PSI ψ 80

KONTRON
MIKROCOMPUTER GMBH

Technische Beschreibung

Diese technische Beschreibung ist mit größter Sorgfalt erstellt worden. Es wird jedoch keine Gewähr für die Freiheit von Fehlern und Irrtümern gegeben. Technische Änderungen bleiben vorbehalten.

Für alle Anfragen stehen Ihnen unsere Technischen Büros bzw. Ihre lokale Verkaufsorganisation zur Verfügung.

PSI Ψ 80

KONTRON MIKROCOMPUTER GMBH

PSI Ψ 80-Computersysteme

auf Basis Z80A-CPU

Systemkommandos
Dienstprogramme
Fehlersuchpaket
Betriebssystem KOS
Hardware
Cross-Reference

Technische
Beschreibung

PSI Ψ 80
TECHNISCHE BESCHREIBUNG

Version: 5.3

1. August 1981

Diese Technische Beschreibung enthält detaillierte Hard- und Software-Informationen über die Kompaktcomputersysteme der PSI80-Reihe. Sie stellt eine Arbeitsgrundlage für den erfahrenen Computer-Anwender dar, der die Leistungsfähigkeit und Flexibilität dieses Computersystems auf System- und Prozessor-Ebene voll nutzen möchte.
Für den Erstbenutzer des PSI80-Systems ist - unabhängig von seiner generellen Erfahrung auf Computersystemen - die sorgfältige Beachtung des PSI80-Bedienungshandbuchs empfohlen.

Diese Technische Beschreibung ist mit größter Sorgfalt erstellt worden. Es wird jedoch keine Gewähr für die Freiheit von Fehlern und Irrtümern gegeben. Technische Änderungen bleiben vorbehalten.

Für alle Anfragen stehen Ihnen unsere Technischen Büros bzw. Ihre lokale Verkaufsorganisation zur Verfügung.

Inhaltsverzeichnis der Technischen Beschreibung:

Inhaltsverzeichnis des Bedienungshandbuchs:

Hier nicht vorhanden da eigenständige Dokumentation

Systemkommandos zum Betriebssystem KOS

Version: 4.3/5.3

1. August 1981

Dieser Teil des Handbuches beschreibt sämtliche Dienstleistungsprogramme des Betriebssystems KOSA/5.
U.a. beinhaltet es Kommandos zur Handhabung von Dateien (COPY, DELETE, DUMP, MOVE, PRINT, RENAME), zur Einrichtung von Ein-/Ausgabetreibern (INISER, EAK), zur Systemsteuerung (TASK, SPOOL) und zur Systemüberwachung (STATUS, STOP). Außerdem sind hier die Programme zur Disketten- und Medienverwaltung (COPYD2, FORMAT, IL) beschrieben.
Die Programmsysteme zur Erstellung von Programmen (ASSEMBLER, LINKER, EDITOR, KDM) und die Hilfsprogramme (Utilities) sind in eigenen Kapiteln beschrieben.

Inhaltsverzeichnis

1. Übersicht

Eine Reihe von Kommandos steht mit dem Betriebssystem KOS zur Erstellung und Verwaltung von Programmen und Texten, zur Systemsteuerung und zur Systemverwaltung zur Verfügung. Diese rufen entweder KOS-interne (residente) oder externe (auf beliebigen Medien residente) Programme auf. Folgende Kommandos sind implementiert:

a) KOS-interne Kommandos

- A Belegung von Speicherbereichen
- C Umschaltung Groß-/Kleinschreibung
- D Freigabe von Speicherbereichen
- F Schließen aller geöffneten Dateien
- I Ausdruck des Inhaltsverzeichnis eines Mediums
- M Ausdruck und/oder Definition des Mastermediums
- N Neuinitialisierung von Medien- oder E/A-Treibern
- P Vor- und Rückwärtsblättern der Anzeige
- R Definition oder Ausführung von Kommandomacros
- S Speicherinhalte auf Medien abspeichern
- X Wiederholung eines Programms (ohne Neuladung)

b) KOS-externe Kommandos

- COPY Kopieren beliebiger Datenquellen auf beliebige Datenziele
- COPY1 Kopieren einzelner Dateien bei Einlaufwerksystemen
- COPYD2 Kopieren einer Diskette bei Mehrlaufwerksystemen
- CPFILE Byteweiser Vergleich von zwei Dateien
- DATE Datumseintrag ins Betriebssystem
- DEFP Definition der Datei-Properties
- DEL Löschen von Dateien oder Dateigruppen
- DO Ausführung einer Kommandodatei
- DUMP Hexadezimaler Ausdruck einer beliebigen Datei
- EAK Verwaltung von E/A-Kanälen
- FORMAT Formatieren einer Diskette
- IL Ausdruck des Inhaltsverzeichnis einer Diskette
- INFO Bildschirm-orientierter Ausdruck einer ASCII-Datei
- INISER Initialisierung der PSI80-Serienschnittstellen
- MAP Ausdruck der momentanen Speicherbelegung
- MOVE Kopieren von Dateien oder Dateigruppen
- PRINT Ausdruck einer ASCII-Datei
- REN Umbenennung von Dateien
- RLOAD Relokatives Laden einer OBJ-Datei
- SPOOL Ausgabe einer ASCII-Datei auf E/A-Treiber
- STATUS Ausdruck der Belegung aller aktiven Medien
- STOP Abbruch von Kommandozeilen oder -dateien
- TASK Task-Verarbeitung

Folgende Programme sind in separaten Kapiteln beschrieben: PSI/BASIC, ASM, LINK, CROSS, EDIT, KDM und Utilities.

1.1 Der Start des Betriebssystems

Nach dem Laden des Betriebssystems von einem Medium führt KOS automatisch das Kommando

DO KOS.INI

aus. Das Programm 'DO' lädt die Kommandodatei KOS.INI und sorgt anschließend über einen entsprechenden Systemaufruf für die Abarbeitung der in der Datei 'KOS.INI' enthaltenen Kommandos. Die Datei 'KOS.INI' enthält standardmäßig das Kommando

RLOAD SELECT

Das Kommando 'SELECT' in der Kommandodatei KOS.INI bietet eine elegante Möglichkeit, nach dem Kaltstart des Betriebssystems über ein Menu in verschiedene Programme zu verzweigen (Siehe SELECT-Kommando).

Der Inhalt der Datei 'KOS.INI' kann durch den EDITOR verändert werden. Ist beispielsweise erwünscht, daß das PSI80-System nach jedem Laden von KOS automatisch BASIC aufruft, so muß die Datei 'KOS.INI' auch (oder nur) das Kommando 'BASIC' enthalten.

Als Hinweis, daß KOS bereit ist, Eingaben zu akzeptieren, erscheint die Meldung

KOS:

am Zeilenanfang. Bis zu 256 Zeichen können daraufhin eingegeben werden. Eine Kommandoeingabe ist grundsätzlich mit der Taste 'RETURN' abzuschließen.

Eingabefehler können jederzeit vor Abschluß einer Kommandozeile korrigiert werden. Hierzu stehen zwei Tasten zur Verfügung:

- RUBOUT (ASCII-Code: 7FH)
  löscht die gesamte bisher eingegebene Zeile
- CURSOR LEFT löscht das zuletzt eingegebene Zeichen

  Dies ist auch mit der Tastenkombination
  CNTRL-H (ASCII-Code: 08H-Backspace) möglich.

Hinweis: Tabulator und nicht druckende Steuerzeichen werden durch CNTRL-H ebenfalls als ein Zeichen behandelt. Die Anzeige am Bildschirm kann dadurch vom logischen Inhalt der Kommandozeile abweichen.

1.2 Kommandoeingabe

KOS unterscheidet zwischen internen und externen Kommandos. Externe Anwender- oder Systemprogramme sind als Dateien mit ablauffähigem Maschinencode auf beliebigen Medien (Disketten, Platte etc.) gespeichert. Der Typ solcher Dateien ist grundsätzlich 'COM'. Dieser Typ darf beim Aufruf eines Kommandos nicht mit angegeben werden.

Allgemeines Eingabeformat:

```
mn:kommando parameter1...parameterN<---
mn:kommando,<---
```

Zum Abschluß einer Kommandoeingabe muß die Taste 'RETURN' (im folgenden mit '<---' symbolisiert) gedrückt werden. Folgt dem Kommando unmittelbar ein Komma mit darauffolgendem RETURN, so wird das Programm nur geladen und nicht ausgeführt. Die Ausführung kann mit dem Kommando 'X' wiederholt ausgelöst werden, solange das von extern geladene Programm nicht überschrieben oder verändert wird.

Die Mediennummer 'mn' ist optional. KOS sucht die entsprechende Datei zunächst auf dem Mastermedium und, falls dort nicht vorhanden, anschließend auf allen übrigen aktiven Medien.

Folgen von Kommandos können in einer bis zu 256 Stellen langen **Kommandozeile** eingegeben werden. Sie werden vom Kommandointerpreter in der Reihenfolge der Eingabe nacheinander abgearbeitet. Bei Eingabe von 'RETURN', spätestens jedoch nach 256 Zeichen, beginnt die Verarbeitung. Trennzeichen zwischen den einzelnen Kommandos ist ein Strichpunkt (;):

```
kommandofeld1;kommandofeld2; ... ;kommandofeldN<---
```

Tastatureingaben zur Steuerung des Kommandoablaufs können im Kommandoaufruf eingeschlossen sein. Sie sind durch Anführungszeichen (") abgeschlossen. RETURN wird durch "<" symbolisiert:

```
BASIC "LOAD DEMO<RUN"
```

1.3 Ausgabesteuerung

Alle Systemprogramme, die Ausgaben auf den Sichtschirm des PSI80-Computers veranlassen, reagieren auf Tastatureingaben wie folgt:

ESCAPE:	Programmabbruch
CNTRL-S:	Geschwindigkeitsumschaltung (= 'Speed')
CNTRL-P:	Ausgabeunterbrechung mit der Möglichkeit, mit den Cursortasten auf den zurückliegenden Text (bis zu 8 Seiten) zurückzublättern, bis eine beliebige andere Taste gedrückt wird (= 'Paging'). Es gilt: CURSOR DOWN :eine Zeile vorwärts CURSOR UP :eine Zeile rückwärts CURSOR RIGHT:eine Seite vorwärts CURSOR LEFT :eine Seite rückwärts
übrige Tasten:	Programmunterbrechung, bis wieder eine beliebige Taste gedrückt wird.

1.4 Syntaxausdruck (Hilfe-Funktion)

Viele KOS-Dienstprogramme geben bei Aufruf ohne Parameter Hinweise über die erforderliche Syntax des korrekten Kommandoaufrufs aus. Derartige Ausgaben erfolgen in Klartext und enthalten mindestens zwei bis drei Textzeilen. Bei komplexeren Kommandos (z.B. COPY) werden detaillierte Hinweise und Beispiele ausgegeben.

1.5 Dateinamen

Ein Kommandofeld enthält meist eine oder mehrere Dateiadressen als Parameter. Ihr Aufbau wird im folgenden erläutert.

Dateiadressen sind symbolische Namen für Informationen auf Floppy Disks oder anderen Speichermedien. Sie geben die Zuordnung zwischen symbolischer und physikalischer Adresse (Name <---> Spur/Sektor) durch die Dateiverwaltung. Diese Zuordnung kann eindeutig, oder, zur Ansprache von Dateigruppen, auch mehrdeutig sein:

```
EDA  ---  eindeutig spezifizierte Datei
MDA  ---  mehrdeutig spezifizierte Datei
```

Dateiadressen bestehen aus zwei Teilen, dem Dateinamen und dem Dateityp. Name und Typ sind durch einen Punkt getrennt; die maximale Zeichenanzahl beträgt 8 für den Namen bzw. 3 für den Typ. Der zulässige Zeichenvorrat umfaßt die für Kommandos (A...Z, a...z, 1...9) erlaubten Zeichen. Zwischen Namen und Typ dürfen neben dem Punkt (.) keine weiteren Trennzeichen stehen.

Jeder Dateiadresse kann eine Mediennummer mn (Ziffer 0 bis 6), gefolgt von einem Doppelpunkt vorangestellt werden:

```
Mediennummer:Dateiname.Dateityp
     *              *       *
     *              *       *------ max. 3 Zeichen
     *              *---------------- max. 8 Zeichen
     *------------------------------- Ziffer 0 bis 6
```

Die Angabe einer Mediennummer ist optional, da KOS beim Eröffnen einer Datei diese auf allen aktiven Medien sucht. Die Suchsequenz beginnt auf dem Mastermedium. Das **Mastermedium** ist dasjenige Medium, auf dem bei Dateireferenzen mit fehlender Mediennummer zuerst gesucht wird. Die Umschaltung erfolgt durch das KOS-internen Kommando 'M'.

Hinweis: Aus Gründen der Aufwärtskompatibilität zur KOS-Version 3.2 können die Mediennummern 7, 8 und 9 nur von Programmen aus verwendet werden, nicht aber bei der Kommandoeingabe.

Dateien sind zusätzlich mit Dateieigenschaften gekennzeichnet. Diese werden ausführlich beim Kommando DEFP beschrieben.

Eindeutige Dateiadressen - EDA

Eindeutige Dateiadressen kennzeichnen in eindeutiger Weise den dieser (symbolischen) Dateiadresse zugeordneten physikalischen Bereich eines Mediums: eine EDA adressiert genau eine Datei auf einem bestimmten Medium.

Eindeutige Dateiadressen erfordern mindestens die Angabe eines vollständigen Dateinamens. Der Dateityp darf in manchen Fällen fehlen, da verschiedene Programme und Operationen bestimmte Dateitypen voraussetzen oder zumindest einen Vorzugstyp haben. Beispielsweise impliziert der Assembleraufruf die Existenz einer Datei vom Typ 'SRC' (Quelle = Source).

Mehrdeutige Dateiadressen - MDA

Eine mehrdeutige Dateiadresse kennzeichnet eine beliebige Anzahl von Dateien mit einer gemeinsamen Eigenschaft. Eine solche gemeinsame Eigenschaft kann beispielsweise sein:

- ein bestimmter Dateityp
- ein bestimmter Buchstabe im Dateinamen
- ein bestimmter Dateiname
- eine bestimmte Dateieigenschaft
- ein bestimmtes Medium

Zur Konstruktion von mehrdeutigen Dateiadressen stehen zwei Sonderzeichen zur Verfügung:

- das Zeichen '*' (ASCII-Code: 2AH) wirkt als Universalbezeichner. Es kann für eine beliebige Anzahl von Zeichen eines der beiden Teile einer vollständigen Dateiadresse stehen.

- Das Zeichen '?' (ASCII-Code: 3FH) repräsentiert ein beliebiges im jeweiligen Bereich zugelassenes Zeichen einer Dateiadresse.

Beispiel für mehrdeutige Dateireferenzen:

a)	*.ASM	alle Dateien vom Typ ASM
b)	ABC.*	alle Dateien des Namens ABC mit beliebigem Typ
c)	ABC*.BAS	alle Dateien, die einen mit ABC beginnenden Namen und den Typ BAS haben
d)	X*.?	alle Dateien, deren Name mit X beginnt und deren Typ aus einem Zeichen besteht

2. KOS-interne Kommandos

KOS-interne Kommandos werden beim Kaltstart des Betriebssystems mitgeladen. Sie sind Teil des Betriebssystems. Alle internen Kommandos verwenden die Kanäle E-1 bzw. A-1 für Ein-/Ausgaben. Interne Kommandos bestehen grundsätzlich aus einem Kennbuchstaben, der sowohl groß, als auch kleingeschrieben werden kann.

2.1 A - Kommando (Allocation)

Aufruf: A adr n<---

Funktion:
Belegung von 'n' Speichersegmenten von je 128 Byte ab Adresse 'adr'. Hiermit können beliebige noch freie Speicherbereiche als belegt deklariert werden (siehe MAP-Kommando). Der Versuch, bereits belegte Segmente erneut zu belegen, wird mit der Fehlermeldung 'Speicherbelegungskonflikt' beantwortet. 'adr' und 'n' sind hexadezimale Zahlen.

2.2 C - Kommando (Change Case)

Aufruf: C<---

Funktion:
Umschaltung zwischen Groß- und Kleinschreibung. Nach dem Laden des Betriebssystems ist 'Großschreibung' eingestellt. Als Hinweis auf den gerade aktiven Modus gibt das Betriebssystem die Meldung 'KOS:' klein- oder großgeschrieben aus.

2.3 D - Kommando (Deallocation)

Aufruf: D adr n<---

Funktion:
Freigabe von 'n' Speichersegmenten von je 128 Bytes ab Adresse 'adr'. Die Größen 'n' und 'adr' sind hexadezimale Zahlenwerte.

2.4 F - Kommando (File close)

Aufruf: F<---

Funktion:
Schließt alle offenen Dateien, auch eine vom SPOOL-Kommando gerade bearbeitete Datei wird geschlossen und das Drucken abgebrochen.

