw. eines Teilprogrammes (mit JMP) in einem nichtaktiven Slot

Achtung: Das Unterprogramm/Teilprogramm muß komplett in einer der 4 Pages liegen!

Variante 1: Adresse: 0159H

Input: Variable BASUB = SLOT von RBASIC
IX = Speicheradresse (< C000H)

Zweck: Rückkehr ins RBASIC (z.B. mit Fehlermeldung)

Variante 2: Adresse 0030H (RST 30H)

Input: 1. Byte nach Aufruf = SLOT (0...3)
2. und 3. Byte nach Aufruf = Speicheradresse (< C000H)

Zweck: Kurzform mit nur 5 Bytes
(RST 30H - Slot - Adresse - RET)

Variante 3: Adresse: 001CH

Input: High-Teil von IY = SLOT (0...3)
IX = Speicheradresse (< C000H)

Zweck: Register freihalten für Parameterübergabe

Variante 1 bis 3:

Output: DI

Register: alle (einschl. IX, IY und Zweitregister)

Parameterübergabe an bzw. von UP/TP kann über AF, BC, DE, HL erfolgen.

Funktion: - DI
- Lesen des Slotkontrollbytes
- Modifikation Slotkontrollbytes entsprechend Slot und Adresse
- Speicher umschalten
- Unter-/Teilprogramm starten
- nach RET im Unter-/Teilprogramm:

```
            Speicher zurückschalten
            Rückkehr in aufrufendes Programm
```

Speicherumschaltung

```
Achtung: Wurzelsegment liegt in PAGE 0 auf SLOT 0!

Adresse:  0024H

Input:    A  = SLOT (0...3)
          HL = Speicheradresse (3FFFH < HL < C000H)

Output:   HL = unverändert
          DI

verwendete Register: AF, BC, DE

Funktion: - DI
          - Lesen des Slotkontrollbytes
          - Modifikation des Slotkontrollbytes entsprechend A und HL
          - Speicher umschalten
          - RET
```

3.1.3. Speicherumschaltung durch direkten Portzugriff

Speicherumschaltung kann auch durch Lesen/Schreiben des EA-Ports des SVG erfolgen.

```
Beispiel:   IN    0A8H  ;A = aktuelles Slotkontrollbyte
            PUSH  AF    ;Slotkontrollbyte retten
         ;  ...         ;A modifizieren
            OUT   0A8H  ;Speicherumschaltung
         ;  ...
         ;  ...
            POP   AF    ;Slotkontrollbyte zurück
            OUT   0A8H  ;Speicher zurückschalten
```

Unter der Voraussetzung, daß jedes Interruptprogramm einen eigenen Stack anlegt und den Speicher wieder so verläßt, wie es ihn vorfindet, braucht die angegebene Befehlsfolge nicht unter DI abzulaufen.

3.2. SCPX 5105

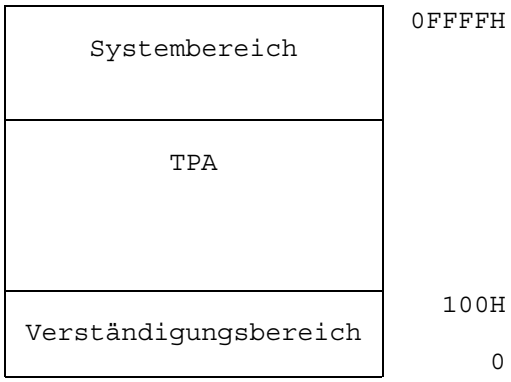
3.2.1. Systemaufbau

Beim Initialisieren des Betriebssystems SCPX 5105 wird der gesamte interne RAM zugeschaltet. (Bild 5)

	SLOT 0	SLOT 1	SLOT 2	SLOT 3
PAGE 3 (C000H bis FFFFH)			RAM	
PAGE 2 (8000H bis BFFFH)			RAM	
PAGE 1 (4000H bis 7FFFH)			RAM	
PAGE 0 (0000H bis 3FFFH			RAM	

Bild 6: Speicherkonfiguration für SCPX

Während des Systemanlaufs wird die für SCP typische Einteilung des 64KByte-RAMs vorgenommen.



Der Aufbau des Verständigungsbereiches ist in Anhang C des SCP-Handbuchs ausführlich beschrieben.
Im Systembereich befinden sich der System-RAM sowie CCP, BIOS und BDOS. Während der System-RAM im Anlauf vollständig initialisiert wird, werden von den 3 anderen Bestandteilen jeweils nur Köpfe (Sprungtabellen und Speicherschaltungen) im RAM aufgebaut. Die eigentlichen Routinen liegen im ROM. Dabei werden teilweise die gleichen Routinen (ROM-BIOS: Treiber für Diskette, Bildschirm, Tastatur sowie Drucker und ROM-BDOS: logische Verwaltung der Diskette) wie bei RBASIC genutzt.

3.2.2. Aufruf von Unterprogrammen in einem anderen SLOT

Analog zu RBASIC erfolgt die Speicherumschaltung bei Nutzung von CCP bzw. der BIOS- oder BDOS-Schnittstelle automatisch. Bei spezieller Nutzung von Routinen aus SCP-Anwenderprogrammen heraus muß der Programmierer die notwendigen Umschaltungen selbst vornehmen. Dabei gelten die gleichen Grundsätze wie bei RBASIC.