2.5 I - Kommando (Index)

Aufruf: I mn:dateiname.typ<---
I mn:dateigruppe<---

Funktion:
Auflistung aller im Parameterfeld des Kommandos spezifizierten Dateien, die nicht geheim (s. DEFP-Kommando) sind. Alle Parameter sind optional. Ist das Parameterfeld nicht besetzt, so werden alle nicht geheimen Dateien des Mastermediums ausgegeben.

Beispiele:

I ABC.*<---	alle nicht geheimen Dateien mit Namen 'ABC'
I ???.BAS<---	alle nicht geheimen Dateien mit Typ 'BAS'
I 3:<---	alle nicht geheimen Dateien auf Medium 3
I<---	alle nicht geheimen Dateien auf dem Mastermedium

Wird keine Datei der angegebenen Spezifikation gefunden, so antwortet KOS mit der Meldung:

Medium mn: Datei nicht vorhanden.

2.6 M - Kommando (Master medium)

Aufruf: M mn<---

Funktion:
Definition und/oder Ausgabe des Mastermediums. Ist kein Parameter spezifiziert, wird nur die Nummer des derzeitigen Mastermediums ausgedruckt. Das Mastermedium ist dasjenige Medium, auf dem bei Dateireferenzen mit fehlender Mediennummer zuerst gesucht wird.

2.7 N - Kommando (New discette)

Aufruf: N $EAT<---

Funktion:
Neuinitialisierung der Dateiverwaltung oder eines E/A-Treibers. Die Funktion wird automatisch beim Wechsel von Disketten durchgeführt, sofern die Diskettennamen sich voneinander unterscheiden. **Wenn zwei Disketten den gleichen Namen haben, dann muß das N-Kommando beim Wechsel gegeben werden.**

Das Kommando 'N $EAT' führt zur Neuinitialisierung eines E/A-Treibers. Es wird die Routine 'INIT' des E/A-Treibers ausgeführt.

Beispiele:

N $EATN<---	Initialisierung des E/A-Treibers $EATN
N 2<---	Initialisierung des Mediums 2
N<---	Initialisierung aller aktiven Medien

2.8 P - Kommando (Page Mode)

Aufruf: P<---

Funktion:
Blättern in den bisherigen Sichtschirmausgaben. Der Bildwiederholspeicher des PSI80-Systems speichert ständig die letzten 8 Textseiten. Nach dem P-Kommando kann mit den Cursortasten ein beliebiger Ausschnitt der letzten 8 Textseiten auf den Sichtschirm gebracht werden.

Durch dieses Kommando können z.B. Bedienfehler oder verpaßte Ausgaben zurückgeholt werden.

Tasteninterpretation:
CURSOR UP :eine Zeile rückwärts schalten
CURSOR DOWN : eine Zeile vorwärts schalten
CURSOR RIGHT:eine Seite vorwärts schalten
CURSOR LEFT :eine Seite rückwärts schalten

Alle anderen Tasten bringen den ursprünglich vorhandenen Bildausschnitt wieder zurück und schließen das P-Kommando ab.

2.9 R - Kommando (Repeat)

Aufruf: R kommando1;kommando2;...kommandoN<---

Funktion:
Definition eines Kommandomacros. Die im Parameterfeld des R-Kommandos aufgeführte Kommandofolge wird zwischengespeichert und kann anschließend beliebig oft durch ein R-Kommando ohne Parameterangabe wiederholt werden. Ein neues Kommandomacro (R mit Parameterangabe) überschreibt ein eventuell bereits vorhandenes Kommandomacro.

Beispiel:
R EDIT TEST.SRC;ASM=TEST/L;LINK TEST/N,TEST/P:100/E<---

Die Ausführung der Kommandofolge EDIT-ASM-LINK kann nun beliebig oft durch das Kommando R<--- wiederholt werden.

2.10 S - Kommando (Save)

Aufruf: S n mn:dateiname.typ adr<---

Funktion:
Abspeichern (Retten) von n x 128 Bytes aus dem Anwenderspeicherbereich in eine Datei. 'n' und 'adr' sind Hexadezimalzahlen. Der Abspeichervorgang beginnt bei Adresse 'adr'. Fehlt dieser Parameter, so wird dafür der Wert 100H eingesetzt.

Beispiele:

S A XDATEI.OBJ<---
Abspeichern des Bereichs 100H bis 5FFH unter dem Namen XDATEI.OBJ.

S 28 ABC.DEF<---
Abspeichern des Bereichs 100H bis 14FFH unter dem Namen ABC.DEF.

S 10 PRT.EAT D000<---
Abspeichern des Bereichs D000 bis D7FF unter dem Namen PRT.EAT.

Bei allen aufgeführten Beispielen wird das Mastermedium adressiert. Der Wert von 'adr' wird mit im Inhaltsverzeichnis abgelegt.

2.11 X - Kommando (Execute)

Aufruf: X p1 p2 ... pN<---

Funktion:
Wiederholung eines zuvor geladenen oder ausgeführten Programms ohne erneutes Laden. Die Angabe der Parameter p1...pN ist optional; ihre Art und Anzahl ist vom jeweils wieder zu startenden Programm abhängig.

Beispiel:

PRINT,<---
X KOS.INF<---
X KOS.INI<---
Laden des Programms PRINT (das Komma verhindert die sofortige Ausführung - nun ist beispielsweise ein Diskettenwechsel möglich). Anschließend werden die Dateien 'KOS.INF' und 'KOS.INI' ausgedruckt, ohne daß das Dienstprogramm 'PRINT' neuerlich geladen werden muß.

3. Diskresidente KOS-Kommandos

3.1 COPY - Kommando (allgemeiner Datentransfer)

```
Aufruf:       COPY quelle ziel<---
Voreinst.:    mn (quelle)  =  0
                           =  Mastermedium, falls ziel = $EAZ
              typ (quelle) =  COM
              mn (ziel)    =  1 falls mn (quelle) = 0 und
                           =  0 falls mn (quelle) = 1
                           =  Mastermedium, falls quelle = $EAQ
              name (ziel)  =  name (quelle)
              typ (ziel)   =  typ (quelle)

Dateispez.:   EDA
```

Funktion:
Das Programm COPY dient zum Transfer von Daten beliebiger Quellen an beliebige Ziele. Als Datenquelle und/oder Datenziel sind möglich:

- eindeutig spezifizierte Dateien auf beliebigen Medien
- E/A-Treiber (gekennzeichnet durch ein $-Zeichen)

Dies ergibt vier verschiedene Transfermöglichkeiten:

- Kopieren einer Datei auf eine andere Datei
- Kopieren einer Datei an einen E/A-Treiber
- Kopieren von einem E/A-Treiber auf eine Datei
- Kopieren von einem E/A-Treiber zu einem anderen E/A-Treiber

Die allgemeinen Kommandoformate lauten:

```
COPY mn:quelldatei.typ mn:zieldatei.typ<---
COPY mn:quelldatei.typ $EAZ<---
COPY $EAQ mn:zieldatei.typ <---
COPY $EAQ $EAZ<---
```

Ist die Zieldatei auf dem Zielmedium bereits vorhanden, so antwortet COPY mit der Rückmeldung:

```
----> COPY: Zieldatei bereits vorhanden - Datei überschreiben? (J)
```

Bei Eingabe von 'J' und 'Y' wird die vorhanden Datei überschrieben.

E/A-Kanäle:

```
Eingabe              --->  Kanal E-1
Rückmeldungen        --->  Kanal A-1
Datenquelle ($EAQ)   --->  Kanal E-5
Datenziel ($EAZ)     --->  Kanal A-5
```

Beispiele:

COPY 0:PSI1 1:PSI2<---
Kopiert die Datei PSI1.COM von Medium 0 auf Medium 1, wo sie unter dem Namen PSI2.COM abgelegt wird. Die Mediennummern dieses Beispiels sind voreingestellt und wären deshalb hier nicht notwendig.

COPY PSI1.COM<---
Kopiert die Datei PSI1.COM von Medium 0 auf Medium 1.

COPY TEST.PRN $SIOA<---
Kopiert die Datei TEST.PRN auf den E/A-Treiber $SIOA.

COPY $KEY $SIOA<---
Kopiert vom E/A-Treiber $KEY (Tastatur) auf den E/A-Treiber $SIOA . Dieser Vorgang endet mit der Eingabe von CNTRL-D (End of transmission).

COPY $PSIA TEST.X<---
Kopiert vom E/A-Treiber $PSIA auf die Datei 'TEST.X'.

Anmerkung:
Es ist Aufgabe des E/A-Treibers, dem COPY-Kommando das Übertragungsende mitzuteilen (siehe "PSIA" in der Utility-Beschreibung).

3.2 COPY1 - Kommando (Kopieren einer Datei bei 1 Laufwerksystemen)

Aufruf:	COPY1 quelldatei zieldatei<---		
Voreinst.:	mn	=	Mastermedium
	typ (quelle)	=	COM
	name (ziel)	=	name (quelle)
	typ (ziel)	=	typ (quelle)
Dateispez.:	EDA		

Funktion:
Kopieren einzelner Dateien bei Ein-Mediensystemen. Das Programm arbeitet mit Benutzerführung und verlangt zunächst das Einlegen der Quelldiskette. Die Quelldatei wird in den Arbeitsspeicher eingelesen. Danach verlangt COPY1 das Einlegen der Diskette, auf der die Kopie erzeugt werden soll. Der Wechsel zwischen Original und Kopie kann bei kleinem Speicher und großen Dateien mehrmals wiederholt werden müssen.

E/A-Kanäle:

Eingabe	--->	Kanal E-1
Rückmeldungen	--->	Kanal A-1

Beispiele:

COPY1 TEST.SRC<---
COPY1 TEST<---
Kopiert die Datei TEST.SRC auf eine Datei gleichen Namens, bzw. kopiert die Datei TEST.COM.

3.3 COPYD2 - Kommando (Diskettenkopieren in 2 Laufwerksystemen)

```
Aufruf:        COPYD2 quellaufwerk>ziellaufwerk param1 param2<---
Voreinst.:     ln1      = Laufwerk 0 (Quelle)
               ln2      = Laufwerk 1 (Ziel)
               param1   = A    alle Spuren
               param2   = J    ja, rückfragen
Dateispez.:    -
```

Funktion:
Kopiert den gesamten Inhalt einer Diskette von Laufwerk ln1 auf Laufwerk ln2. Die Zieldiskette muß formatiert sein. Die Angabe sämtlicher Parameter ist optional. Der Medienname der Quelldiskette wird übernommen. Bei KOS 4.x ist für param1 'S' anzugeben, wenn nur die Systemspuren (0-4) kopiert werden sollen!

Zur Verhinderung von unbeabsichtigtem Löschen von Disketten stellt COPYD2 zunächst die Rückfrage:

```
Original von Medium M-0
Kopie auf Medium M-1
Medien bereit ? (J/N)
```

Diese Rückfrage kann durch die Option 'N' (nein, nicht rückfragen) umgangen werden. Der Kopiervorgang startet im Falle einer Rückfrage nach der Eingabe eines 'J', wobei Satz für Satz übertragen wird. Nach jedem Satz findet ein automatischer Vergleich zwischen Original und Kopie statt. Fehler führen zu Abbruch und Fehlermeldung. Während des Kopiervorgangs erfolgt die Anzeige der gerade kopierten Satznummer. Das Programm kann jederzeit durch die Taste 'ESC' abgebrochen werden.

Das Kommando ist nur dann sinnvoll, wenn beide Medien physikalisch gleich sind (z.B. zwei Floppy Disks).

E/A-Kanäle:

```
Eingaben        --->   Kanal E-1
Status          --->   Kanal E-1
Rückmeldungen   --->   Kanal A-1
```

Beispiele:

COPYD2<---
Kopiert alle Spuren von Laufwerk 0 auf Laufwerk 1.

COPYD2 1>0 N<---
Kopiert die Diskette im Laufwerk 1 (Orginal) nach Laufwerk 0, wobei keine Rückfrage gestellt wird.

Nur bei KOS 4.x
COPYD2 S<---
kopiert nur die ersten 4 Spuren (=Systemspuren).

Hinweis: COPYD2 stellt sich automatisch auf das jeweilige Diskettenformat ein (DD/SS, DD/DS, SD/SS).

3.4 CPFILE-Kommando (Vergleich von 2 Dateien)

Aufruf: CPFILE mn:name1.typ mn:name2.typ<---
Voreinst.: mn = Mastermedium
typ = COM
Dateispez.: EDA

Funktion:
Byteweiser Vergleich des Inhalts von zwei Dateien beliebigen Typs und beliebiger, auch unterschiedlicher Länge.
Die Anzahl und der Inhalt der unterschiedlichen Bytes wird ausgegeben.
Abbruch des Kommandos durch 'ESCAPE'.

Beispiel:

CPFILE DATEI 1:DATEI<---

Vergleicht den Inhalt der Datei DATEI.COM auf dem Mastermedium mit dem Inhalt der Datei DATEI.COM auf dem Medium 1.

3.5 DATE-Kommando (Datumseintrag/-ausgabe)

Aufruf: DATE<---
Voreinst.: -
Dateispez.: -

Funktion:
Anzeigen und eintragen eines Datums ins Betriebssystem. Das Programm arbeitet mit Benutzerführung.
Soll ein bestimmtes Datum erzeugt werden, können die notwendigen Parameter auch schon beim Aufruf übergeben werden:

DATE "Jddmmyy"

Die Angaben für das Datum müssen mit führenden Nullen übergeben werden:

dd - Tagesangabe
mm - Monatsangabe
yy - Jahresangabe

DATE fragt die Speicherstellen OEH und OFH im KOS-reservierten Bereich des Speichers ab und errechnet daraus das Datum. Dieses Datum kann in einem Dialog verändert werden. Das geänderte Datum wird kodiert in die Speicherstellen OEH/OFH übertragen und auf die dafür vorgesehenen Stellen in das Directory des Mastermediums geschrieben.

Codierung:	OEH:	Bit 0...4	Tag, hexadezimal
		Bit 5...7	Jahrzehnt, hex.
	OFH:	Bit 0...3	Monat, hexadezimal
		Bit 4...7	Jahr, hex.

3.6 DEFP - Kommando (Dateieigenschaften (Properties) definieren)

Aufruf: DEFP mn:name.typ<---
Voreinstellung: mn = Mastermedium
typ = COM
Dateispez.: EDA

Funktion:
Anzeigen und Editieren der Dateiproperties.

Die Dateieigenschaften werden in folgender Form dargestellt:

```
Eigenschaften der Datei
-----------------------
x  KOS Systemdatei
x  Datei schreibgeschützt
x  Datei löschgeschützt
x  Eigenschaften geschützt
x  Reserviert
x  Directory Datei
x  Benutzerkennzeichen
x  Geheim
```

Wobei x entweder für "-": nicht gesetzt
oder "*": gesetzt

steht.

Falls das 'Properties geschützt'-Flag nicht gesetzt ist, erfolgt das Editieren der Properties cursororientiert mit Hilfe folgender Befehle:

S	=	Setzen
R	=	Rücksetzen
<CR>	=	Unverändert übernehmen
K	=	Rückschreiben der editierten Properties und Rückkehr zu KOS

Bedeutung der einzelnen Property-Flags:

KOS Systemdatei: Darunter fallen die Dateien vom Typ COM, die als Dienstprogramme auf der KOS-Systemdiskette enthalten sind.

Datei schreibgeschützt: Ein Versuch, in die Datei zu schreiben, wird von KOS verhindert und mit der Meldung:

Datei schreib-/löschgeschützt

quittiert.

Datei löschgeschützt: KOS verhindert das Löschen der Datei. Schreibzugriffe auf diese Datei sind gewöhnlich nur über die Random-Funktion möglich.

Properties geschützt: Das DEFP-Kommando zeigt die Properties zwar an, aber ein Verändern ist nicht mehr möglich.
Das Setzen des 'Properties geschützt'-Flag ist endgültig, eine Veränderung der Properties ist daraufhin nicht mehr möglich.

Benutzerkennzeichen: Falls dieses Flag gesetzt ist, ist ein Zugriff auf die Datei nur nach Angabe des Benutzerkennzeichens möglich.
Wird dieses Flag im DEFP-Kommando gesetzt, so fordert das System die Eingabe eines Benutzerkennzeichens, das dann der Datei zugewiesen wird.

Geheim:
Die Datei wird z.B. beim I- oder IL-Kommando übergangen, wenn nicht im Kommando als Parameter 'P=S' oder 'P=*' angegeben wird.

Dateieigenschaften bei den Kommandos IL, MOVE, REN, DEL

Bei diesen Kommandos kann zur optionalen Spezifikation beliebiger Property-Kombinationen der Parameter 'P=properties' angegeben werden. Für 'properties' steht eine beliebige Folge der Zeichen: K, W, E, P, R, D, U, S, *.