Im RAM (Wurzelsegment) befindet sich ein Spezialprogramm zum Aufruf von Unterprogrammen im Adreßbereich 0000H-BFFFH in einem beliebigen SLOT.
Der Adreßbereich C000H-FFFFH kann dabei nicht umgeschaltet werden. Das Spezialprogramm, das einen eigenen STACK anlegt, den SLOT umschaltet und das eigentliche Unterprogramm ruft, verändert **keine** CPU-Register!!!
Sämtliche Registerinhalte kommen im gerufenen Unterprogramm an, nach RET werden alle Registerinhalte an das Anwenderprogramm zurückgegeben.
Der Aufruf erfolgt mit:

```
      CALL  Spezialprogramm
      DEFW  Unterprogrammadresse
      DEFB  SLOT
```

Die Anfangsadresse des Spezialprogramms steht im RAM auf den Adressen 3EH/3FH. Das Spezialprogramm ist nur dann im RAM verfügbar, wenn vom CCP ein Anwenderprogramm aufgerufen wurde, es ist nicht verfügbar für nachladbare Treiber, die z.B. vom BIOS oder BDOS aufgerufen werden! (Grund: Doppelnutzung von CCP-RAM-Bereichen)
Außer dem Spezialprogramm sind über die Adresse 3EH noch weitere system-interne Routinen zugänglich. Die Anfangsadressen lassen sich aus der folgenden Tabelle ermitteln:

```
003EH:      DEFW  ANW   ;Anfangsadresse einer Tabelle im RAM
```

Tabelle im Wurzelsegment (Adreßbereich 0C000H-0FFFFH):

```
ANW:  JR ANCA     ;ANW+00 Spezialprogramm zum Unterprogrammaufruf
      JR ANRD     ;ANW+02 Spezialprogramm (RAM) Lesen eines Bytes
      JR ANWR     ;ANW+04 Spezialprogramm (RAM) Schreiben eines Bytes
```

```
      DEFW ANRD   ;ANW+06 gleiche Programme im ROM
      DEFW ANWR   ;ANW+08
      ...
      ...
      DEFW SLCTRL ;ANW+26 Adresse des Slotkontrollbytes
```

Erläuterungen zu den verwendeten Marken

ANCA Verzweigung zum Spezialprogramm

ANRD Programm zum Lesen eines Bytes aus einem beliebigen SLOT (Register B) von Adresse HL in Register A. Außer AF werden keine Register verändert. Nach dem Aufruf ist der Interrupt erlaubt (EI Interrupt wird abgearbeitet).

ANWR Programm zum Schreiben eines Bytes aus Register A auf Adresse HL in einem beliebigen SLOT (Register B). Außer AF werden keine Register verändert. Nach dem Aufruf ist der Interrupt erlaubt (EI wird abgearbeitet).

Achtung! Die Seite, in der sich das Programm ANRD bzw. ANWR befindet, darf nicht weggeschaltet werden! Aus diesem Grunde sind beide Programme jeweils in verschiedenen Seiten vorhanden. Beim Aufruf mit JR ANRD bzw. JR ANWR wird das Programm auf PAGE 3 (Adreßbereich 0C000H-0FFFH) erreicht. Ansonsten muß der Aufruf mit Bankumschaltung erfolgen. Die Adressen sind aus DEFW ANRD bzw. DEFW ANWR zu ermitteln. Die einzuschaltende Bank ist aus DEFW SLCTRL zu ermitteln.

SLCTRL Adresse im Wurzelsegment, auf der die systeminterne Speicherkonfiguration (Sloteinstellung) abgespeichert ist. Diese Speicherkonfiguration ist zm Aufruf eines Unterprogramms im ROM zu verwenden. Die Befehlsfolge zum Ermitteln und Einstellen dieser Konfiguration ist im folgenden Beispiel enthalten.

Beispiel:
Aufruf des Unterprogramms SELSCR in der ROM-Sprungleiste

```
            ASM (SCPX-RBWS) V 1/0         SEITE 1

                        ;SCREEN-Umschaltung
                        .Z80
0000  DD 2A 003E        LD    IX,(3EH)
                        ;
0004  DD 6E 1A          LD    H,(IX+26)   ;SLCTRL-Adresse holen
0007  DD 66 1B          LD    H,(IX+27)
000A  7E                LD    A,(HL)
000B  32 0021           LD    (ANSL),A
                        ;
000E  3A 005D           LD    A,(5DH)     ;1.Parameter aus CCP-Kommando
0011  E6 0F             AND   00001111B
0013  47                LD    B,A         ;aktiv-Modus
                        ;
0014  3A 006D           LD    A,(6DH)     ;2.Parameter aus CCP-Kommando
0017  E6 0F             AND   00001111B
0019  4F                LD    C,A         ;aktiv-Seite
                        ;
001A  50                LD    D,B         ;vis-Modus = aktiv-Modus
001B  59                LD    E,C         ;vis-Seite = aktiv-Seite
                        ;
001C  CD 0023           CALL  JPIX        ;Aufruf Spezialprogramm
001F  005F        ANADR:DEFW  005FH       ;Adresse von SELSCR
0021  00          ANSL: DEFB  0           ;SLOT wird mit SLCTRL überladen
                        ;
0022  C9                RET
                        ;
```

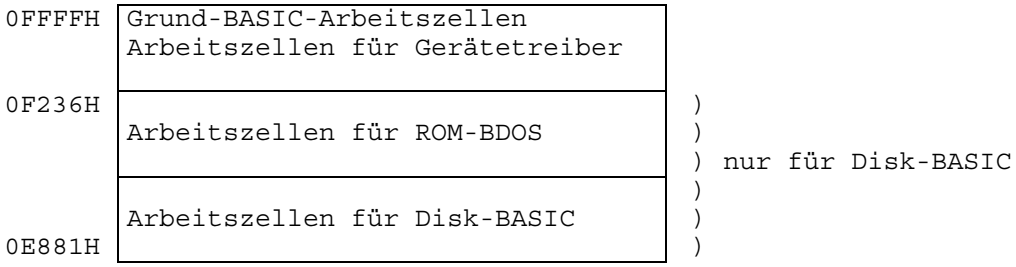
```
0023  DD E9       JPIX: JP    (IX)
                        ;
                    END
```

Der Anhang B enthält ein weiteres komplexeres Beispiel für die Nutzung von Betriebssystemteilen unter SCPX.

3.3. Systemspeicher

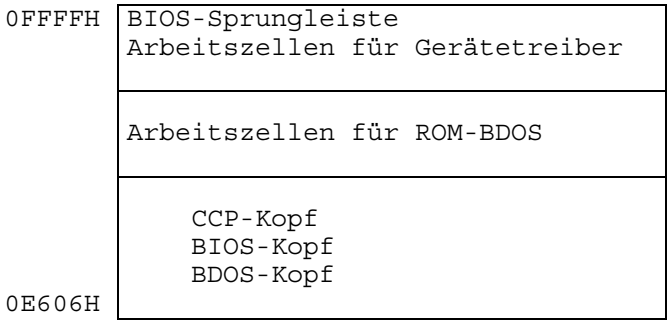
Der Systemspeicher liegt in beiden Betriebssystemen im oberen Teil des Wurzelsegments. Im Systemanlauf erfolgt jeweils die Initialisierung. Der Systembereich besteht aus einem Speicherbereich, der für beide Systeme gleich ist (z.B. Arbeitszellen für Gerätetreiber, Musikpuffer), sowie einem speziellen Bereich. Somit wird beim Übergang von einem Betriebssystem zum anderen ein großer Teil des Systemspeichers anders verwendet.

Aufteillung des Systemspeichers unter RBASIC
(Adreßangaben für Version 2.0)



Der Zeiger auf die erste Speicherzelle des Systembereichs befindet sich bei der Version 2.0 auf der Adresse 0FBB8H.

Aufteilung des Systemspeichers unter SCPX
(Adreßangaben für Version 2.0)



Auf den Adressen 6 und 7 im Verständigungsbereich steht die Anfangsadresse des Systembereichs (für Version 2.0 ohne Erweiterungen - Treiber, resident gemachte transiente Kommandos - 0E606H). Das ist zugleich der Haupteintrittspunkt für BDOS. Die Adresse des zweiten Sprungs in der BIOS-Sprungleiste (Warmstart) sind den Adressen 1 und 2 im Verständigungsbereich zu entnehmen (für Version 2.0 0FD03H).

Anhang A

Beschreibung der nutzbaren Unterprogramme
Stand: 30.11.89
Alle Adressen beziehen sich auf SLOT 0.
Unter "IN:" werden notwendige Eingangsparameter und unter "OUT:" sinnvolle Ausgabeparameter genannt. Bei "Reg.:" werden die jeweils veränderten Register angegeben.

0 RESET

Kaltstart - Gleiche Wirkung wie beim Einschalten des Computers bzw. beim Drücken der RESET-Taste.

4 Adresse des Standardzeichengenerators (Bank 0)

6 Anfangsadresse des Unterprogramms INJOY

Abfrage des aktuellen Speicherhebelzustandes

```
IN:   A=0 für Joy 1     A=1 für Joy 2
OUT:  A  Bit 0...3      Richtung
         Bit 4,5        Aktionstasten von Joy 1
         Bit 6,7        Aktionstasten für Joy 2
      EI !
Reg.: AF, B
```

8 JCPSTX

Zeichenvergleich zum Syntaxtest, Stimmen die Zeichen nicht überein, erfolgt die Ausgabe "Syntax error". Andernfalls wird das nächste signifikante Zeichen gesucht (TCHAR).
Achtung! Vor Nutzung dieses Unterprogramms muß PAGE 1 in SLOT 0 umgeschaltet werden.

```
IN:   HL Zeiger auf Zeichen
      Byte hinter RST-Befehl - Vergleichsbyte
OUT:  HL Zeiger auf nächstes signifikantes Zeichen
      A  Zeichen
      Flags CY=1 Zeichen ist eine Zahl    Z=1 Anweisungsende
Reg.: AF HL
```

0CH LREB

Lesen eines Bytes von einem beliebigen (nicht aktiven) Slot.

```
IN:   A  Slotnummer
      HL Quelladresse
OUT:  A,E gelesenes Byte
Reg.: AF BC DE
```

10H JTCHAR

Test auf nächstes signifikantes Zeichen im RBASIC-Programm
Achtung! Vor Nutzung dieses Unterprogramms muß PAGE 1 in SLOT 0 umgeschaltet werden.

```
IN:   HL  Zeiger auf zuletzt interpretiertes Zeichen
OUT:  HL  Zeiger auf nächstes signifikantes Zeichen
      A   Zeichen
      F   CY=1 Ziffer   Z=1 Anweisungsende
Reg.: AF, HL
```

14H LWRB

Schreiben eines Bytes auf einen beliebigen (nicht aktiven) Slot.

```
IN:   A  Slotnummer
      HL Zieladresse
      E  Byte
Reg.: AF, BC, D
```

18H CHARO

Zeichenausgabe an File, Drucker oder Bildschirm

```
IN:   A Zeichen
      Speicherzellen ACTFCB,ACTFCB+1 (Fileadresse <> 0)
                     PTORSC (=0 -> Bildschirm, <>0 -> Drucker)
OUT:  A Zeichen
      bei Fehler Direktsprung zur Fehlerroutine
Reg.: F
```

1CH L3CALL

Rufen eines Programms auf einem beliebigen (nicht aktiven) Slot

```
IN:   High-Teil von IY Slotnummer
      IX Zieladresse
      Keine Argumente im 2. Registersatz übergeben!
OUT:  DI !
      abhängig vom gerufenen Programm
Reg.: abhängig vom gerufenen Programm
      AF', BC', DE', HL'
```

20H CPREG

Vergleich der Doppelregister HL und DE

```
IN:   HL, DE
OUT:  Flags CY=1 DE>HL
            Z=1 HL=DE
Reg.: AF
```