Es bedeuten:

K - KOS Systemdatei
W - Datei schreibgeschützt (Write protection)
E - Datei löschgeschützt (Erase protection)
P - Properties geschützt
R - reserviert
D - Directory Datei
U - Datei hat Benutzerkennzeichen (USER-ID)
S - Datei ist geheim (Secret)
* - alle Properties

Beispiel:

Alle Dateien, die mindestens schreibgeschützt und 'geheim' sind, sollen mit dem MOVE-Kommando kopiert werden. Dazu ist folgendes Kommando notwendig:

```
MOVE * P=WS<---      oder:
MOVE * P=SW<---
```

Die **Reihenfolge der Parameter** spielt keine Rolle, solange insgesamt nicht mehr als drei Parameter benötigt werden. Bei mehr als drei Parametern muß die Spezifikation 'P=properties' als letzte eingegeben werden, andernfalls bleibt der letzte Parameter unberücksichtigt.

Beispiele für richtige Anwendungen der 'P=properties'-Option:

```
MOVE     P=KWS dateigruppe J
MOVE     dateigruppe  J P=KWS
MOVE 0:dateigruppe 2: J P=KWS
IL P=KWS dateigruppe
DEL 1:dateigruppe P=KWSU
```

Werden die Kommandos IL, MOVE, DEL und REN ohne die 'P=properties'-Option aufgerufen, so bleiben alle Dateien die geheim und/oder Directory Dateien sind, unberücksichtigt.

3.7 DEL - Kommando (Datei löschen)

```
Aufruf:        DEL mn:name.typ param<---
Voreinst.:     mn     =  Mastermedium
               typ    =  *
               param  =  J     = rückfragen
                      =  P     = nicht geheim
Dateispez.:    EDA/MDA
```

Funktion:
Löschen von einzelnen Dateien oder Dateigruppen. Im allgemeinen erfolgt vor dem Start des Löschvorgangs zunächst eine Rückfrage an den Benutzer, ob die Datei tatsächlich gelöscht werden soll (Parameter 'param' = J). Fünf Eingaben sind daraufhin möglich.

J - Die Datei wird gelöscht
Y - Die Datei wird gelöscht
N - Die Datei wird nicht gelöscht
A - Alle folgenden Dateien der angegebenen
    Spezifikation werden gelöscht;
    Rückfragen werden nicht mehr gestellt
K - Abbruch des Kommandos (KOS-Rücksprung)

Ist param = 'N' im Parameterfeld spezifiziert, so wird von Anfang an keine Rückfrage gestellt. Die gelöschten Dateien werden allerdings trotzdem ausgedruckt. Das Kommando kann jederzeit durch die Taste 'ESC' abgebrochen werden.

Bei Dateien mit Datei-Properties muß der Parameter 'P=properties' angegeben werden (siehe 'DEFP'-Kommando).

E/A-Kanäle:

```
Eingabe    ---> Kanal  E-1
Status     ---> Kanal  E-1
Ausgabe    ---> Kanal  A-1
```

Beispiele:

DEL DATEI.ABC<---
Löscht die Datei 'DATEI.ABC' nach vorheriger Rückfrage (und der Benutzereingabe 'J').

DEL *.PRN N<---
Löscht alle Dateien des Typs 'PRN' ohne Rückfragen zu stellen.

DEL *.COM P=K
Löscht alle Dateien des Typs 'COM' mit Property = K, die nicht löschgeschützt sind.

3.8 DO - Kommando (Ausführung einer Kommandodatei)

Aufruf DO mn:name.typ p1 p2 ... p9<---
Voreinst. mn = Mastermedium
typ = KMD
Dateispez.: EDA

Funktion:
Ausführung von KOS-Kommandos aus einer Kommandodatei. Eine Kommandodatei ist eine mit dem PSI/EDITOR erstellte Datei, die eine beliebige Anzahl von KOS-Kommandos enthält. Zur Trennung einzelner Kommandos innerhalb der Datei dienen die Zeichen Strichpunkt (;) sowie RETURN (Zeilenende).

DO-Kommandos dürfen beliebig ineinander verschachtelt sein. Die Schachtelungstiefe ist nur durch den zur Verfügung stehenden Arbeitsspeicher begrenzt. Vor der Ausführung der KOS-Kommandos wird der Inhalt der Kommandodatei ausgedruckt. Beliebige Parameter können beim Aufruf des DO-Kommandos übergeben werden. Als Parameterrepräsentanten innerhalb einer Kommandodatei dienen die Ausdrücke #1 #2 ... #9. Zugelassen sind maximal neun Parameter. Die Angabe von 'mn:' und '.typ' ist optional.

E/A-Kanäle:
Eingaben ---> -
Ausgaben ---> Kanal A-1

Beispiele:

DO TEST<---
Ausführung der Kommandos in der Datei TEST.KMD. Diese Datei habe beispielsweise folgenden Inhalt:

```
BASIC;M;I 1:*.BAS;EDIT TEST.KMD;I TEST.*;M
```

Das DO-Kommando lädt die Kommandodatei in den höchstmöglichen Speicherbereich, schützt diesen und ruft anschließend den Kommandointerpreter von KOS auf.

DO KDATEI 0 1<---
Im zweiten Beispiel sind die beiden Parameter '0' und '1' angegeben. Diese ersetzen in der Kommandodatei die Ausdrücke #1 bzw. #2. KDATEI.KMD enthalte beispielsweise folgenden Kommandostring:

```
M #1;I #2:*.COM
```

Der Aufruf ergibt dann: M 0;I 1:*.COM

3.9 DUMP - Kommando (Ausdruck binärer Dateiinhalte)

```
Aufruf:         DUMP mn:name.typ param<---
Voreinst.:      mn    =  Mastermedium
                typ   =  COM
                param =  O=$MON
Dateispez.:     EDA
```

Funktion:
Ausdruck des Inhalts einer eindeutig spezifizierten Datei im Hex- und ASCII-Code.
Pro Zeile wird ein 16 (=10H) Byte Block, inklusive der Anfangsadresse (1. Spalte) und der entsprechenden ASCII-Äquivalente ausgedruckt. Nicht darstellbare Bytes erscheinen im ASCII-Feld als Punkt. Der Ausgabekanal kann explizit angegeben werden.

E/A-Kanäle:

```
     Status          --->  Kanal E-1
     Fehlermeldungen --->  Kanal A-1
     Dateiausdruck   --->  Kanal A-2 oder A-6 falls O=$EAT
```

Beispiele:

DUMP TEST.PRN<---
Ausdruck der Datei TEST.PRN in hexadezimalem Format. Der Ausdruck gelangt auf Kanal A-2, der vom System dem Sichtschirm zugeordnet ist. Diesem Kanal kann durch das EAK-Kommando ein Drucker zugeordnet werden.

DUMP O=$SIOA 0:TEST<---
Ausdruck der Datei TEST.COM auf dem Ausgabetreiber $SIOA. Der Treiber muß zuvor mit dem Kommando

```
     EAK $SIOA=AKTIV<----
```

aktiviert werden; die Zuordnung auf Kanal A-6 erfolgt automatisch.

3.10 EAK - Kommando (Ein-/Ausgabetreiber-Verwaltung)

3.10.1 Übersicht und Kommandosyntax

Das Programm EAK (Ein-/Ausgabe Kanal) ermöglicht folgende Aktivitäten:

- Aktivierung eines Ein-/Ausgabetreibers (E/A-Treiber)
- Deaktivierung eines E/A-Treibers
- Aktivierung eines Medientreibers
- Deaktivierung eines Medientreibers
- Zuordnung einer Kanalnummer für E/A- oder Medientreiber
- Auflistung der aktivierten E/A-Kanäle
- Auflistung der Syntax des Programms EAK

Hierzu stehen folgende EAK - Kommandos zur Verfügung:

```
EAK $(mn:)EATN=AKTIV<---
EAK $E/ATN=DEAKTIV<---
EAK X-n=$EATN<---
EAK LIST<---
EAK SYNTAX<---
```

Bei jedem Aufruf des Programms EAK können bis zu 9 Aktivitäten als Parameter angegeben werden. Fehlt jeglicher Parameter, so werden Syntax und Beispiele für das E/AK-Kommando ausgegeben.

Beispiel:

```
EAK $EATN=AKTIV X-n=$EATN LIST<---
```

Die Abkürzungen hier und im folgenden bedeuten:

```
EATN = Ein-/Ausgabetreiber Name
MEDN = Medientreiber Name
EAT  = Ein-/Ausgabetreiber
       X    = E für Eingabekanal, A für Ausgabekanal
              M für Mediennummer
       n    = Kanalnummer (1...9) auch log. Gerätenummer
```

3.10.2 E/A-Treiber Aktivierung

Aufruf: EAK $mn:EATN=AKTIV<---
Voreinst.: mn = Mastermedium

Funktion:
Aktivierung des Treibers EATN bzw. MEDN. Das Kommando ist mit einem 'Relocater' ausgerüstet, der E/A- oder Medientreiber automatisch in den höchstmöglichen Speicherbereich lädt.

Es ist deshalb nicht notwendig, Treiber zu linken. Das Programm EAK sucht zunächst die OBJ-Datei eines Treibers und - erst wenn dieses nicht gefunden wurde - die EAT-Datei. Für den Relocater darf der E/A-Treiber nur aus einem einzigen Modul bestehen und außerdem weder Globals noch Externals enthalten.

Die Aktivierung bewirkt folgendes:

a) Laden der Datei EATN.OBJ.
   Diese muß den Konventionen für E/A-Treiber
   (s. E/A-Treiberformat, Utility-Beschreibung) entsprechen
b) Schutz des durch die Datei EATN.OBJ beanspruchten
   Speicherbereichs
c) Eintrag des Namens EATN in eine Tabelle der E/A-Verwaltung
d) Ausführung der im E/A-Treiber enthaltenen
   Kanalinitialisierungsroutine ('INIT')

Hinweis:
Dem Treiber EATN wird durch die Aktivierung noch keine Kanalnummer zugeordnet.

3.10.3 E/A-Treiber Deaktivierung

Aufruf: EAK $EATN=DEAKTIV<---
Voreinst.: -

Funktion:
Deaktivierung des Ein-/Ausgabetreibers EATN. Dieses Kommando bewirkt folgendes:

a) Freigabe des durch die Datei EATN beanspruchten
   Speicherbereichs
b) Löschen des Namens EATN aus der E/A-Tabelle der
   E/A - Verwaltung
c) Ausführung der im E/A-Treiber enthaltenen
   'CLOSE'-Routine

3.10.4 Auflistung der aktiven Treiber

Aufruf: EAK LIST<---

Funktion:
Auflistung der momentan aktivierten Ein-/Ausgabetreiber, inklusive der dazugehörigen Kanalnummern und Startadressen.

Nach dem Laden des Betriebssystems sind folgende E/A- bzw. Medientreiber aktiviert:

```
$KEY  E-0     --->  Tastatureingabe    (Keyboard)
$KEY  E-1     --->  Tastatureingabe
$MON  A-0     --->  Sichtschirmausgabe (Monitor)
$MON  A-1     --->  Sichtschirmausgabe
$MON  A-2     --->  Sichtschirmausgabe
$KSM  A-3     --->  System-/Fehlermeldungen Ausgabe
$DSK0 M-0     --->  Floppy Disk (rechtes Laufwerk) *
$DSK1 M-1     --->  Floppy Disk (linkes  Laufwerk) *
```

3.10.5 Kanalzuweisung

Aufruf: EAK x-n=$EATN

Funktion:
Weist dem Kanal n einem Treiber zu. Die Kanäle E-0 und A-0 sind fest den Treibern $KEY und $MON zugewiesen und können nicht umdefiniert werden.

Allgemeines Kommandoformat für die Zuweisung:

```
EAK E-n=$EATN   ; für Eingabekanäle  (1 < n < 9)
EAK A-n=$EATN   ; für Ausgabekanäle  (1 < n < 9)
EAK M-n=$MEDN   ; für Medientreiber  (0 < n < 9)
```

* $WIN0 bzw. $WIN1 bei KOS 5.3/MW5

3.11 FORMAT - Kommando (Diskette formatieren)

Format: FORMAT mn<---
Voreinst.: -
Dateispez.: -

Funktion:
Formatiert eine softsektorierte Diskette (16 Sektoren pro Spur). Ist die Laufwerknummer mn nicht spezifiziert oder unzulässig, so stellt FORMAT die Rückfrage:

Auf welchem Laufwerk soll formatiert werden? (0/1)

FORMAT fragt nach dem Identifikationsnamen, der die Diskette kennzeichnen soll. Dieser Name darf maximal 8 Zeichen lang sein; gleiche Namen sollten nach Möglichkeit vermieden werden. Bei COPYD2 wird der Name der Quelldiskette in die Kopie eingetragen.

Zum Start muß auf die Frage:

Diskette in Laufwerk n bereit? (J/N)

die Taste 'J' gedrückt werden.

Andere Eingaben brechen das FORMAT-Programm ab (KOS-Rücksprung). Die Nummer der gerade formatierten Spur wird angezeigt.

E/A-Kanäle:

Eingaben ---> Kanal E-1
Ausgaben ---> Kanal A-0

Beispiele:

FORMAT 1<---
Formatiert die Diskette in Laufwerk 1 (ohne Rückfrage 1)

FORMAT<---
Formatiert eine Diskette mit allen Rückfragen.

Hinweis: FORMAT hält die Hintergrundverarbeitung an.
FORMAT stellt sich automatisch auf die jeweilige Laufwerkskonfiguration des PSI80-Systems ein.

3.12 IL - Kommando (Inhaltsausgabe im Langformat)

Aufruf:	IL mn:name.typ param<---	
Voreinst.:	mn	= Mastermedium
	name	= *
	typ	= *
	param	= P=nicht geheim
		= O=$MON
Dateispez.:	EDA/MDA	

Funktion:
Ausdruck des Inhaltsverzeichnis eines Mediums in langem Format. Neben dem Dateinamen/-Typ wird die Anzahl der von der jeweiligen Datei belegten Sektoren sowie die dafür auf dem Medium reservierten Blöcke (Vielfache von 1k Byte) ausgegeben. Gewünschte Datei-Properties sind anzugegeben (siehe 'DEFP'-Kommado); der Ausgabekanal kann explizit angegeben werden.

E/A-Kanäle:

Status	--->	Kanal E-1
Fehlermeldungen	--->	Kanal A-1
Inhaltsverzeichnis	--->	Kanal A-2 oder A-6 falls O=$EAT

Beispiele:

IL TEST.*<---
Auflistung aller nicht geheimen Dateien des Namens TEST (Typ beliebig) des Mastermediums.

IL 1: P=* O=$SIOB
Auflistung aller Dateien von Medium 1 auf den Ausgabetreiber $SIOB. Die Zuordnung auf Kanal A-6 erfolgt automatisch.

IL P=D
Ausgabe der Directory-Datei des Masterlaufwerks.

3.13 INFO-Kommando (Ausgabe einer ASCII-Datei)

```
Aufruf:        INFO mn:name.typ<---
Voreinst.:     mn     = Mastermedium
               typ    = INF
Dateispez.:    EDA
```

Funktion:
Ausgabe einer ASCII-Datei auf den Sichtschirm des PSI80-Computers (oder auf ein beliebiges anderes Peripheriegerät). Die Ausgabe erfolgt bildschirm-orientiert:

```
1 Seite  =  24 Zeilen
1 Zeile  =  Folge von ASCII-Zeichen mit CR abgeschlossen
```

mögliche Eingaben:

```
- Leertaste   ---> nächste Seite
- Return      ---> eine Seite zurück
- ESC         ---> Programm abbrechen
```

Status- und Fehlermeldungen:

```
?             ---> Datei nicht gefunden
*             ---> Datei kann nicht geladen werden
'Ende'        ---> letzte Seite
```

E/A-Kanäle:

```
Status          ---> Kanal E-1
Fehlermeldungen ---> Kanal A-1
Dateiausdruck   ---> Kanal A-2
```

Beispiele:

INFO KOS<---
Ausgabe der Datei KOS.INF auf den Bildschirm.

INFO<---
Ausgabe der ersten Datei des Masterlaufwerkes vom Typ '.INF'.

3.14 INISER - Kommando (Initialisierung von Serienschnittstellen)

Aufruf: INISER param<---
Voreinst. param = -

Funktion:
Programmierung der PSI80-Serienschnittstellen A und B für asynchrone Übertragung. Alle hierfür relevanten Parameter wie Baudrate, Parity etc. sind einstellbar. 'INISER' arbeitet mit Benutzerführung (falls als Parameter 'param' = P gegeben wurde) und ermöglicht die Abspeicherung von einmal eingegebenen Initialisierungsparametern auf Diskette. Eine neuerliche Initialisierung mit denselben Parametern kann nun beliebig oft ohne neuerliche Eingaben durchgeführt werden durch den Aufruf "INISER<---".

E/A-Kanäle:
Eingaben ---> Kanal E-1
Ausgaben ---> Kanal A-1

Beispiele:

INISER<---
Programmiert eine der Serienschnittstellen (Kanal A oder B) entsprechend der momentan auf Diskette (Platte) abgespeicherten Initialisierungsparameter.

INISER P<--
Alle Parameter werden neu erfragt. Sie können bei Bedarf auf Diskette (Platte) abgelegt werden.