24H BANKUM

Aktivieren eines beliebigen Slots für die Page, die die angegebene Speicheradresse enthält.

```
IN:   A  Slotnummer
      HL Speicheradresse
OUT:  DI !
Reg.: alle
```

28H JTYPTS

Typ des Inalts des GK-Akkus feststellen
Achtung! Vor Nutzung diese Unterprogramms muß PAGE 1 in SLOT 0 umgeschaltet werden.

```
IN:   Gleitkommaakku AC1
OUT:  Flags entsprechend Typ
Reg.: F
```

2DH REZEMU

Muster (8 Bytes) eines Zeichens aus dem Speicher des Zeichengenerators lesen und in den Puffer ab Systemzelle ZEIPUF schreiben.

```
IN:   A Zeilennummer
OUT:  8 Bytes in ZEIPUF ff.
      EI !
Reg.: AF, BC, DE
```

3CH L2CALL

Rufen eines Programms auf einem beliebigen (nicht aktiven) Slot. Die Funktion ist wie bei L3CALL, aber anderer Aufruf.

```
IN:   Byte nach RST-Befehl ist die Banknummer.
      Nächste 2 Bytes sind die Zieladresse.
      Keine Argumente im 2. Registersatz übergeben!
OUT:  DI !
      abhängig vom gerufenen Programm
Reg.: abhängig vom gerufenen Programm
      AF', BC', DE', HL', IX, IY
```

33H TCHAR0

Absuchen einer BASIC-Quellzeile nach dem nächsten signifikanten Zeichen.
Achtung! Vor Nutzung dieses Unterprogramms muß PAGE 1 in SLOT 0 umgeschaltet werden.

```
IN:   HL  Zeiger in Quellzeile
OUT:  F   Z=1, wenn Ende der Quellzeile
      HL  Zeiger auf nächstes Zeichen
      A   nächstes Zeichen
Reg.: alle
```

36H Adresse der Systemzelle NACALL

38H Sprung zu einem Patch-Ruf, dann DI!, HALT

3BH MUINI

Initialisierung des Musikausgabeteils im Speicherverwaltungsschaltkreis sowie der Systemzellen für Musikausgabe (schließt MUINIT 90H ein).

```
OUT:  EI !
Reg.: A, E
```

3EH FTINI

Programmierbare Funktionstasten mit Standardwert belegen (siehe auch VFTINI)

```
Reg.: alle
```

4AH RDVRM

Ein Wort aus dem Video-RAM lesen.

```
IN:   HL  VRAM-Adresse
OUT:  A,C Lowbyte              (Textmodus: Zeichencode)
      B   Highbyte             (Textmodus: Farbnummer)
      EI !
Reg.: AF, BC
```

4DH WRTVRM

Ein Wort in den Video-RAM schreiben.

```
IN:   HL VRAM-Adresse
      A  Lowbyte        (Textmodus: Zeichencode)
      B  Highbyte       (Textmodus: Farbnummer)
OUT:  EI !
Reg.: AF, BC
```

56H OUTBCA

Ein Byte aus Register A BC-mal (verdoppelt als Wort) in den Video-RAM schreiben.

```
IN:   BC Anzahl
      A  Zeichen
      HL Video-RAM-Adresse
Reg.: F, BC, HL
```

59H INBCDE

Lesen einer Anzahl von Worten aus dem Video-RAM und Schreiben in einen Puffer (Max. 127 Worte)

```
IN:   BC Anzahl
      DE Pufferadresse
      HL VRAM-Adresse
Reg.: alle
```

5CH OTBCDE

Lesen einer Anzahl von Worten aus einem Puffer und Schreiben in den Video-RAM (max. 127 Worte)

```
IN:   BC Anzahl
      DE Pufferadresse
      HL VRAM-Adresse
Reg.: alle
```

5FH SELSCR

Auswahl des Bildschirmmodus und der Seite

```
IN:   B Modus aktiv
      C Seite aktiv
      D Modus visuell
      E Seite Visuell
OUT:  Nummern in den Systemzellen PAGEAK ff.
Reg.: alle
```

66H NMI

Nicht-Maskierbarer-Interrupt (Patch-Ruf, dann RETN)

```
Reg.: keine
```

69H SPRRES

Systemzellen zur Spriteverwaltung initialisieren

```
Reg.: AF, BC, HL
```

6CH SSCR0

Einstellen des Textmodus 40*25 Seite 0

```
Reg.: alle
```

6FH SSCR1

Einstellen des Textmodus 80*25 Seite 0

Reg.: alle

72H SSCR2

Einstellen des Grafikmodus 320*200 Seite 0 4 Farben

Reg.: alle

75H SSCR3

Einstellen des Grafikmodus 640*200 Seite 0 4 Farben

Reg.: alle

8DH TXTGSC

Ausgabe eines Zeichens 8*8 auf den aktiven Grafikschirm

IN: A Zeichen
Reg.: keine

90H MUINIT

Initialisierung der Musikausgabe

OUT: EI !
Reg.: keine

93H MUOTAE

Ausgabe eines Bytes an den Musikausgabeteil des Speicherverwaltungsschaltkreises

IN: A Registernummer des SVG
E Byte
OUT: EI !
Reg.: keine

9CH ICONST (INKEY)

Statusabfrage des CONSOL-Gerätes (Standard: Prüfen des Tastaturpuffers)

OUT: EI !
F Z=0, wenn mind. ein Zeichen im Puffer
Reg.: AF

9FH ICONIN (RDRIPU)

Eingabe vom CONSOL-Gerät (Standard: Lesen eines Zeichens aus dem Tastaturpuffer und Warten, falls er leer ist.