3.15 MAP - Kommando (Auskunft über Speicherbelegung)

Aufruf: MAP<---
Voreinst.: -

Funktion:
Ausdruck der aktuellen Speicherbelegung von KOS. Zur Verwaltung des Speichers teilt KOS den vorhandenen Speicher in Segmente zu je 80H Byte ein. Ein 'B' im Ausdruck kennzeichnet ein belegtes, ein '.' ein freies Segment. Bei folgenden Gelegenheiten werden Speichersegmente belegt:

- Laden eines Programms
- Aktivierung eines E/A-Treibers
- Ausführung einer Kommandodatei (DO - Kommando)
- Ausführung des A-Kommandos

Die Freigabe der entsprechenden Segmente erfolgt analog dazu bei:

- der Rückkehr von einem Programm
- der Deaktivierung eines E/A-Treibers
- dem Ende einer Kommandodatei-Abarbeitung
- der Ausführung des D-Kommandos

E/A-Kanäle:
Eingaben: ---> Kanal E-1
Ausgaben: ---> Kanal A-1

3.16 MOVE - Kommando (Kopieren von Dateien)

Aufruf:	MOVE quellmedium:name.typ zielmedium param<---		
Voreinst.	quellmedium	=	0
	typ	=	*
	zielmedium	=	1 falls quellmedium = 0 und
			0 falls quellaufwerk = 1
	param	=	N nein, nicht rückfragen
		=	P=nicht geheim
Dateispez.	EDA/MDA		

Funktion:
Kopieren von einzelnen Dateien oder ganzen Dateigruppen. Der Dateiname bleibt hierbei erhalten. Bereits vorhandene Dateien des angegebenen Namens werden überschrieben. Die Angabe von quellmedium:, .typ, zielmedium und param ist optional, die Reihenfolge von zielmedium und param ist beliebig.

Ist für param 'J' angegeben, so erfolgt vor dem Start jedes Kopiervorgangs eine Rückfrage an den Benutzer. Fünf Eingaben sind daraufhin möglich.

- J - Die Datei wird kopiert
- Y - Die Datei wird kopiert
- N - Die Datei wird nicht kopiert
- A - Alle folgenden Dateien der angegebenen Spezifikation werden kopiert. Rückfragen werden nicht mehr gestellt.
- K - Abbruch des Kommandos (KOS-Rücksprung) (dies ist auch mit der Taste 'ESC' möglich)

Im Unterschied zum COPY-Kommando können mit dem MOVE-Kommando neben einzelnen Dateien auch Dateigruppen kopiert werden. Bei Dateien mit Property muß der Paramter 'P=properties' angegeben werden (siehe 'DEFP'-Kommando).

E/A-Kanäle:

Eingabe	--->	Kanal E-1
Status	--->	Kanal E-1
Ausgabe	--->	Kanal A-1

Beispiele:

MOVE *<---
Kopiert alle nicht geheimen Dateien von Medium 0 auf Medium 1.

MOVE 1:TEST.* J P=*<---
Kopiert alle Dateien des Namens 'TEST' (Typ beliebig) von Medium 1 auf Medium 0, param = 'J' bewirkt eine Rückfrage, so daß der Anwender entscheiden kann, ob eine bestimmte Datei übertragen werden soll oder nicht.

3.17 PRINT - Kommando (Ausgabe einer ASCII-Datei)

```
Aufruf:       PRINT mn:name.typ param<---
Voreinst.:    mn        = Mastermedium
              typ       = PRN
              param     = O=$MON
Dateispez.:   EDA
```

Funktion:
Ausgabe einer ASCII-Datei auf den Sichtschirm des PSI80-Computers oder auf ein beliebiges anderes Peripheriegerät. Die Angabe von mn: und .typ ist optional. Der Ausgabekanal kann explizit angegeben werden.

Es wird überprüft, ob die spezifizierte Datei ausschließlich ASCII-Zeichen enthält. Ist dies nicht der Fall, so erfolgt der Ausdruck bis zum ersten Nicht-ASCII-Zeichen und der Hinweis:

```
    ---> PRINT: Datei enthält unzulässige (NICHT-ASCII-) Zeichen
```

E/A-Kanäle:

Status	---> Kanal E-1
Fehlermeldungen	---> Kanal A-1
Dateiausdruck	---> Kanal A-2 oder A-6 falls O=$EAT

Beispiele:

```
PRINT 1:TEST.SRC<---
```
Ausdruck der Datei TEST.SRC von Medium 1 auf den Bildschirm.

```
PRINT TEST.PRN O=$SIOA<---
```
Ausdruck der Datei TEST.PRN auf den Ausgabetreiber $SIOA. Die Zuweisung auf Kanal A-6 erfolgt automatisch.

3.18 REN - Kommando (RENAME = Umbenennung einer Datei)

```
Aufruf:          REN mn:namealt.typ nameneu.typ param<---
Voreinst.:       mn     = Mastermedium
                 typ    = *
                 param  = N (nein, nicht rückfragen)
                        = P=nicht geheim
Dateispez.:      EDA (Typ auch mehrdeutig)
```

Funktion:
Umbenennung der Datei namealt.typ in nameneu.typ. Bei der Datei namealt kann der Typ durch den Universalbezeichner '*' ersetzt werden. In solchen Fällen werden alle Dateien des Namens namealt umbenannt. Die Namen der umbenannten Dateien werden ausgegeben. Ist für param = J angegeben, so erfolgt vor der Umbenennung eine Rückfrage an den Benutzer, worauf fünf Eingaben möglich sind:

- J - Die Umbenennung wird durchgeführt
- Y - Die Umbenennung wird durchgeführt
- N - Die Umbenennung wird nicht durchgeführt
- A - Alle folgenden Dateien der angegeben Spezifikation werden (ohne neuerliche Rückfragen) umbenannt.
- K - Abbruch des Kommandos (KOS-Rücksprung)

Die Datei-Properties müssen angegeben werden (siehe 'DEFP'-Kommando).

E/A-Kanäle:

```
Eingabe          ---> Kanal E-1
Status           ---> Kanal E-1
Rückmeldungen    ---> Kanal A-1
```

Beispiele:

```
REN BASIC.COM BASICNEU.COM<---
```
Die Datei 'BASIC.COM' wird in 'BASICNEU.COM' umbenannt. Eine Rückfrage wird nicht gestellt.

```
REN TEST.* TEST1 J<---
```
Alle Dateien des Namens 'TEST' (Typ beliebig) werden in 'TEST1' umbenannt, wobei zuvor jeweils eine Rückfrage gestellt wird.

```
REN *.SRC *.MAC<---
```
Alle Dateien mit Typ 'SRC' werden in Dateien mit Typ 'MAC' umbenannt.

```
REN ALT.DIR NEU P=D<---
```
Ändern des Mediennamens.

3.19 RLOAD - Kommando (Lade relokativ)

Aufruf: RLOAD mn:name.typ
Voreinst.: mn = Mastermedium
typ = OBJ
Dateispez.: EDA

Funktion:
Die spezifizierte Objektdatei 'name' wird geladen und relokiert (gelinkt). Bei der Objektdatei muß es sich um ein Modul ohne Externals und Globals handeln. Der vom Relokator erzeugte Code wird so hoch wie möglich im Speicher abgelegt und der entsprechende Speicherplatz wird belegt.

Abschließend wird das erzeugte Programm gestartet.

3.20 SELECT - Kommando

```
Aufruf:       RLOAD SELECT mn:programmliste.typ
Voreinst.:    mn            = Master
              programmliste = PROGLIST
              typ           = SEL
Dateispez.:   EDA
```

Funktion:
Ermöglicht unter Benutzerführung das Verzweigen in bis zu zehn verschiedene Programme. Die Programmliste ist eine eigenständige Datei und wird über den Editor erstellt. Für jedes auszuführende Programm enthält diese Liste zwei Zeilen. In der ersten Zeile steht ein beliebiger benutzerdefinierbarer Text, der vom SELECT-Kommando angezeigt wird. In der zweiten Zeile steht die dazugehörige KOS-Kommandozeile. Enthält die Programmliste eine ungerade Zeilenzahl oder mehr als 20 Zeilen, so ist die Datei für das SELECT-Kommando nicht auswertbar. Die Angabe der Mediennummer und des Typs ist optional.

Eine baumartige Verschachtelung des SELECT-Kommandos ist möglich und nur durch die Kapazität des Systemspeichers begrenzt.

Beispiel einer Programmliste:

```
Neueste Informationen
INFO 0:KOS
Serielle Schnittstelle für 4800 Baud initialisieren
EAK $SIOA=AKTIV; INISER P "181432NJ"
Mondlandung spielen
BASIC "LOAD MONDLAND<RUN"
```

Der Name dieser Datei sei 'LIST.SEL', damit lautet der Aufruf:

```
RLOAD SELECT LIST
```

3.21 SPOOL - Kommando (Ausgabeaufträge annehmen)*

Aufruf: SPOOL dateiname $EATN
oder SPOOL =LIST
Voreinst.: -
Dateispez.: EDA

Funktion:
Einfügen eines Ausgabeauftrages in eine Warteschlange. Die Datei wird eröffnet. Es wird geprüft, ob der E/A-Treiber aktiviert ist. Anschließend wird der Zeiger auf den DSB der Datei und der Treibername an die TASK 'SPOOL' übergeben. Maximal können 10 Aufträge in die Warteschlange aufgenommen werden. Die Task 'SPOOL' muß mit dem Kommando 'RLOAD PTASK' aktiviert sein. Diese Task gibt die Dateien aus der Warteschlange nacheinander durch die angegebenen Treiber in der Hintergrundverarbeitung aus.

Mit dem Kommando 'SPOOL =LIST' wird die Liste der Dateien in der Warteschlange ausgegeben.

Anmerkungen:

1. Während des Ausdrucks der Datei darf diese nicht im Vordergrund geschlossen werden. Die Kommandos 'N', 'F' und 'STATUS' schließen alle Dateien auf allen Medien.

2. Geöffnete Dateien sollten nicht anderweitig bearbeitet werden. Es ist z.B. zwar möglich, aber unsinnig, die Datei TEST.PRN im Hintergrund auszudrucken, und gleichzeitig im Vordergrund die Datei TEST.SRC mit dem Befehl 'ASM =TEST/L' zu assemblieren.

E/A-Kanäle:

Fehlermeldung A-1

Beispiele:

SPOOL 1:TEST.PRN $SIO
Die Datei TEST.PRN soll durch den E/A-Treiber '$SIO' ausgegeben werden. Sie wird in die Warteschlange aufgenommen.

SPOOL =LIST
Mit diesem Kommando wird die Liste der noch wartenden Aufträge ausgegeben.

* Nicht bei KOS 5.0x

3.22 STATUS - Kommando (Information über Medienbelegung)

Aufruf: STATUS mn<---
Voreinst.: -
Dateispez.: -

Funktion:
Ausgabe der Medienbelegung entsprechend des untenstehenden Beispiels. Vor dem Ausdruck führt 'STATUS' automatisch eine Neuinitialisierung aller aktiven Medien durch. Bei Angabe der Mediennummer mn (0<mn<9) wird nur der Status dieses Mediums ausgegeben.

E/A-Kanäle:

Ausgaben ---> Kanal A-1

Beispiele:

STATUS<---
Ausgabe des Status aller aktiven Medien.

STATUS2<---
Ausgabe des Status von Medium 2.

Beispiel:

STATUS:

Tabelle der Speichermedien

Kanal	Treiber	Medien-Name	Kapazität	belegt	frei	Recordl.
0	$DSK0	KOS53DDS	308k	275	33	256 Byte
1	$DSK1	UTI53DDS	308k	175	133	256 Byte

3.23 STOP - Kommando (Programmierte Unterbrechung von Kommandodateien)

Aufruf: STOP param<---
Voreinst.: param = J
Dateispez.: -

Funktion:
Das Kommando 'STOP' führt eine Unterbrechung der Ausführung von Kommandodateien oder Kommandozeilen durch und ermöglicht danach einen Abbruck der Abarbeitung der Kommandozeile oder -datei. Hierzu stellt 'STOP' die Rückfrage:

<--- STOP: Kommandozeile/datei abbrechen? (J/ESC)

und wartet anschließend auf eine Benutzereingabe. Bei den Tasten 'J' und 'ESC' werden weitere Kommandos in der Kommandozeile oder -datei nicht mehr ausgeführt. Bei allen anderen Zeichen werden die Kommandos nach dem STOP-Kommando weiter bearbeitet. Ist die Option 'N' angegeben, so unterbricht 'STOP' nur dann, wenn irgendeine Taste gedrückt war.

E/A-Kanäle:

Eingaben ---> Kanal E-1
Ausgaben ---> Kanal A-1

Beispiel:

EDIT TEST.SRC;ASM =TEST/L;STOP;LINK TEST/N,TEST/E<---
Das LINK-Kommando wird nicht ausgeführt, wenn beim STOP-Kommando die Tasten 'J' oder 'ESC' gedrückt werden. (Sinnvoll, wenn beim ASM-Lauf Fehler auftraten).

Eine weitere Möglichkeit, die Fortführung eines DO-Kommandos zu verhindern, ist die Eingabe von CTRL-C.

3.24 TASK - Kommando (Information über Hintergrundverarbeitung)*

Aufruf: TASK<---
Voreinst.: -
Dateispez.: -

Funktion:
Information über Hintergrundverarbeitung und Funktionsausführungen. Die Liste der aktiven Hintergrund-Tasks wird angezeigt. Außer dem Namen werden folgende Parameter einer TASK ausgegeben.

STATUS	-	Status der TASK
AMS	-	Anzahl belegter Memory-Segmente
PCNT	-	Preset-Counter
TEP	-	TASK Entry Point
TLA	-	Task Load Address

Die Bedeutung der Parameter sind im Kapitel Hintergrundverarbeitung beschrieben.

Anschließend wird ein Funktionscode abgefragt, mit dem verschiedene Aktionen, wie Deaktivieren oder Reaktivieren, aufgerufen werden können:

* nicht bei KOS 5.0x

Funktionscode:

1 = Nach KOS zurückspringen
Beenden des Programms TASK und Rückkehr zum Betriebssystem KOS

2 = Task deaktivieren
Task-Nr. wird abgefragt und anschließend deaktiviert, d.h. aus der Liste der Hintergrund-Tasks gestrichen und der belegte Speicherbereich freigegeben.

3 = Task temporär deaktivieren
Task-Nr. wird abgefragt und anschließend nur zeitweise deaktiviert, d.h. sie wird nicht aus der Liste gestrichen, sondern vorübergehend nicht ausgeführt.

4 = Task reaktivieren
Task-Nr. wird abgefragt und eine temporär deaktivierte Task wird wieder aktiviert, d.h. die Task wird wieder ausgeführt.

5 = Prioritäten ändern
Prioritätsliste wird abgefragt und danach die Tasks umsortiert. Die Prioritätsliste besteht aus den Task-Nr. Nicht angegebene Tasks werden angeschlossen.

6 = Parameter setzen
Task-Nr. wird abgefragt und eine Parameterübergabe an die Task wird ausgeführt.-

Wenn eine Task-Nr. abgefragt wird, kann anstatt mit der Nr. mit der Eingabe des Zeichens 'K' zum Funktionsverteiler zurückgesprungen werden.

E/A-Kanäle:
Eingabe ---> E-0
Ausgabe ---> A-0

Beispiel:

TASK<---
Gibt die Liste der Hintergrund-Tasks aus. Anschließend können die verschiedenen Funktionen angewählt werden.

4. Kommandos für Hintergrundverarbeitung *

Mit dem Kommando 'RLOAD' können OBJ-Dateien geladen und relokiert werden. Dies ist insbesondere in Verbindung mit der Vordergrund/Hintergrund-Verarbeitung erforderlich. Folgende Programme mit Backgroundtasks sind auf der Systemdiskette vorhanden:

PTASK
TIME

4.1 PTASK-Kommando (Ausgabe von ASCII-Dateien in der Hintergrundverarbeitung) *

Aufruf: RLOAD PTASK
Voreinst.: -
Dateispez.: -

Funktion:
Aktivierung der Task 'SPOOL' und Ausgabe von ASCII-Dateien in der Hintergrundverarbeitung durch Ausgabetreiber.

Vor dem erstmaligen Aufruf des Kommandos 'SPOOL' muß durch 'PTASK' die Task 'SPOOL' aktiviert werden. Die Datei- und Treibernamen für die Ausgabe werden durch das Programm 'SPOOL' in eine Warteschlange eingetragen. 'PTASK' verwaltet diese Warteschlange. (Die Task 'SPOOL' ist ein Teil des Kommandos 'PTASK' und darf nicht mit dem 'SPOOL'-Kommando verwechselt werden).

Die Datei und der Treiber des ersten Auftrags in der Warteschlange werden geöffnet. Dieser Auftrag wird dann in der Warteschlange gestrichen. Bei jeder Taskausführung werden dem Treiber Zeichen zur Ausgabe übergeben, bis dieser 'Busy' meldet. Wenn EOF der Datei erkannt wird, werden die Datei und der Treiber geschlossen. Der nächste Auftrag wird ausgeführt.