OUT: A Zeichen
Reg.: AF

A2H ICONOUT (OUTASC)

Ausgabe auf das CONSOL-Gerät (Standard: Ausgabe eines Zeichens auf den aktiven Textbildschirm, Stellen des Kursors auf die nächste Position).

IN: A Zeichen
Reg.: keine

A5H PRTREI

Ausgabe eines Zeichens auf den Drucker (ohne Auswertung DRUTYP)

```
IN:   A Zeichen
OUT:  F CY=0, Ausgabe i.O.
        CY=1, Abbruch mit CTRL-STOP
      EI !
Reg.: F
```

A8H ILISTST (PRTSTA)

Statusabfrage für LIST-Gerät (Standard: Test des Druckerstatus)

```
OUT:  CY=0, Z=1 nicht bereit
      CY=0, Z=0 bereit
      CY=1, Z=1 Abbruch mit CTRL-STOP (nicht bereit)
Reg.: AF
```

ABH CHRTST

Wandlung der 2-Byte-Grafikzeichen in das entsprechende Zeichen
Erkennung von ESC und Steuerzeichen

```
IN:   A Zeichen
OUT:  A gewandeltes Zeichen
      F CY=0, Z=1 A war 1 (Grafikkennbyte)
        CY=1, Z=0 A war 2 Grafikbyte A=A-40H
        CY=0, Z=0 sonst
Reg.: AF
```

AEH EDI1

Eingabe einer Zeile von der Tastatur auf den Bildschirm und in den EDI-Puffer, CTRL-U löscht die Zeilennummer im AUTO-Modus nicht.

```
OUT:  HL Anfangsadresse des EDI-Puffers - 1
Reg.: alle
```

B1H EDI2

wie EDI1, aber CTRL-U löscht im AUTO-Modus die Zeilennummer

```
OUT:  HL Anfangsadresse des EDI-Puffers - 1
Reg.: alle
```

B4H GETST

Ausgabe eines Fragezeichens und eines Leerzeichens, danach weiter mit EDI2

```
OUT:  HL Anfangsadresse des EDI-Puffers - 1
Reg.: alle
```

B7H DSTPTS

Prüfen auf gedrückte CTRL-STOP-Taste;
unter DI prüfen auf CTRL-STOP

```
OUT:  F CY=1 nach CTRL-STOP
Reg.: AF
```

BAH STPTST

Prüfen der STOP-Taste, wenn (STOPFL)=0
Nach Drücken der STOP-Taste wird in einer Schleife bis zum erneuten Drücken der STOP-Taste gewartet. Nach Drücken von CTRL-STOP wird, falls der STOP-Interrupt erlaubt ist, der Interrupt angemeldet oder ansonsten das laufende

Programm abgebrochen. Werden weder STOP noch CTRL-STOP gegeben, hat die Routine keine Wirkung.

OUT: Z=0, wenn (STOPFL)=0
sonst EI !

BDH BREAK

wie STPTST; von BASIC genutzt

OUT: wie STPTST, nach Drücken, nach Drücken der STOP-Taste HL=0
Reg.: AF, ggf. HL

C0H BEEP

Erzeugen eines Pieptons

OUT: EI !
Reg.: AF, BC, E

C3H ZCLS

Löschen des aktiven Bildschirms, wenn Z=1

IN: F
Reg.: AF, BC, DE

C6H CRSPOS

Setzen des Kursors im Textscreen

IN: H=Spalte, L=Zeile
Reg.: AF

C9H BKEYAN

F-Tastenbelegung anzeigen, falls erlaubt

IN: FTANZ
Reg.: alle

CCH KEYOFF

F-Tastenanzeige aus

OUT: FTANZ=0
Reg.: alle

CFH KEYON

F-Tastenanzeige ein

OUT: FTANZ=FF
Reg.: alle

D2H LATXTM

Einschalten des als letztes verwendeten Textmodus

Reg.: alle

D5H GSTICK

Lesen des Status von Kursortasten bzw. Joystick

```
IN:   A =0 Kursortasten
        =1 Joystick 1
        =2 Joystick 2
OUT:  A Richtungswert
      EI !
Reg.: alle
```

D8H GSTRIG

Lesen des Status der SPACE- bzw. Feuertasten

```
IN:   A=0 Tastatur
       =1,2,3,4 Abzufragende Feuertaste
OUT:  A=0 keine Taste gedrückt
       =FF Taste gedrückt
      EI !
Reg.: AF, BC
```

E1H INVT

Motor des Kassettengerätes einschalten, Warten auf den Vorton

```
OUT:  DI !
      Flag CY=1; falls CTRL-STOP gedrückt
Reg.: alle
```

E4H INBYTE

Lesen eines Bytes vom Magnetband

```
OUT:  A gelesenes Byte
      Flag CY=1, falls CTRL-STOP gedrückt
Reg.: alle
```

E7H MOFF

Motor des Kassettengerätes ausschalten

```
OUT:  EI !
Reg.: keine
```

EAH OUTVT

Motor des Kassettengerätes einschalten, Ausgabe eines Vortons

```
IN:   A=0 kurzer Vorton
      A<>0 langer Vorton
OUT:  DI !
Reg.: AF,BC,HL
```

EDH OUTBYT

Schreiben eines Bytes auf Band

```
IN:   A zu schreibendes Byte
Reg.: AF, BC, HL
```

F0H WMOFF

Warten und Motor des Kassettengerätes ausschalten

```
OUT:  EI !
Reg.: keine
```

F3H MOTST

Motor des Kassettengerätes aus-, ein- bzw. -umschalten

IN: A=0 aus, =1 ein, =FF umschalten
Reg.: AF

FCH APRGT1

Grafikkursor 1 Pixel rechts

IN: POIADR, POIMA
OUT: POIADR, POIMA (neu)
Reg.: AF

FFH APLFT1

Grafikkursor 1 Pixel links

IN: POIADR, POIMA
OUT: POIADR, POIMA (neu)
Reg.: AF

102H APUP10

Grafikkursor 1 Pixel hoch

IN: POIADR, POIMA
OUT: POIADR, POIMA (neu)
Reg.: AF

105H APUP

Grafikkursor 1 Pixel mit Test auf oberen Rand

IN: POIADR, POIMA
OUT: POIADR, POIMA (neu)
Flag CY=1, falls oberste Bildschirmzeile erreicht
Reg.: AF