* nicht bei KOS 5.0X

Anforderungen an den Ausgabetreiber:
folgende E/A-Funktionen müssen implementiert sein.

Funktion 81: Ausgabestatus

A = 0 Nur Status abfragen
Z = 0 BUSY
Z = 1 NICHT BUSY

A = 1 Status abfragen
Z = 0 BUSY
Z = 1 NICHT BUSY

Wenn NICHT BUSY, Zeichen in Register L ausgeben.

Funktion 8D: INIT, OPEN, CLOSE

A = 0 INIT
A = 1 OPEN
A = 2 CLOSE

E/A-Kanäle:

Ausgabetreiber ---> A-8
Ausgaben ---> A-1

Beispiel:

Aufruf: RLOAD PTASK<---

Die Task 'SPOOL' ist aktiviert. Die Liste der Aufträge in der Warteschlange ist leer. Erst wenn durch das Kommando SPOOL Aufträge übergeben werden, werden diese im Hintergrund abgearbeitet.

4.2 TIME-Kommando (Ausgabe der Uhrzeit auf dem Bildschirm) *

Aufruf: RLOAD TIME
Voreinst.: -
Dateispez.: -

Funktion:
Aktivierung des Programms 'TIME', die in der Hintergrundverarbeitung eine Uhr softwaremäßig nachbildet. Die Task meldet sich mit der Aufforderung:

Bitte Uhrzeit eingeben (hh:mm:ss):

Die Eingabe von Minuten und Sekunden ist nicht zwingend vorgeschrieben. Als Trennzeichen sind :', ':' und ' ' zulässig.

Hinweis: Mit 'RLOAD TIME' werden die Tasks 'CLOCK' und 'DISPLAY' aktiviert. Die Task 'CLOCK' sollte die am höchsten priorisierte Task sein. Dies wird automatisch erreicht, wenn 'TIME' vor anderen Hintergrundprogrammen aufgerufen wird.

Die am Bildschirm angezeigte Uhrzeit ist im Arbeitsberich von KOS unter folgenden Adressen hinterlegt:

Adresse (hex)	Inhalt (hex)
001C	Sekunden
001D	Minuten
001E	Stunden

Der Quellcode dieses Programms ist als Beispiel für ein TASK-orientiertes Programm auf der Utility-Diskette enthalten.

* nicht bei KOS 5.0x

PSI ψ 80 — Utility-Diskette

Version: 4.3/5.3

1. August 1981

Dieser Teil des Handbuches beschreibt die mit PSI80 ausgelieferten Hilfsprogramme (="Utilities") für den Systemprogramierer: das Grafikpaket und Basistreiber. Beispielprogramme und die Beschreibung der Hardwaretestprogramme schließen sich an.

Inhaltsverzeichnis

1. Übersicht

Die Hilfs-Diskette enthält folgende Programme:

PSIA	SRC		Serieller Treiber
PSIA	OBJ		
PSIB	SRC		Serieller Treiber
PSIB	OBJ		
SIOA	SRC		Serieller Treiber
SIOA	OBJ		
SIOB	SRC		Serieller Treiber
SIOB	OBJ		
OKI	SRC		Druckertreiber
OKI	OBJ		
MAKEDEU	KMD		
MAKEENG	KMD		
MAKESER	KMD		
MAKEPAR	KMD		
MAKEOBJ	KMD		
PIO	SRC		Allgemeiner paralleler Treiber
PIO	OBJ		
GRAPHB	OBJ		Grafikpaket
GRAPHV	OBJ		
GRAPHA	OBJ		
GRAPHD	SRC		
GRAPHD	OBJ		
GRAPHD	COM		Grafik-Beispielprogramm
GRALINK	KMD		
GRAP	OBJ		Grafiktreiber
GRAPTASK	OBJ	(nur KOS 5.x)	
KDM	OBJ		Debug Monitor
KDMMSG	OBJ		
KDMLINK	KMD		
BASIC	KMD		
TIME	SRC		Beispielprogramme
MAGIC	BAS		
MONDLAND	BAS		
HURKLE	BAS		
PIANO	BAS	(nur KOS 5.x)	
TONLEIT	DAT	(nur KOS 5.x)	
INFO	SRC		
INFOMSG	SRC		
INFO	OBJ		
INFOMSG	OBJ		
MAKE	GER		
MAKE	ENG		
MEM64	COM		Hardwaretestprogramme
PSITEST	COM		
UTILITY	INF		Ergänzende Informationen

2. PSI80-Grafikpaket

2.1 Einführung

Das Grafik-Paket bietet die Möglichkeit

- Punkte
- Vektoren
- Alphanumerische Zeichen

im Grafik-Mode darzustellen. Die Auflösung beträgt 512 Spalten x 256 Zeilen.

Das Grafik-Paket besteht aus 3 Moduln in relokativem Object-Code. Diese können durch LINK mit OBJ-Moduln aus **ASSEMBLER, FORTRAN, BASIC-Compiler BASCOM und BASIC-Interpreter MBASIC** verknüpft werden.

Für den BASIC-Interpreter MBASIC sind die Grafikmodule passend zu linken.

Die Parameterübergabe entspricht den FORTRAN-Konventionen. Jeder Parameter ist 2 Byte lang.

Die Grafik-Funktionen werden wie Unterprogramme aufgerufen. Sie sind vorher der Sprache entsprechend zu deklarieren.

In **PSI/BASIC** sind spezielle Grafik-Befehle implementiert; das Grafik-Paket ist hierfür zusammen mit dem Schnittstellenmodul BG.OBJ in dem Treiber **GRAP.OBJ** enthalten.

Die Aktivierung erfolgt über die Kommandodatei **BASIC.KMD** durch den Aufruf:

```
DO BASIC<---
```

Die Kommandodatei BASIC.KMD enthält alle hierzu nötigen Kommandos:

```
für KOS 4.3:  EAK $GRAP=AKTIV
              BASIC

für KOS 5.3:  RLOAD GRAPTASK
              EAK $GRAP=AKTIV
              BASIC
```

Beim Betriebssystem KOS 5.x sorgt die Task 'GRAPTASK' dafür, daß die Grafikmodule auch auf den Adressen 8000H...C000H (Bildwiederholspeicher) ablauffähig sind. Daher ist diese Task für Grafikausgaben zu aktivieren.

2.2 Programmieren mit dem PSI80 Grafik-Paket

Die Grafik-Software gliedert sich in Grafik-Basis-Software (Modul GRAPHB) und Grafik-Aufbau-Software (Moduln GRAPHV,GRAPHA). Zum Betrieb der Grafik-Aufbau-Software ist die Grafik-Basis-Software erforderlich.

Initialisierungen (GRAPHB)

INITGR
Funktion: Initialisiert die graphische Betriebsart
Löscht den Bildspeicher

INITAL
Funktion: Initialisiert die alphanumerische Betriebsart
Löscht den Bildspeicher

Wichtig:
Mehrfaches Aufrufen des gleichen Initialisierungsprogramms (INITGR, INITAL) ist nicht zulässig!

CLEARV
Funktion: Löscht den Bildspeicher in graphischer Betriebsart

Punkt-Manipulationen (GRAPHB)

Parameterübergabe: Adresse von x in HL
Adresse von y in DE
x (0...511); y (0...255)

SETXY

Funktion: Setzt den durch die Koordinaten x,y bestimmten Punkt

RESXY

Funktion: Löscht den durch die Koordinaten x,y bestimmten Punkt

INVXY

Funktion: Invertiert den durch die Koordinaten x,y bestimmten Punkt

TESTXY

Funktion: liefert in A=0, wenn Punkt dunkel
ungleich 0, wenn Punkt hell

Plotten (GRAPHV)

PLOT
Parameterübergabe: Adresse von x in HL
Adresse von y in DE
Adresse von PEN in BC

Funktion: Schreibt Vektor bzw. Rechteck vom letzten Plot-Punkt zu dem Punkt mit den Koordinaten x,y.

Für 'PEN' gilt:
P = 0: Punkt anfahren ohne Zeichnen
P = 1: Vektor zeichnen
P = 2: Vektor löschen
P = 3: Vektor invertieren
P = 4: Rechteckrahmen zeichnen
P = 5: Rechteckrahmen löschen
P = 6: Rechteckrahmen invertieren
P = 7: Rechteckfläche zeichnen
P = 8: Rechteckfläche löschen
P = 9: Rechteckfläche invertieren

Es ist einzuhalten: x < 512 und y < 256.

Alphanumerische Zeichen (GRAPHA)

SYMBOL
Parameterübergabe: Adresse von x in HL (x < 512)
Adresse von y in DE (y < 256)
Zeiger auf Adressenliste in BC

Die Adressenliste enthält:

Adresse von FAKTOR
Adresse von TEXT
Adresse von RICHTG
Adresse von LÄNGE

FAKTOR (1...15); RICHTG (0,90,180,270) Grad

Funktion: TEXT ist die Anfangsadresse eines Strings mit der Länge LÄNGE. Dieser wird um den Faktor FAKTOR vergrössert in der Richtung RICHTG auf den Bildschirm geschrieben. Der Punkt mit den Koordinaten x,y ist der linke untere Eckpunkt des 1.Zeichens.

2.3 Einbau der Grafik-Moduln in Anwenderprogramme

2.3.1 Grafik in ASSEMBLER-Programmen

Zur Anwendung sind 2 Schritte erforderlich

- Bereitstellung der Parameter
- Aufruf der Funktion

Beispiel: Zeichnen eines Vektors von (100,100) bis (200,200)

```
EXTERNAL INITGR, INITAL, PLOT

TEST:
       CALL INITGR

       LD HL,100        ; Parameter laden
       LD (X1),HL
       LD (Y1),HL
       LD HL,200
       LD (X2),HL
       LD (Y2),HL

       LD HL,X1         ;PLOT 100,100,0
       LD DE,Y1
       LD BC,PENUP
       CALL PLOT

       LD HL,X2         ;PLOT 200,200,1
       LD DE,Y2
       LD BC,PENDOWN
       CALL PLOT

       CALL INITAL
       RET

PENUP: DEFW    0
PENDOWN:DEFW   1

X1:    DEFS 2          ;PARAMETER
Y1:    DEFS 2
X2:    DEFS 2
Y2:    DEFS 2

       END  TEST
```

Siehe auch das Beispielprogramm GRAPHD. Darin wird die Verwendung jeder Grafik-Funktion gezeigt.

Übersetzen und Binden eines Grafik-Programms:

Beispiel:
```
ASM =GRAPHD<---
DO GRALINK GRAPHD<---
```

erzeugt die Dateien: GRAPHD.OBJ
GRAPHD.COM

Aufbau der Kommandodatei GRALINK.KMD:

```
0:EDIT #1.SRC
0:ASM  =#1
0:LINK #1/N,GRAPHB,GRAPHV,GRAPHA,#1/E
```

2.3.2 Grafik in FORTRAN- und BASCOM-Programmen

Grafik-Funktionsaufrufe werden in FORTRAN und Basic-Compiler BASCOM wie externe Unterprogramme behandelt. Die Reihenfolge der Parameter entspricht der bei 'Parameterübergabe' beschriebenen.

Beispiel für FORTRAN:

```
      EXTERNAL INITGR,INITAL,PLOT,SETXY
      INTEGER PEN
      CALL INITGR
      X1=0.0
      Y1=0.0
      PEN=0
      CALL SETXY(X1,Y1)
      CALL PLOT(X1,Y1,PEN)
      X2=200.0
      Y2=200.0
      PEN=1
      CALL PLOT(X2,Y2,PEN)
      DO 31 I=1,4900
      DO 20 I=1,10900
20    B=10*100
31    A=10*100
      CALL INITAL
      END
```

Ein Programm für die Einbindung der Graphiksoftware in BASCOM verwendet die Aufrufe entsprechend:

```
100 X1% = 100
200 X1% = 100
300 PEN% = 0
400 CALL INITGR
500 CALL PLOT (X1%, Y1%, PEN%)
600 CALL INITAL
```

Beim Linken des BASCOM-Programmes sind die Graphikmodule GRAPHB, GRAPHV, GRAPHA mit einzubinden.

2.3.3. Grafik in MBASIC-Programmen

Alle Graphik-Funktionen des PSI80 Graphikpaketes außer SYMBOL sind von MBASIC aus ansprechbar.
Dazu dient das Graphikmodul MBGRAP.COM.
Dieses wird in den Speicher geladen, aber nicht sofort ausgeführt. Dies erreicht man durch das Kommando

MBGRAP, <---

Durch den Aufruf

DO MBGRAP <---

wird automatisch das Graphikpaket und MBASIC geladen.

Die Graphikfunktionen sind dann über eine Sprungtabelle am Anfang dieses Moduls ansprechbar.
Die Adressierung der Sprungtabelle ist aus dem Proramm 'DEMO.BAS' auf der MBASIC Diskette zu ersehen.

Das Graphikmodul MBGRAP kann durch das Kommando

DO MBGLINK adr

auf eine bestimmte Adresse gelinkt werden. Die Voreinstellung ist B000H; wird das Programm auf eine andere Adresse gelinkt, muß dies im BASIC Programm entsprechend berücksichtigt werden (siehe DEMO). Außerdem sind dazu die Module 'GRAPHB', 'GRAPHV' und 'GRAPHA' von der UTILITY Diskette auf die BASIC Diskette zu kopieren.

Die Grafiktask GRAPTASK muß bei KOS 5.x ebenfalls geladen sein.
RLOAD GRAPTASK <---

2.3.4 Grafik in PSI/PASCAL-Programmen

Die Anwendung der Grafiksoftware in PSI/PASCAL ist im PSI/PASCAL-Handbuch beschrieben.

3. Ein-/Ausgabe-Treiber

Die seriellen Treiber sprechen entweder die Serienschnittstelle A (PSIA, SIOA, OKI) oder die Serienschnittstelle B (PSIB, SIOB) an. Die Programme liegen sowohl als Quellprogramm (Typ '.SRC') als auch als relokative übersetzte Programme (Typ '.OBJ') vor.

3.1 Drucker-Treiber (OKI-MICROLINE 80)

Dateien:

```
OKI.SRC
OKI.OBJ
```

Durch die Kommandodateien

MAKEDEU.KMD	auf deutsche Meldungen schalten
MAKEENG.KMD	auf englische Meldungen schalten
MAKESER.KMD	für seriellen Betrieb einstellen
MAKEPAR.KMD	für parallen Betrieb einstellen

kann der Treiber auf die gewünschte Konfiguration durch automatische Editierung zugeschnitten werden.

Standardeinstellung ist 1200 Baud für den seriellen Betrieb; änderbar im Quellprogramm oder durch INISER P nach dem Aktivieren des Treibers.

Das Kommando

```
DO MAKEOBJ<---
```

bringt die Kommandodatei MAKEOBJ.KMD auf der Utility-Diskette zur Ausführung, die den modifizierten Quelltext assembliert und einen durch

```
EAK $OKI=AKTIV<---
```

aktivierbaren spoolfähigen Treiber erzeugt.

3.2 Allgemeine serielle Treiber

Dateien:

PSIA.SRC	SIOA.SRC
PSIA.OBJ	SIOA.OBJ
PSIB.SRC	SIOB.SRC
PSIB.OBJ	SIOB.OBJ

PSIA/PSIB

Der Treiber dient zur Datenübertragung über die seriellen Schnittstellen A und B; insbesondere ist er dafür ausgelegt, zwei PSI80-Systeme oder ein PSI80-System und einen anderen Rechner zu koppeln.

Beide PSI80-Systeme aktiviert man mit dem Kommando 'EAK $PSIA=AKTIV'.

Zuerst muß der empfangende PSI80 mit dem Kommando

COPY $PSIA datei.typ<---

gestartet werden. Erst dann darf der sendende PSI80 seine Daten übertragen; das Kommando lautet:

COPY datei.typ $PSIA<---

Die Übertragung erfolgt vollduplex. Ist der empfangende PSI80 nicht bereit (da er z.B. gerade einen Teil der Daten auf Floppy abspeichert), überträgt er an den sendenden PSI80 das Zeichen 19H (CTRL-Y). Der sendende PSI wartet so lange, bis er das Zeichen 16H (CTRL-V) bekommt, und fährt dann mit der Übertragung fort.
Bei Dateiende überträgt der sendende PSI80 die Zeichenfolge 04H,04H (CTRL-D,CTRL-D); dies interpretiert der Treiber auf der Empfängerseite als EOF (End of file) und schließt die Datei. Muß der sendende PSI80 das Zeichen 04H übertragen (was nur bei Dateien des Typs '.COM' vorkommen kann, nicht aber bei ASCII-Dateien), sendet er die Zeichenfolge 04H,FFH. Der empfangende PSI entfernt dann das Zeichen FFH, und trägt in die Datei nur das Zeichen 04H ein.

Bei der Kopplung mit anderen Rechnern muß dessen Serienkanaltreiber diese Software-Synchronisation nachbilden, oder es muß der Treiber PSIA/PSIB angepaßt werden.

Die Treiber arbeiten mit:

9600 Baud	kein Parity
2 Stopbits	CTS nicht abfragen

Diese Parameter können mit dem Kommando 'INISER P' geändert werden.

Belegung des Kabels:

PSI80-1		PSI80-2
2	------	3
3	------	2
7	------	7

SIOA/SIOB

Allgemeiner serieller Treiber für die seriellen Schnittstellen A und B. Der bidirektionale Treiber überträgt alle Zeichen und ist spoolfähig (siehe 'SPOOL'-Kommando).

Der Treiber arbeitet mit:

1200 Baud
2 Stopbits
kein Parity
CTS nicht abfragen

Diese Parameter können mit dem Befehl 'INISER P' geändert werden.

Ausführlicher wird dieser Treiber im Kapitel 3.4 'Ein-/Ausgabetreiber-Aufbau' betrachtet.

3.3 Allgemeiner paralleler Treiber

Dateien:
PIO.SRC
PIO.OBJ

PIO

Allgemeiner paralleler Ausgabe-Treiber für die parallele Schnittstelle (Centronics). Der Treiber überträgt alle Zeichen; er ist spoolfähig (siehe 'SPOOL'-Kommando).

3.4 Ein-/Ausgabetreiber-Aufbau

Ein-/Ausgabetreiber sind in der Regel Programe zur Ansteuerung einer spezifischen Hardwareschnittstelle, an der ein Gerät angeschlossen ist. Unter KOS können jedoch auch andere Funktionen als Treiber realisiert werden und so bei Bedarf durch Aktivierung (EAK-Kommando) zum residenten Teil des Betriebssystems dazugebunden werden. Ein Beispiel ist der Umsetzmodul für CP/M-Aufrufe $CPM, siehe Utility-Beschreibung.

KOS zeichnet sich dadurch aus, daß es bis zu 20 verschiedene E/A-Treiber verwalten kann. Jedem E/A-Treiber wird hierbei eine logische Kanalnummer zugeordnet, über die dieser von der E/A-Verwaltung aus adressierbar ist. Diese Zuordnung ist beliebig; die E/A-Verwaltung ist transparent. Deswegen können grundsätzlich alle Ein-/Ausgaben auf beliebige Kanäle gelenkt werden, ohne daß Änderungen den Anwendungsprogrammen notwendig werden.

E/A-Treiber residieren für gewöhnlich über längere Zeit im Arbeitsspeicher des Computers, beanspruchen also Speicherplatz, der damit anderen Programmen nicht mehr zur Verfügung stehen kann. Um für Programme einen möglichst großen zusammenhängenden Speicherbereich bereitzustellen ist es zweckmäßig, E/A-Treiber möglichst weit nach oben zu laden.

Besteht der Treiber nur aus einem Modul und enthält weder Externals noch Globals, übernimmt diese Aufgabe das EAK-Kommando; anderenfalls muß dies mit dem 'LINK'- und 'S'-Kommando geschehen.

Folgendes allgemeine Format eines E/A-Treiberprogramms ist erforderlich:

```
        JP INOUT
        JP STATUS
        JP INIT
        JP OPEN
        JP CLOSE
INOUT:
        .
        .
        .
```

Erläuterung der aufgeführten Unterprogramme

INOUT
Zuständig für den eigentlichen Datentransfer. Dieser erfolgt byteweise, wobei das zu übertragende Byte im Z80A-Akkumulator übergeben wird. Die Routine 'INOUT' ist nur über folgende E/A-Funktionen zu erreichen:

84H: INPUT	- für Eingaben
86H: OUTPUT	- für Ausgaben
81H: OSTATUS	- Status des Ausgabekanals
82H: ISTATUS	- Status des Eingabekanals
8CH: IOCINP	- INIT, OPEN, CLOSE für Input
8DH: IOCOUTP	- INIT, OPEN, CLOSE für Output

Wird eine Datei nicht mit dem 'COPY'-Kommando, sondern mit dem "Spooler" im Hintergrund ausgedruckt, müssen im Treiber zusätzlich die Funktionen 'OSTATUS', 'ISTATUS', 'IOCINP' und 'IOCOUTP' implementiert sein.

Das 'SPOOL'-Kommando kann die Routine STATUS, INIT, OPEN und CLOSE nur über die INOUT-Routine erreichen. Daher muß in dieser Routine die Möglichkeit geschaffen sein, diese 4 Routinen über Funktionsnummern anzusprechen.

Die Routinen OSTAT und ISTAT im folgenden Beispiel (SIOA) überprüfen, ob Zeichen empfangen bzw. gesendet werden können. Ist dies der Fall, erfolgt die Übertragung sofort; ist der Treiber noch mit dem vorherigen Zeichen beschäftigt und daher nicht bereit, wird dies dem Spooler mit dem Z-Flag mitgeteilt (siehe Betriebssystem-Beschreibung).

Über die Routine IOC werden die Unterprogramme INIT, OPEN und CLOSE angesprochen.

Andere Funktionen sind mit dem Fehlercode 81H (unerlaubte Funktion) zu beantworten. Unidirektionale Treiber antworten bei einem Richtungskonflikt mit dem Fehlercode 87H (unerlaubte Datenrichtung).

Beispiel einer INOUT-Routine (ohne Spooling):

```
INOUT:
        LD D,A                  ; Akku retten
        LD A,(IX+1)             ; Funktionsnummer in Akku laden
        CP 84H                  ; Funktion 84H?
        JR Z,INPUT              ; dann Byteeingabe
        CP 86H                  ; Funktion 86H
        JR Z,OUTPUT             ; dann Byteausgabe
        LD (IX+5),81H           ; Fehlercode 81H: unerlaubte Funktio
        RET
INPUT:
        LD A,D                  ; (**)
        CALL GIBEIN             ; Byteeingabe (*)
        RET
OUTPUT:
        LD A,D                  ; (**)
        CALL GIBAUS             ; Byteausgabe (*)
        RET
```

* Bei unidirektionalen Treibern müßte INPUT oder OUTPUT folgenden Befehl enthalten:

```
    LD (IX+5),87H   ; Fehlercode 87H: unerlaubte Datenrichtung
```

** In den Routinen 'INPUT' bzw. 'OUTPUT' sollten alle benötigten CPU-Register in den Stack gerettet werden.

STATUS
Nur für Eingabekanäle von Bedeutung: liefert ein Resultat entsprechend der E/A - Funktion 82H

ein Byte steht bereit ---> Z-Flag = 0
kein Byte stehtbereit ---> Z-Flag = 1

Hinweis:
Wird dem Kanal E-1 (standardmäßig: $KEY) ein anderer E/A-Treiber zugeordnet, so wird automatisch auch das Ziel der Statusabfrage entsprechend verändert. Bei 'Nur-Ausgabekanälen' besteht die Routine 'STATUS' lediglich aus einem RET-Statement. Die 'STATUS-Routine' darf außer den Flags keine CPU-Register verändern!

INIT
Initialisierung des E/A-Treibers: besteht im allgemeinen Fall aus zwei Teilen, nämlich:

Initialisierung der E/A-Treibersoftware
Initialisierung der E/A-Treiberhardware

Die Routine 'INIT' wird bei der Aktivierung eines Kanals durch das EAK-Kommando aufgerufen und muß folgenden Binärwert zur Identifikation im ACCU zurückliefern:

'0' bei Ausgabetreibern
'1' bei Eingabetreibern
'2' bei bidirektionalen Treibern
'3' bei Medientreibern

OPEN
Kennzeichnet den Beginn eines Datentransfers und wird vom COPY-Kommando aufgerufen. In 'OPEN' können gerätespezifische Aktivitäten ausgeführt werden (z.B.: Seitenvorschub bei Druckertreibern).

CLOSE
Kennzeichnet das Ende eines Datentransfers und wird ebenfalls vom COPY-Kommando aufgerufen. Auch hier ist es Sache des E/A-Treibers, welche Aktivitäten in der 'CLOSE'-Routine ausgeführt werden sollen.

Beispiel eines einfachen Ausgabetreiberprogramms (SIOA)

Dieses Treiberprogramm benützt die Serienschnittstelle A des PSI80-Computers und arbeitet mit 1200 Baud. Mit diesem Treiber kann ein beliebiges seriell arbeitendes Datenendgerät bedient werden. Dieses Beispiel zeigt auch, wie E/A-Treiber, falls erforderlich, gelinkt, auf Diskette abgespeichert und aktiviert werden können.

Bei den Routinen 'INIT', 'OPEN' und 'CLOSE' dürfen alle CPU-Register verwendet werden.

```
SIOA:
       JP      INOUT
       JP      STATUS
       JP      INIT
       JP      OPEN
       JP      CLOSE


INOUT:
       LD      B,A
       LD      A,(IX+1)        ;Funktionsnummer prüfen
       CP      86H             ;OUTPUT?
       JP      Z,OUTPUT
       CP      84H             ;INPUT?
       JP      Z,INPUT
       CP      81H             ;OUT STATUS?
       JP      Z,OSTAT
       CP      82H             ;IN STATUS?
       JP      Z,ISTAT
       CP      8CH             ;INIT,OPEN,CLOSE INPUT ?
       JP      Z,IOC
       CP      8DH             ;INIT,OPEN,CLOSE OUTPUT ?
       JP      Z,IOC
       LD      (IX+5),81H      ;Fehlercode 81H: unerlaubte Funktion
       JP      IOEND           ; INOUT beenden

OUPUT:  IN      A,(SIOCHA+2)    ;Transmit buffer leer?
        BIT     TXEM,A
        JR      Z,OUTPUT        ; nein
        LD      A,B
        OUT     (SIOCHA),A
        JP      IOEND

INPUT:  IN      A,(SIOCHA+2)    ;Rx character verfügbar?
       BIT     RXAV,A
       JR      Z,INPUT         ; nein
       IN      A,(SIOCHA)      ; ja
       LD      (IY+7),A        ; character in 'IY-Stack'
       JP      IOEND
```

```
OSTAT:  IN      A,(SIOCHA+2)    ;Treiber busy?
        BIT     TXEM,A
        JR      NZ,OSTAT1       ;nein
        LD      (IX+5),42H      ; nein, Funktion busy
        LD      (IY+6),0FFH     ;Z-Flag = 1
        JP      IOEND
OSTAT1: LD      (IY+6),0        ;Z-Flag = 0
        LD      A,B             ;A=0?
        AND     A
        JP      Z,IOEND         ; ja, nur Status melden
        LD      A,L             ; nein,Zeichen auf SIO ausgeben
        OUT     (SIOCHA),A
        JP      IOEND

ISTAT:  LD      (IY+6),0        ;Z-Flag = 0
        CALL    STATUS          ;bereit zum Empfangen eines Zeichens?
        PUSH    BC
        PUSH    AF
        POP     BC
        LD      (IY+6),C        ;FLAG
        POP     BC
        JP      NZ,IOEND        ; ja,ein Zeichen liegt vor
        LD      (IY+6),0FFH     ; nein, Z-Flag = 1
        JP      IOEND

IOC:    LD      A,B
        CP      0               ;INIT?
        JR      NZ,MB_OP        ; nein
        CALL    INIT            ; ja
        LD      (IY+7),A
        JP      IOEND
MB_OP:  CP      1               ;OPEN?
        JR      NZ,MB_CL        ; nein
        CALL    OPEN            ; ja
        JP      IOEND
MB_CL:  CP      2               ;CLOSE?
        JR      NZ,E81          ; nein, error 81
        CALL    CLOSE           ; ja
        JP      IOEND
E81:    LD      (IX+5),81H      ;ERROR 81
        JP      IOEND
IOEND:  RET                     ;end INOUT
```

```
STATUS:
        IN      A,(SIOCHA+2)    ;bereit zum empfangen?
        BIT     RXAV,A
        RET

INIT:                           ;1200Bd,no parity,2 Stopbit
                                ;8bit/chr,CTS abfragen
        LD      A,CTCMODE       ;INIT CTC
        OUT     (CTC1C2),A
        LD      A,TIMEC
        OUT     (CTC1C2),A
        LD      C,SIOCHA+2      ;INIT SIO
        LD      B,SIOLEN
        LD      HL,SIOTAB
        OTIR
        LD      IX,WRVECT       ;INIT-MESS.
        LD      HL,INIMSG
        RST     8
        LD      A,DIR           ;Rückmeldung
        RET

SIOTAB: DEFB    04H             ;Zeiger auf Write Register 4
        DEFB    4CH             ;CKMODE,2 STPBIT, no parity
        DEFB    05H
        DEFB    0E8H            ;Tx 8bit/chr, Tx enable
        DEFB    03H
        DEFB    0C1H            ;Rx 8bit/chr, Rx enable ,CTS abfragen
        DEFB    01H
        DEFB    00H             ;no interrupt
SIOLEN  EQU     $-SIOTAB

OPEN:
        IN      A,(SIOCHA+2)    ;buffer leer? (Rx char available?)
        BIT     RXAV,A
        JR      Z,OPEND         ; ja
        IN      A,(SIOCHA)      ; nein, clear INPUT buffer
        JR      OPEN
OPEND:  RET

CLOSE:
        RET

INIMSG:
        DEFW    CRLF
        DEFM    " $SIOA - aktiv  (1200 Baud)"
        DEFW    CRLF
        DEFB    00H
WRVECT: DEFB    0,87H,0,0,0,0,0,0
```

```
        ;EQUATES

SIOCHA EQU      04H
CTC1C2 EQU      0AH
TIMEC  EQU      104
CTCMODE EQU     47H
RXAV    EQU      0              ;RX character available (SIO read reg. 0)
TXEM    EQU      2              ;TX-buffer empty (SIO read reg. 0)
STRING  EQU87H                 ;KOSCAL
CRLF    EQU 0A0DH
ASCICR EQU   0DH
ASCILF EQU   0AH
BSPACE EQU   08H
ASCIFF EQU   0CH
ASCTAB EQU   09H
DIR     EQU    2               ;Ausgabe-Treiber:          DIR = 0
                               ;Eingabe-Treiber:          DIR = 1
                               ;bidirektionale Treiber:   DIR = 2
        END     SIOA
```

Assemblieren und Linken eines Treiberprogramms

Nach dem Editiervorgang wird das Programm assembliert durch das Kommando "ASM =SIOA/L<---".

Besteht der Treiber nur aus einem einzigen Modul ohne Externals und Globals (wie in diesem Beispiel), muß er nicht auf eine bestimmte Adresse gelinkt werden, sondern das EAK-Kommando linkt den Treiber automatisch auf den freien Speicherbereich, der direkt unterhalb des Betriebssystems noch frei ist.

Bei Treibern, die aus mehreren Moduln bestehen, muß mit dem 'LINK'- und 'S'-Kommando gearbeitet werden; die einzelnen Schritte werden an Hand des Treibers SIOA erklärt. Existieren von einem Treiber beide Typen (OBJ und EAT), so verwendet das 'EAK'-Kommando die Datei mit dem Typ OBJ.

Das Linken erfolgt mit dem Kommando "LINK SIOA/P:A000/E<---".

Da der Linker nur ab Adresse 100H abspeichert, muß der Treiber SIOA.EAT "per Hand" mit dem KOS-internen Kommando S auf Diskette abgespeichert werden:

```
S 4 SIOA.EAT A000<---
```

Er kann anschließend mit dem EAK-Kommando aktiviert werden:

```
EAK $SIOA=AKTIV<---
```

EAK ruft dabei die Routine 'INIT' des E/A-Treiberprogramms auf und schützt den Speicher ab Adresse A000H (siehe MAP-Kommando). $SIOA ist nun aktiviert, hat allerdings noch keine Kanalnummer zugewiesen. Dies ist möglich mit dem Kommando "EAK A-n=$SIOA<---".

Ist hierbei z.B. n = 1, so gelangen ab sofort alle KOS-Ausgaben auf die Serienschnittstelle.

Beim Datentransfer mittels des COPY-Kommandos werden folgende Aktivitäten ausgeführt:

1. Aufruf der OPEN - Routine
2. Datenübertragung
3. Aufruf der CLOSE - Routine

Hinweis:
Das COPY-Kommando ordnet einem E/A-Treiber automatisch die Kanalnummer zu. Eine vorherige Zuweisung mit dem EAK-Kommando ist deshalb nicht erforderlich.

Soll die Datei mit dem SPOOL-Kommando ausgedruck werden, wird zuerst die Drucker-Task aktiviert (s. PTASK-Kommando, Systemkommandos) durch das Kommando
"RLOAD PTASK<---".

Anschließend wird der Spool-Auftrag mit dem Kommando erteilt:
"SPOOL KOS.INF $SIOA<---".

Ist der Spooler gerade mit dem Ausdruck einer anderen Datei beschäftigt, wird der Druckauftrag für KOS.INF in eine Warteschlange eingereiht. Andernfalls erfolgt sofort der Ausdruck in der Hintergrundverarbeitung.

3.5 Virtueller Medientreiber ($VMED)

Der Medien-Treiber $VMED verwendet einen definierbaren Teil des Systemspeichers als virtuelles Medium. Dieses Medium ist bezüglich seines Verhaltens vollkommen identisch zu Medien im herkömmlichen Sinne des Wortes (Floppy Disk etc.).