108H APDWN1

Grafikkursor 1 Pixel runter

IN: POIADR, POIMA
OUT: POIADR, POIMA (neu)
Reg.: AF

10BH APDWN

Grafikkursor 1 Pixel runter mit Test auf unteren Rand

IN: POIADR, POIMA
OUT: POIADR, POIMA (neu)
Flag CY=1, falls unterste Bildschirmzeile erreicht
Reg.: AF

10EH TPPOS

Koordinatentest, Begrenzung auf mögliche Koordinaten

IN: BC x-Koordinate
DE Y-Koordinate
OUT: BC, DE begrenzte Koordinaten
Flags CY=0, falls Koordinaten außerhalb lagen
Reg.: AF, BC, DE

111H MKPMA

Bestimmen der physischen Koordinaten

IN: BC x-Koordinate
DE y-Koordinate
OUT: POIADR, POIMA
Reg.: alle

114H LDPMA

Punktadresse und Punktmaske aus dem Speicher in Register laden

IN: POIADR, POIMA
OUT: HL Punktadresse
A Punktmaske
Reg.: AF, HL

117H STPMA

Punktadresse und Punktmaske aus den Registern in die Systemzellen POIADR und POIMA

IN: HL Punktadresse
A Punktmaske
OUT: POIADR, POIMA
Reg.: keine

11AH STAINK

Aktuelle Farbe in ACTINK und PALCOL, PALCOL+1 eintragen

IN: A Farbnummer
Reg.: AF

11DH GPOINT

Lesen der Farbe des mit POIADR, POIMA bestimmten Pixels

IN: POIADR, POIMA
OUT: A Farbnummer
Reg.: AF

120H SPOINT

Setzen der Farbe des mit POIADR, POIMA bestimmten Pixels.

IN: POIADR, POIMA, PALCOL, PALCOL+1
Reg.: AF

123H FILL

Setzen einer Anzahl von Pixeln rechts vom Grafikkursor

IN: HL Anzahl
POIADR, POIMA, PALCOL, PALCOL+1
Reg.: alle

126H DELLI

Laden der Standardellipsenparameter

OUT: DE 256/elli
HL elli/256
Reg.: DE, HL

129H PAIU1

Übernahme der Randfarbe für PAINT in die Systemzellen PARAND und PARAND+1

IN: A Randfarbe
OUT: PARAND, PARAND+1
Reg.: AF

12CH PAIU2

Setzen der Pixel rechts vom Grafikkursor bis max. Anzahl gesetzt
bis Randfarbe erreicht
bis Bildrand erreicht

IN: DE Anzahl der zu testenden Pixel
Reg.: alle

12FH PAIU3

Setzen der Pixel links vom Grafikkursor bis max. Anzahl gesetzt
bis Randfarbe erreicht
bis Bildrand erreicht

IN: DE Anzahl der zu testenden Pixel
Reg.: alle

132H CAPLED

CAPS-LED ein-, ausschalten

IN: A=0 CAPS-LOCK ein
<>0 CAPS-LOCK aus
Reg.: AF

138H DBAABF

Lesen des Slotkontrollbytes

OUT: A aktuelles Slotkontrollbyte
Reg.: A

13BH DBAUM

Schreiben des Slotkontrollbytes (Speicherumschaltung)

IN: A gewünschtes Slotkontrollbyte
Reg.: keine

141H DTMABF

Direkte Tastenmatrixabfrage

IN: A Reihennummer
OUT: A Wert, Bit x=0 Taste in Spalte x gedrückt.
EI !
Reg.: AF, C

14AH TAFCB

Test, ob I/O-Operation mit einem File (über FCB) erfolgt

OUT: A=0 I/O fertig
<>0 aktueller FCB liegt vor
Flag Z=C I/O fertig
<>0 aktueller FCB liegt vor
Reg.: AF

14DH ILIST (PRTOUT)

Ausgabe auf LIST-Gerät (Standard: Zeichenausgabe auf angeschlossenen Drucker mit Auswertung der Systemzelle DRUTYP)

IN: A auszugebendes Zeichen
OUT: A Zeichen (unverändert)
bei Fehler Direktsprung zur Fehlerroutine
Reg.: F

156H LORIPU

Löschen des Tastaturpuffers

Reg.: AF, HL

159H L1CALL

Aufruf einer BASIC-Interpretererweiterung

IN: IX Adresse
BASUB=Slotnummer (von RBASIC)
keine Argumente im 2. Registersatz übergeben!
OUT: DI !
abhängig vom gerufenen Programm
Reg.: abhängig vom gerufenen Programm
AF', BC', DE', HL'

15CH VFTINI

Bildspeicherbereich für Anzeige der F-Tastenbelegung mit der aktuellen Belegung (ab FKEY1) laden

Reg.: alle

15FH RDMOD

Grafik-Display-Kontroller für Lesen eines Wortes aus dem Bildspeicher vorbereiten
- Warten bis FIFO leer
- Lesekommando ausgeben
- Warten, bis 1. gelesenes Byte bereit

Reg.: AF

162H WRMOD

Grafik-Display-Kontroller für Lesen eines Wortes in den Bildspeicher vorbereiten
- Warten bis mind. 1 Byte im FIFO frei
- Schreibkommando ausgeben
- Warten, bis mind. 1 Byte im FIFO für Schreiben frei

Reg.: AF

165H SETBOR

Randfarbe im Video-Interface-Schaltkreis einstellen und in Systemzelle VIREG2 merken

IN: A Farbnummer
Reg.: AF

168 BSINIT

Initialisierung der Bildschirmanzeige
- Standardzeichengenerator laden
- Video-Interface-Schaltkreis programmieren
- Grafik-Display-Kontroller programmieren
- Systemzellen für Bildausgabe initialisieren

(läuft unter DI !)