Die Kapazität des Mediums $VMED ist in Schritten von 4 kByte von 8 k bis 32 kByte einstellbar. $VMED bietet extrem kurze Zugriffszeiten und ist deshalb immer dann besonders zu empfehlen, wenn Programme oder Daten häufig benötigt werden.

Beispiel für die Aktivierung von $VMED:

```
EAK $VMED=AKTIV M-2=$VMED LIST<---
N 2<---
M2
```

Dieses Beispiel aktiviert den Treiber $VMED und ordnet diesem die Mediennummer 2 zu. Selbstverständlich belegt $VMED Speicherplatz (siehe MAP-Kommandos). Je nach Anwendungsfall sollte dem Medium $VMED deshalb so wenig Speicher wie möglich zugewiesen werden. Die Größe des zugewiesenen Speichers ist bei der Initialisierung im EAK-Kommando oder durch "N $VMED" definierbar. Durch das Kommando "N $VMED" werden alle Dateien des Mediums $VMED gelöscht. Zu beachten ist außerdem, daß $VMED nur für temporäre Dateien geeignet ist. Jeder Reset beendet zwangsläufig die Existenz des Mediums und damit der dort gespeicherten Dateien.

4. Umsetztreiber

4.1 Umsetztreiber für CP/M-Aufrufe

CP/M-kompatible (CP/M-Versionen 1.4 und 2.0) Programme verwenden als Systemaufruf die Instruktion RST5. Über das Umsetzprogramm **CPM.OBJ** werden diese Funktionen in KOS-kompatible Aufrufe (RST8) transformiert.

Dieses Programm ist als Treiber organisiert. Durch

```
EAK $CPM=AKTIV<---
```

wird der Umsetzer zum Betriebssystem dazugeladen.

$CPM ist notwendig für FORTRAN, MBASIC, BASCOM, alle mit diesen Übersetzern erzeugten Programme und für AUTOTEXT und WORDSTAR.

Bei fehlender Aktivierung erscheint die Meldung

```
$CPM aktiv?
```

und das Programm wird abgebrochen.

Ablauf von CP/M* Programmen auf KOS

Ende des verfügbaren Speicherbereiches. Durch die Möglichkeit von Multitasking, Interrupts, und die interne System Struktur erfordert KOS mehr Stack Speicher als ein CP/M System. Da für den Programm Stack mindestens 128 Byte reserviert sind, konnten bisher einige wenige CP/M Programme unter KOS nicht benutzt werden. Dieses Problem existiert mit dem CP/M Translatormodul 2.1/1 nicht mehr. Stack overflows werden unter normalen Situationen verhindert.

Dateizugriff machen (z.B.: Spooling etc.), und parallel zu einem CP/M Programm laufen, das nur einige Bytes für seinen eigenen Stack belegt hat.
Bei MICROSOFT's BASIC COBOL u. FORTRAN Compilern darf bei Hintergrundverarbeitung die Zahl der gleichzeitig geöffneten Dateien 5 nicht übersteigen.

CP/M KOS Translator $CPMX

$CPMX aus geschlossen. Dies ist sinnvoll, wenn Compiler benutzt werden, die eine Reihe von Overlay Dateien erzeugen und so mehr als 8 Dateien gleichzeitig geöffnet wären. Für Programme mit 'random file I/O' ist der Einsatz von $CPMX jedoch nicht garantiert. Hier ist eine andere Methode sinnvoll um bis zu 16 Dateien gleichzeitig offen zuhalten: Mit dem KOS internem A-Kommando generiert man ein 'Loch' von mindestens 8 Segmenten im Speicher. Das garantiert genug Platz für bis zu 16 gleichzeitig geöffneten Dateien. Um $CPM von $CPMX unterscheiden zu können meldet sich $CPMX mit 2.1/2.

4.2 Umsetztreiber für KOS 3.2-formatierte Disketten

Der Übergang von KOS3.2-formatierten Disketten auf KOS 4/5-Format wird durch den Treiber **D32.OBJ** geleistet.

Bei seiner Aktivierung durch

```
EAK $D32=AKTIV M-1=$D32<---
```

fragt der Treiber nochmals nach dem Laufwerk, das die KOS3.2-kompatible Diskette aufnehmen soll, i.a. ist dies Laufwerk 1. Durch MOVE können dann die gewünschten Dateien auf eine KOS 4/5-kompatible Diskette transferiert werden:

```
MOVE,<---          (Diskettenwechsel in Laufwerk 0)
N $D32<---
X 1:* J<---
```

Hinweise:
- Programme sind neu zu übersetzen
- $D32 kann nur lesen
- keine KOS3.2-Systemprogramme auf KOS 4/5-Disketten bringen!

Die Deaktivierung von $D32 und die Wiederzuweisung des KOS 4/5-Disktreibers $DSK1 erfolgt durch

```
EAK $D32=DEAKTIV M-1=$DSK1<---
```

4.3 Umsetztreiber für Single- auf Double-Density-Format

PSI80D-Systeme können PSI80-Disketten lesen und schreiben, wenn auf das entsprechende Laufwerk der Treiber **MIC.OBJ** aktiviert wird:

```
EAK $MIC=AKTIV M-1=$MIC<---
```

Deaktivierung durch: EAK $MIC=DEAKTIV M-1=$DSK1.

Wenn KOS3.2-kompatible Programme auf PSI80D-Systeme übernommen werden sollen,lautet der Aufruf :

```
EAK $D32=AKTIV M-1=$D32 <---
MOVE,<---       (Diskettenwechsel in Laufwerk 0)
N $D32<---
X 1:* J<---
```

Die Rückkehr zu einem reinen Double-Density-System erfolgt durch:

```
EAK  $D32=DEAKTIV M-1=$DSK1<---
```

5. Beispielprogramme auf der Utility-Diskette

5.1 INFO-Kommando (Ausgabe von ASCII-Zeichen)

Dateien:

INFO.SRC
INFOMSG.SRC
INFO.OBJ
INFOMSG.OBJ
MAKE.GER
MAKE.ENG

INFO ist ein Beispiel eines Assembler-Programms:

- Kopf und Ende eines Assembler-Programms
- Schnittstelle zu KOS (Systemaufrufe)
  Datei eröffnen, Record lesen
  Zeichenausgabe auf Bildschirm usw.

INFOMSG ist die Text-Datei für INFO. Alle Meldungen des Kommandos INFO sind in dieser separaten Datei abgespeichert, um Änderungen, z.B. für Fremdsprachen, zu erleichtern.

Mit den Kommandodateien **MAKE.GER** und **MAKE.ENG** kann das Programm INFO in eine deutsche oder englische Version umgestellt werden. Dieses Beispiel demonstriert die Möglichkeiten des Betriebssystems, eine Datei automatisch zu editieren, Änderungen vorzunehmen und wieder abzuschließen.

5.2 BASIC-Programme

Dateien:

MAGIC.BAS
MONDLAND.BAS
HURKLE.BAS
PIANO.BAS (nur bei KOS 5.x)
TONLEIT.DAT (nur bei KOS 5.x)

Dies sind Beispiele für PSI/BASIC-Programme. Insbesondere demonstriert **MAGIC** die leistungsfähige Grafik in PSI/BASIC und **PIANO** die akustische Ausgabe über den Lautsprecher (nur KOS 5.x).

Aufrufe:

```
DO BASIC "LOAD MAGIC<RUN"<---
BASIC "LOAD MONDLAND<RUN"<---
BASIC "LOAD HURKLE<RUN"<---
BASIC "LOAD PIANO<RUN"<---
```

5.3 Hintergrund-Programm

Datei: TIME.SRC

Beispiel für ein Programm, das als TASK im Hintergrund abläuft (siehe auch Systemkommando 'TASK').

Das Programm TIME zählt in der Task 'CLOCK' die im 20ms-Rythmus abgelaufenden Intervalle und bringt jede Sekunde die Anzeige der Uhrzeit auf den Bildschirm. Dazu dient die Task 'DISPLY'.

TIME aktiviert diese beiden Tasks, die von da an im Hintergrund, gesteuert durch Interrupt, ablaufen.

6. Testprogramme

6.1 Test der Peripherie-Bausteine

Aufruf:

PSITEST<---

Dieses Testprogramm überprüft die wichtigsten Funktionen der hochintegrierten Peripherie-Bausteine. Bausteintyp, Betriebsart, Portadresse und Fehlerart werden in einer übersichtlichen Tabelle angezeigt. Eingeschriebenes und ausgelesenes Datenwort wird nur im Fehlerfall angezeigt.

Die Frage 'Testloop im Fehlerfall' ist immer mit 'N' zu beantworten (nur für Service bestimmt).

Achtung:
Der PIO-Baustein kann nur erfolgreich geprüft werden, wenn er auch vorhanden ist (nur PSI80M-Serie)!

Eine Fehlermeldung des PIO kann durch eine externe Beschaltung bedingt sein.

Warnung:
PSITEST überschreibt den letzten Sektor der letzten Spur (Sektor 16, Spur 76) auf Laufwerk 0.

6.2 Speichertest

Aufruf:

MEM64<---

Hier handelt es sich um Testprogramme für den Arbeitsspeicher des PSI80-Computers. Nach dem Aufruf laufen sie in einer endlosen Schleife und zählen jeden Schleifendurchlauf. Eventuell aufgetretene Fehler werden angezeigt.

Abbruch des Programms nur durch ESC (Escape).

6.3 Laufwerk/Disketten-Test

Aufruf:

DISKTEST<---

Dieses Programm stellt bei Verwendung einwandfreier Disketten die Funktion des Laufwerks sicher. Treten mit einer bestimmten Diskette in einem einwandfreien Laufwerk Fehler auf, so handelt es sich um Bit-Fehler auf der Diskette.(siehe auch UTILITY.INF).

KDM – KONTRON DEBUGGING MODULE

Version 5.3

1. August 1981

KDM - Kontron Debugging Module - ist ein wichtiges Hilfsmittel zum Test von Programmen in Z80-Assembler-Sprache. Es ermöglicht den protokollierenden Ablauf von Programmen bis herunter zur Mikroprozessorebene. Das zu testende Programm braucht dazu nicht verändert zu werden, es wird in seiner ablauffähigen Form als '.COM'-Datei zusammen mit KDM geladen und unter KDM gestartet.

Inhaltsverzeichnis

1. Aufruf von KDM

Die PSI80/D-Serie ist mit einem automatischen promresidenten Urlader ausgerüstet, der das Betriebssystem wahlweise von einem der Floppy Disk Laufwerke lädt. Ein promresidentes 'Basis-Operating-System (BOS)', wie im technischen Handbuch 1980 (Kapitel: KDM 3.2) beschrieben, ist nicht mehr implementiert. Zum Austesten von Programmen steht der disk-residente Debug-Monitor KDM zur Verfügung.

2. Bedienung von KDM

```
Aufruf:        KDM (mn:)dateiname(.typ)<---
Voreinst.:     mn  =  Mastermedium
               typ =  COM
Dateispez.:    EDA
```

KDM diese Datei automatisch in den Anwenderspeicherbereich entsprechend der Startadresse dieses Programms. Die Angabe von 'mn:dateiname.typ' ist optional.

Bei der Ausführung eines Programms unter KDM-Kontrolle (Haltepunkte, Einzelschritt etc.) ist darauf zu achten, daß Haltepunkte nur innerhalb des Anwenderprogramms gesetzt werden, nicht aber im Betriebssystembereich. Haltepunkte in Programmen mit graphischer Betriebsart des Sichtschirms sind nicht möglich. Der Anwender muß sicherstellen, daß beim Erkennen eines Haltepunktes der alphanumerische Modus eingeschaltet ist. Programmzähler und Stackpointer der virtuellen Anwender-CPU sind mit zulässigen Werten vorbesetzt (PC=Startadresse; SP innerhalb KDM). Damit kann das Programm mit 'G' gestartet werden. KDM bietet die Möglichkeit, 'Keyboard Breaks' zu generieren. Hierzu muß die Taste CTRL-K gedrückt werden. Auch unendliche Schleifen können hiermit verlassen werden. KDM zeigt nach dem Erkennen eines 'Keyboard Breaks' (wie beim normalen Haltepunkt) den Registersatz der CPU an.

Beispiel:

```
KOS:KDM BASIC<---
```

Lädt die Datei BASIC.COM unter KDM-Kontrolle. BASIC kann anschließend durch das KDM-Kommando 'G' (oder 'J 100') ausgeführt werden.

Bei Angabe eines Dateinamens wird diese Datei zusammen mit KDM geladen und kann dann getestet werden. KDM selbst liegt im Speicherbereich ab Adresse 8000H. Das Programm kann mit der Kommandodatei KDMLINK.KMD (siehe Utility Diskette) auf eine andere Adresse gelinkt werden, z.B. 'DO KDMLINK 3000<---'.

E/A-Kanäle:

```
Eingabe   --->  Kanal E-1
Status    --->  Kanal E-1
Ausgaben  --->  Kanal A-1
```

2.1 Aufbau eines Kommandos

Jede Kommandoeingabe (jedes Kommandofeld) besteht aus einem Identifikations- und einem Parameterfeld (ID-/P-Feld), sowie dazwischenliegenden Trennzeichen.

```
<---ID-Feld--->
KOMMANDOAUFRUF TZ P1 TZ P2 TZ...Pi TZ...Pn <CR>
<--------Parameterfeld-------->
```

Das ID-Feld muß mit einem Großbuchstaben (A bis Z) beginnen und reicht bis zum ersten Trennzeichen (TZ1). Da der Kommandointerpreter in der Regel nur einen, höchstens aber zwei Zeichen zur Identifikation eines Kommandos benötigt, bleiben eventuell vorhandene weitere Zeichen im ID-Feld bedeutungslos.

Als Trennzeichen sind zugelassen:

20H (Leerzeichen)
09H CNTR-I (Tabulation)

Ihre Anzahl zwischen den Parametern spielt keine Rolle. Das Parameterfeld ist für verschiedene Kommandos optional. Es enthält mit Ausnahme des Registerkommandos nur hexadezimale Zahlenwerte (0...9 und A..F). Die Eingabe des Parameterfeldes ist formatfrei, d.h. führende Nullen brauchen nicht mit eingegeben zu werden.

Beispiel: Die Eingaben

```
9
09
009
0009
```

sind gleichbedeutend und werden als vierstellige Hexadezimalzahl 0009 betrachtet.

Bei mehr als vier Zeichen werden lediglich die letzten vier berücksichtigt. Entsprechendes gilt auch dann, wenn KDM nur 2 hexadezimale Zeichen als Eingabe erwartet. Wie bereits erwähnt, können pro (logischer) Zeile mehrere Kommandos hintereinander eingegeben werden. Der Zeilenpuffer ist 256 Bytes groß, das entspricht mehr als drei Zeilen des PSI80-Sichtschirmes.

Zur Trennung zweier Kommandos dient das Semikolon (ASCII-Code 3BH). Der ASCII-Code 0DH (<CR>), Taste RETURN) schließt die Kommandozeile ab.

Es ergibt sich das folgende Format für die Kommandozeile:

```
<--------logische Zeile (256 Bytes max)-------->
KOMMANDO1;KOMMANDO2;...;KOMMANDOi;...;KOMMANDOn<---
```

2.2 Eingabekorrektur

Eine Korrektur der laufenden Eingabe vor Abschluß der Zeile geschieht durch

- die Taste RUBOUT (ASCII-Code 7FH):
  Löschen des gesamten Zeilenpuffers

- die Tastenkombination CNTR-H
  oder die Taste 'Cursor links' (ASCII-Code 08H):
  Löschen des zuletzt eingegebenen Zeichens.

In beiden Fällen wird an den Ausgabetreiber pro zu löschendes Zeichen die Kombination 'Backspace-Blank-Backspace' übertragen.

Bei Löschen von am Bildschirm nicht als 1-stellige Zeichen sichtbaren Codes (TAB, Controlzeichen) stimmt die Darstellung am Bildschirm (Cursorposition) nicht mit dem Abbild des Eingabepuffers überein.

2.3 Eingabefehler

Die Fehlermeldungen zeigen einen der folgenden Zustände an:

- die Kommando-Identifikation ist unbekannt
- das Kommandofeld beginnt mit einem unzulässigen Zeichen
- das Parameterfeld enthält unzulässige Zeichen
- das Parameterfeld enthält nicht die geforderte Anzahl von Parametern.

Die Fehlermeldungen aus KDM sind in Klartext gehalten.

Grundsätzlich führt eine Fehlermeldung zum Abbruch des Kommandos. Weitere Kommandos einer logischen Zeile werden allerdings ordnungsgemäß abgearbeitet.

2.4 Kommandoabbruch

Kommandos, die einen längeren Ausdruck auf dem Sichtschirm bewirken, können jederzeit unterbrochen, wieder fortgesetzt oder abgebrochen werden. Dazu überprüft das Monitorprogramm während der Kommandoausführung periodisch, ob in der Zwischenzeit eine Taste gedrückt wurde.