OUT: EI !
Reg.: alle

16BH TGRASC

Test, ob aktiver Bildschirmmodus Text- oder Grafikmodus ist

IN: Systemzelle SCRNRA
OUT: A aktiver Modus
Flag CY=0 Grafikmodus 2,3,5
CY=1 Textmodus 0,1,8,9
Reg.: AF

16EH SETPAL

Palettenfarben entsprechend Palettennummern im Video-Interface-Schaltkreis einstellen und in Systemzellen VIREG4...6 merken (gerade Nummer: Palette 0, ungerade Nummer: Palette 1)

IN: A Palettennummer
Reg.: AF

171H SETPAP

Hintergrundfarbe im Video-Interface-Schaltkreis einstellen und in der Systemzelle VIREG3 merken

IN: A Farbnummer
Reg.: AF

174H KOMSCR

Bildschirmmodus zur Eingabe eines Kommandos einstellen (Seite 0 des letzten verwendeten Textmodus)

Reg.: alle

177H WRZEMU

Muster eines Zeichens (8 Bytes) in den Speicher des Zeichengenerators einschreiben

IN: C Zeichennummer
HL Zeiger auf 1. Byte im Quellpuffer
OUT: EI !
Reg.: AF, BC, HL

17AH PUTSPR

Sprite (8*8 Pixel) in Bildspeicher schreiben
- falls Sprite bereits gesetzt, dann alten Hintergrund regenerieren
- Test, ob Kollision mit anderem bereits gesetztem Sprite vorliegt (ja -> Kollisionsanmeldung)
- Hintergrund retten
- Sprite schreiben
- UP läuft aus Zeitgründen unter DI !

IN: A Spritenummer

BC x-Koordinate
DE y-Koordinate
OUT: Flag CY=1, falls Sprite nicht definiert
EI !
Reg.: alle

17DH MONTV

Automatisches Einstellen des Monitormodus (Bild breit), wenn DSE angeschlossen, sonst Einstellen des TV-Modus (Bild schmal)
für SCP SCREEN 9 Seite 0
für RBASIC SCREEN 8 Seite 0

IN: A Kennbyte A=0 --> SCREEN 9
A=FF --> SCREEN 8
Reg.: alle

180H GETA35

Test, ob Zeichen im Ringpuffer der Tastatureingabe

OUT: Flag Z=0 Zeichen im Puffer
Reg.: AF, HL

183H RIPUZI

Zeiger in Tastatur-Ringpuffer weiterstellen

IN: HL Zeiger
OUT: HL Zeiger auf nächste Pufferposition
Reg.: AF, HL

186 CAPSB

Reaktion auf Drücken der CAPS-Taste
- Inhalt der Systemzelle CAPSSW negieren
- CAPS-LED umschalten

Reg.: AF, HL

189H Adresse der Systemzelle ZLLAST
18BH Adresse der Systemzelle FKEY1
18DH Adresse der Systemzelle SCHRIP
18FH Adresse der Systemzelle STCSSW
191H Adresse der Systemzelle CAPSSW
193H Anfangsadresse der Interrupttabelle

195H WAITBC

Warteschleife; Länge entsprechend Inhalt von BC (BC + 20µs)
Rücksprung, wenn BC=0

IN: BC
Reg.: AF, BC

198H REMASK

Maskenregister des Grafik-Display-Kontrollers auf FFFFH setzen (z.B. notwendig vor der Textausgabe nach Grafikausgabe)

Reg.: AF, BC, HL

ANHANG B

Beispiel für die Nutzung von Betriebssystemteilen unter SCPX

Nach der Eingabe des CCP-Kommandos "SLOTDUMP page slot" werden die ersten 256 Bytes des mit page und slot spezifizierten Speicherbereichs auf dem Bildschirm angezeigt.

```
         TITLE   Speicheranzeige
;*******************************
;FILE  :SLOTDUMP.MAC
;AUTOR :Lutz Elssner
;STAND :29.08.89
;-------------------------------
               .Z80
;==============================
         LD      DE,UTX
         LD      C,9
         CALL    5             ;Ausgabe: Text ab Marke UTX
;==============================
;Unterprogramm Speicheranzeige
;IN: (5DH)=Page, (6DH)=Slot    Kommandopuffer des CCP
;------------------------------
DUMP:    LD      A,(5DH)       ;1. Parameter Page ......xx
         SLA     A
         SLA     A
         SLA     A
         SLA     A
         SLA     A
         SLA     A
         LD      H,A           ;xx000000
         LD      L,0           ;Anfangsadresse
;
         LD      A,(6DH)       ;2. Parameter Slot ......xx
         AND     00000011B     ;0 1                000000xx
         LD      B,A
         SLA     B
         SLA     B
         OR      B             ;2 3                0000xxxx
         SLA     B
         SLA     B
         OR      B             ;4 5                00xxxxxx
         SLA     B
         SLA     B
         OR      B             ;6 7                xxxxxxxx
         LD      B,A           ;Slotnummer für alle Pages eintragen
;
         PUSH    HL
         PUSH    BC
         LD      DE,SLTX
         LD      C,9
         CALL    5             ;Ausgabe: Text "Slot: "
         POP     BC
         LD      A,B
         PUSH    BC
         CALL    PRNBY         ;Ausgabe: Slotkontrollbyte
         CALL    PCRLF
         POP     BC
         POP     HL
;------------------------------
         LD      E,16          ;Zähler Zeilenzahl
;Anzeigen 1 Zeile
;IN: HL Adresse
;------------------------------
DUMPZ:   PUSH    DE
```

```
         PUSH    HL
         PUSH    BC
         CALL    PCRLF        ;Ausgabe: CR/LF
         LD      E,L
         LD      D,H
         CALL    PRNAD        ;Ausgabe: Adresse
         CALL    PRNSP        ;Ausgabe: 2 Leerzeichen
         CALL    PRNSP
         POP     BC
         POP     HL
;
         LD      C,16         ;Zähler 16 Bytes pro Zeile
         PUSH    BC
         PUSH    HL
;Anzeigen HEX
DUMPB:   CALL    READB        ;Byte aus Speicher lesen
;
         PUSH    HL
         PUSH    BC
         CALL    PRNBY        ;Ausgabe: Byte (hexadezimal)
         CALL    PRNSP        ;Ausgabe: Space
         POP     BC
         POP     HL
         INC     HL           ;Adresse weiterzählen
;
         DEC     C
         JR      NZ,DUMPB     ;Zeile voll ?
         CALL    PRNSP
         CALL    PRNSP        ;Ausgabe: 2 Leerzeichen
         POP     HL           ;Adresse 16 Bytes pro Zeile
         POP     BC           ;Zähler 16 Bytes pro Zeile
;Anzeigen ASCII
DUMPA:   CALL    READB        ;Byte aus dem Speicher lesen
;
         PUSH    HL
         PUSH    BC
         AND     7FH          ;)
         CP      7FH          ;)
         JR      Z,DUMPA2     ;) für Byte 20H bis 7EH zugehöriges
         CP      ' '          ;) ASCII-Zeichen, sonst Punkt zuordnen
         JR      NC,DUMPA1    ;)
DUMPA2:  LD      A,'.'        ;)
DUMPA1:  LD      E,A          ;)
         LD      C,2
         CALL    5            ;Ausgabe: Zeichen
         POP     BC
         POP     HL
         INC     HL           ;Adresse weiterzählen
;
         DEC     C
         JR      NZ,DUMPA     ;Zeile voll ?
;
         POP DE
         JR      NZ,DUMPZ     ;alle Zeilen ausgeben ?
;
         RET                  ;ja -> fertig
;------------------------------
UTX:     DB      0DH,0AH,'SLOTDUMP 1.00   ',0DH,0AH
         DB      'Syntax: SLOTDUMP seite bank'
         DB      '    Parameter 0...3',0DH,0AH,0AH,'$'
SLTX:    DB      'SLOT: $'
;==============================
;Unterprogramm Byte lesen aus beliebigem Slot
;Übergabe im Register A
;HL=Adresse, B=Slot
RDADB:   LD      IX,(3EH)
         LD      DE,0C000H
```

```
         OR      A            ;CY=0
         PUSH    HL
         SBC     HL,DE
         POP     HL
         JR      NC,RE1       ;Adresse im Wurzelsegment
;HL = 00C0-BFFF
         LD      DE,2
         ADD     IX,DE        ;IX Adresse des Spezialprogramms ANRD
         RES     6,B          ;Page 3, Slot 2
         SET     7,B
;
JPIX:    JP      (IX)         ;IN: HL Adresse
;                                   B Slotkontrollbyte
;------------------------------
;HL = C000-FFFF
RE1:     LD      E,(IX+6)     ;ROM-Adresse ANRD ermitteln
         LD      D,(IX+7)
         LD      (ANADR),DE
;
         LD      A,B          ;Slot in B vorbereiten
         AND     11000000B    ;Bit 7,6 bleiben
         LD      B,A
;
         LD      E,(IX+26)    ;Speicherkonfiguration für
         LD      D,(IX+27)    ;Aufruf ANRD ermitteln
         LD      A,(DE)
         LD      (ANSL),A
;
         AND     00111111B    ;Bit 5,4,3,2,1,0 von SLCCTRL
         OR      B            ;übernehmen
         LD      B,A          vollst. Speicherkonfiguration in B
         CALL    JPIX         ;Aufruf ANRD
;                             ;IN: HL Adresse
;                                   B Slotkontrollbyte
ANADR:   DEFW    0
ANSL:    DEFB    0
;
         RET
;============================
;Unterprogramm Hexadezimalanzeige
;IN: DE Integerzahl
;OUT: DE Integerzahl bleibt erhalten
PRNAD:   LD      A,0          ;Highbyte
         CALL    PRNBY
         LD      A,E          ;Lowbyte
;                              Byte ausgeben
PRNBY:   PUSH    AF
         RLCA
         RLCA
         RLCA
         RLCA                 ;1. ASCII-Zeichen
         CALL    NUMP1
         POP     AF           ;2. ASCII-Zeichen
;Zuordnung Halbbyte zu Zeichen 0...9,A,B,C,D,E,F
NUMP1:   AND     0FH
         ADD     A,'0'
         CP      3AH
         JR      C,NUMP2      ;0...9
         ADD     A,7          ;A...F
NUMP2:   PUSH    DE
         LD      E,A
         LD      C,2
         CALL    5            ;Ausgabe: Zeichen
         POP DE
         RET
;Unterprogramm Ausgabe Leerzeichen
PRNSP:   PUSH    BC
```

```
         PUSH    HL
         LD      E,' '
         LD      C,2
         CALL    5
         POP     HL
         POP     BC
         RET
;=============================
;Unterprogramm Ausgabe Zeichen CR/LF
PCRLF:   PUSH    BC
         PUSH    HL
         LD      E,0DH        ;CR
         LD      C,2
         CALL    5
         LD      E,0AH        ;LF
         LD      C,2
         CALL    5
         POP     HL
         POP     BC
         RET
;=============================
         END
```