Für KDM gelten die KOS-Konventionen:

- CNTR-S schaltet die Geschwindigkeit der Anzeige um
- ESC bricht das Kommando ab
- Jede Eingabe stoppt und startet die Kommandoausführung
- CNTR-P schaltet die 'Paging-Funktion' ein

3. Kommandobeschreibung

Für jedes Kommando folgt nach einer kurzen einleitenden Funktionsbeschreibung die Syntax von Identifikations- und Parameterfeld. Für die Taste 'RETURN' steht das Symbol '<---'.

Folgende Kommandos stehen zur Verfügung:

B	KOS Speicherbelegungsplan ausgeben
D adr	Anzeigen und Ändern von Speicherstellen
D adr len	Speicherbereich anzeigen
D adr1-adr2	Speicherbereich anzeigen
DA adr len	Speicherbereich disassemblieren
DA adr1-adr2	Speicherbereich disassemblieren
DH	Haltepunkt anzeigen
DP adr	Port mit Adresse adr anzeigen
DR lrn mn segm adr	Record lrn von Medium mn in Puffer ab adr anzeigen
DW	Window anzeigen
F adr n by	Speicher ab adr mit by füllen (n Stellen)
F adr1-adr2 by	Speicher ab adr1 bis adr2 mit by füllen
G (adr)	Anwenderprogramm ab adr ausführen
J adr	nach adr springen und Programm starten
K	Rückkehr zu KOS
L adr n data data..	ab adr bis adr+n Folge data.. lokalisieren
L adr1-adr2 data...	ab adr1 bis adr2 Folge data.. lokalisieren
N n	Einzelschritt im Anwenderprogramm (n mal)
SH adr	Haltepunkt auf adr setzen
SP adr by	Port adr auf Wert by setzen
SR lrn mn segm adr	Record lrn auf Medium mn mit Daten aus Puffer ab adr beschreiben
SW adr len	Window (Speicheranz. n. break) len Byte ab adr
SW adr1-adr2	Window von adr1 bis adr2 setzen
R	Register anzeigen
R rn	Anzeigen und Ändern von Register rn
T adr1 adr2	Übertragen von Bytes von adr1 nach adr2
X	KOS Kommandomanager aufrufen

3.1 Speicherbelegungsplan (Bitmap) der Speicherverwaltung

Format: B<---

Ausdruck des aktuellen Speicherbelegungsplans, ähnlich dem MAP-Kommando in KOS.

3.2 Darstellen von Speicher- und Port-Inhalten

Das Darstelle-Kommando (Display) dient zum Ausdruck von Speicherinhalten, Ports und der Haltepunkt-Adresse. Speicherstellen können sowohl einzeln (interaktiver 'Display and Alter Mode') als auch im Block (mit ASCII-Äquivalent) ausgegeben werden.

a) Darstellen und Ändern einzelner Speicherinhalte

Format: D adr<---

Ausdruck des Inhalts der Speicheradresse adr.

Die Angabe von adr ist optional;
In diesem Modus arbeitet das D-Kommando interaktiv, d.h. nach dem Ausdruck des Speicherinhalts kann der Benutzer wahlweise den Inhalt des angezeigten Speicherplatzes verändern und/oder auf den nächsten/vorhergehenden Speicherplatz weiterschalten. Der Abbruch einer derartigen 'Display and Alter Sequenz' erfolgt mit dem Zeichen 'Q' (Quittierung).

Ausgabeformat: adr BB

Fünf Funktionen können nun durch Eingabe folgender Zeichen (-folgen) veranlaßt werden:

<---	Weiterschalten und Ausdruck des Inhaltes der Speicherstelle adr+1
^<---	Weiterschalten und Ausdruck des Inhaltes der Speicherstelle adr-1
xx<---	Ersetzen des Inhaltes von adr durch den Wert xx mit anschließendem Weiterschalten auf adr+1
xx Q<---	wie oben, mit anschließendem Abbruch des Kommandos
Q<---	Abbruch des Kommandos

b) D-Darstellen von Speicherbereichen

Format: D adr1-adr2<---
D adr1 n<---

mit adr1 Anfangsadresse
adr2 Endadresse
n Anzahl der Speicherstellen

Entspricht die Differenz adr2 - adr1 bzw. die Größe n keinem ganzzahligen Vielfachen von hexadezimal 10, so erfolgt automatisch die Aufrundung auf das nächste ganzzahlige Vielfache von 10H. Im Modus 'Bereichausdruck' sind deshalb minimal 16 Speicherinhalte ab adr1 ausdruckbar, was einer physikalischen Zeile auf dem Sichtschirm des PSI80 entspricht.

Der Ausdruck erfolgt in folgendem Format:

adr1 B0 B1 B2.....BE BF *A0A1.....AF*

Die Größen A0 bis AF kennzeichnen die ASCII-Äquivalente der Bytes B0 bis BF. Nicht abdruckfähige ASCII-Codes erscheinen als Punkt (.).

Nur im Ausgabeformat unterscheidet sich das **Disassembliere-Kommando:**

DA adr1 n bzw. DA adr1-adr2

c) Darstellen des Haltepunktes

Format: DH<---

Ausdruck der Adresse, auf der momentan ein Haltepunkt gesetzt ist. Diese Adresse hat den Wert 0, falls bisher kein Haltepunkt gesetzt war, bzw. ein ehemals gesetzter Haltepunkt gelöscht wurde.

Ausgabeformat: adr

d) Darstellen von Port-Inhalten

Format: D portadr<---

Ausdruck des Inhaltes der Ein-/Ausgabeadresse portadr. Die korrekte Programmierung der Ein-/Ausgabe-Bausteine ist den jeweiligen Handbüchern zu entnehmen.

e) Darstellen von logischen Sätzen (display record)

Format: DR lrn mn segm adr<---

mit:
- lrn logische Satznummer (logical record number)
- mn Mediennummer
- segm Nummer des adressierten 8 MByte-Segments eines Mediums
- adr Pufferadresse ab der der Satz abgelegt wird

Voreinst.:
- lrn = 0
- mn = 0
- segm = 0
- adr wird von KDM dynamisch festgelegt

Ausgabe des Inhalts der logischen Satznummer lrn von Medium mn im Segment segm auf die Adresse adr.
Wird ein Parameter nicht angegeben, so gelten als Voreinstellung die Werte bei der letztmaligen Ausführung des DR-Kommandos. Das DR-Kommando liest den Satz in einen Speicherbereich ein. Dort sind die üblichen Modifikationen möglich; diese Änderungen können mit dem Kommando SR auf die Diskette zurückgeschrieben werden (siehe 3.9).

f) Darstellen des aktuellen Speicherfensters (display Window)

Format: DW<---

Ausdruck von Anfangs- und Endadresse des aktuellen Speicherfensters (siehe 3.9).

3.3 Fülle-Kommando

Mit dem Fülle-Kommando (Fill) können beliebige Speicherbereiche mit einer Konstanten gefüllt werden.

Format: F adr1-adr2 by<---
F adr1 n by<---

mit
- adr1 Anfangsadresse
- adr2 Endadresse
- n Anzahl
- by einzuschreibendes Byte (Hex.)

Vorbesetzung der Parameter ist 0.

Bei diesem Kommando ist darauf zu achten, daß die von KDM und KOS benötigten Speicherbereiche nicht zerstört werden.

3.4 Programmstart 'Gehe' (Goto)

Das G-Kommando ermöglicht den Start eines Anwenderprogramms bei beliebiger Adresse, bzw. die Fortsetzung eines Programms nach einem Haltepunkt. Vorher werden grundsätzlich die Register der Anwender-CPU rückgespeichert.

Format: G adr<---

Ist der Parameter adr spezifiziert, erfolgt ein Sprung auf die Adresse adr, andernfalls wird der momentane Stand des Anwender-PC (Programmzählers) als Startadresse adr verwendet. Soll also beispielsweise ein Programm nach einem Haltepunkt fortgesetzt werden, so genügt das Kommando G ohne Parameterangabe. Die ordnungsgemäße Weiterführung des Programms ist gewährleistet, da das Monitorprogramm zunächst überprüft, ob die Sprungadresse einen Haltepunkt enthält. Trifft dies zu, so wird der Haltepunkt vorübergehend deaktiviert und das Programm anschließend fortgesetzt. Ein Haltepunkt bleibt bestehen, bis er explizit durch das Kommando SH oder Neuinitialisierung gelöscht wird.

3.5 Jump-Kommando

Im Gegensatz zum G-Kommando führt das J-Kommando direkt zum Sprung auf eine angegebene Adresse, ohne daß die Register der Anwender-CPU rückgespeichert werden.

Format: J adr<---

Ist der Wert adr nicht spezifiziert, wird stattdessen 0 eingesetzt.

3.6 KOS-Kommando: Verlassen von KDM

Mit der Eingabe von

KOS<---

wird KDM verlassen.

3.7 Lokalisieren von Bytefolgen

Format: L adr n b1 b2...<---
L adr1-adr2 b1 b2...<---

Durch dieses Kommando können Bytefolgen im Speicher gesucht werden. Dabei kann der zu durchsuchende Speicherbereich und eine beliebig lange Folge (begrenzt durch die Länge einer Eingabezeile = 255 Zeichen) von Bytes in Hexadezimaldarstellung angegeben werden.

Die gefundene Folge wird in Form des D-Kommandos, beginnend 16 Byte vor dem 1. Byte der Folge ausgegeben. Bei erfolgloser Suche erfolgt keine Ausgabe.

3.8 Nächster Programmschritt

Ausführung von 1...255 Einzelschritten mit Ausdruck der CPU-Register nach jedem Schritt.

Format: N n<---

Die n (1 < n < FFH) nächsten Programmschritte ab dem momentanen Stand des Anwender-PC werden ausgeführt; n = 1 falls nicht spezifiziert.

Nach jedem Schritt werden die CPU-Register gerettet und anschließend auf den Sichtschirm ausgegeben (Format und Reihenfolge siehe Registerkommando). Beim N-Kommando wird ein eventuell gesetzter Haltepunkt gelöscht.

Da das N-Kommando interruptgesteuert ist, darf der Z80-Maschinenbefehl 'LD I,A' **nicht als Einzelschritt** ausgeführt werden. Der Befehl DI (Disable Interrupt) wird vom N-Kommando ignoriert.

3.9 Setzen von Ports, Haltepunkten, Sätzen und Speicherbereichen.

Mit diesem Kommando können Ports und Haltepunkte gesetzt, logische Sätze auf ein Medium geschrieben und Speicherbereiche (Window) festgelegt werden.

a) Setze Haltepunkt

Format: SH adr<---

Setzt einen Haltepunkt auf die Adresse adr, falls der Wert adr spezifiziert wurde. Andernfalls wird ein bisher aktiver Haltepunkt deaktiviert.

Das Programm KDM arbeitet mit Software-Haltepunkten. Das Prinzip besteht darin, den Maschinencode der Speicherzelle adr mit dem Code FFH (Restart 38H) zu ersetzen. Der Befehlscode FFH bewirkt einen Restart bei der Adresse 38H, womit schließlich nach der Rettung der CPU-Register ein definierter Rücksprung vom Anwenderprogramm nach KDM erfolgt. Da der Restart-Befehl einem Call-Befehl entspricht, muß der Stapelzeiger der 'Anwender-CPU' auf einen zulässigen Speicherbereich zeigen, um das Überschreiben bereits belegter Speicherzellen zu verhindern. Um dies sicherzustellen, sollte der Anwender dafür sorgen, daß entweder

- der Stapelzeiger im Anwenderprogramm selbst definiert wird oder
- der Stapelzeiger vor dem Start des Anwenderprogramms per Register-Setz-Kommando (R SP) auf den gewünschten Wert gesetzt wird.

Es ist zu beachten, daß Haltepunkte nur dann als solche erkannt werden können, wenn der Parameter adr auf das erste Byte eines Befehls zeigt. Um dies unmittelbar nach der Eingabe des Kommandos SH überprüfen zu können, wird der Speicherbereich adr-8 bis adr+7 vor und nach dem Einsetzen des Haltepunkts mit dem Format des D-Kommandos automatisch ausgedruckt. Läuft ein Programm auf einen Haltepunkt auf, so meldet sich das Monitorprogramm mit dem Ausdruck der Registerinhalte, sowie dem als 'Window' definierten Speicherbereich.

b) Setze Port

Format: SP adr by<---

Das Byte by wird auf den Port mit der Adresse adr übertragen. Sind Parameter nicht spezifiziert, wird dafür der Wert 0 eingesetzt. Nach dem SP-Kommando wird automatisch das DP-Kommando ausgeführt.

c) Setze logischen Satz (set record)

Schreibt einen logischen Satz auf ein aktives Medium.

```
Format:      SR lrn mn segm adr<---
Voreinst.:   lrn  =  0
             mn   =  0
             segm =  0
             adr  =  wird von KDM dynamisch festgelegt

      mit    lrn     logische Satznummer (logical record number)
             mn      Mediennummer
             segm    Nummer des adressierten 8 MByte-Segments
                     eines Mediums
             adr     Pufferadresse aus der der Satz geschrieben
                     werden soll
```

Schreibt den logischen Satz lrn auf Medium mn in das Segment segm ab Adresse adr. In Verbindung mit dem DR-Kommando kann damit gezielt ein Satz auf einem Medium geändert werden. Um unabsichtliche Zerstörungen von Medien zu vermeiden verlangt das SR-Kommando nach dem Ausdruck der Daten eine Bestätigung vom Anwender. Erst dann erfolgt der Schreibvorgang.

d) Setze Speicherfenster (set window)

Format: SW adr1-adr2<---
SW adr1 n<---

Dieses Kommando bestimmt denjenigen Speicherbereich, der nach einem Haltepunkt automatisch im Format des D-Kommandos ausgedruckt wird. Das SW-Kommando eignet sich insbesondere zum Ausdruck von Vektorinhalten bei Systemaufrufen. Im Gegensatz zu früheren KDM-Versionen darf ein Haltepunkt auch auf die Adresse 8 (KOSCAL) gesetzt werden.

Das SW-Kommando kann durch die Eingabe von SW 0<--- rückgängig gemacht werden.

3.10 Register-Kommando

Ausdruck der Registerinhalte der in KDM 'virtuell' vorhandenen Anwender-CPU. Diese stehen in einem für KDM reservierten Speicherbereich. Beim Programmstarten mit dem G- oder N-Kommando (3.4 bzw. 3.8) werden alle CPU-Register auf die Werte der Anwender-CPU gesetzt. Entsprechend werden die Registerinhalte der Anwender-CPU beim Rücksprung in das Debug Module in den dafür vorhandenen Speicherbereich gerettet.

Format: R rn<---

Wird kein Registername rn spezifiziert, erfolgt der Ausdruck des gesamten Registersatzes der CPU in untenstehender Reihenfolge:

A F B C D E H L A' F' B' C' D' E' H' L' I IX IY PC SP

Die Angabe eines Registernamens führt in einen interaktiven Anzeigemodus (Display and Alter Mode), in dem einzelne Registerinhalte angezeigt und modifiziert werden können. Das Ausgabeformat entspricht dem D-Kommando (Abschnitt 3.2).

Der Benutzer hat wieder die Möglichkeit, den angezeigten Registerinhalt zu verändern und/oder auf das nächste Register weiterzuschalten. Es gelten die Konventionen des D-Kommandos. Ein Zurückschalten auf das vorhergehende Register ist allerdings nicht möglich. Die Reihenfolge der Register entspricht der Reihenfolge beim R-Kommando (ohne rn) von links nach rechts. Die Initialisierung von KDM setzt alle Register mit Ausnahme des I-Registers auf den Wert 0. Da das N-Kommando interruptgesteuert ist, darf das I-Register der Anwender-CPU gewöhnlich nicht verändert werden, es sei denn, der Anwender sperrt den Interrupt der CPU (Befehl: DI) und verzichtet auf die interruptgesteuerten Abläufe.

Interruptbasierende Anwenderprogramme sollten die im Schreib-/Lesespeicher liegende Interrupttabelle, wofür 100H Bytes reserviert sind, verwenden.

3.11 Transfer-Kommando

Das Transfer-Kommando verschiebt einen beliebigen Speicherbereich in einen anderen.

```
Format:      T adr1 adr2 n<---
Voreinst:    adr1  = 0
             adr2  = 0
             n     = 0
```

verschiebt n Bytes ab Adresse adr1 in den Speicherbereich ab adr2. Es gilt folgende Zuordnung:

```
adr1      -----> adr2
adr+1     -----> adr2+1
adr1+2    -----> adr2+n
```

Auch bei diesem Kommando ist zu beachten, daß die für den Anwender unerlaubten Speicherbereiche nicht überschrieben werden.

3.12 KOS-Kommandointerpreter aufrufen

Format: X<---

Dieses Kommando ermöglicht den Aufruf des KOS-Kommandointerpreters. Hierzu kann nach dem X-Kommando ein beliebiges KOS-Kommando eingegeben werden. Nach der Ausführung des Kommandos erfolgt die Rückkehr zu KDM.

BETRIEBSSYSTEM

BESCHREIBUNG

Stand: 15. August 1981

Version: Rev. 4/Rev. 5

Diese Beschreibung gilt den Betriebssystemversionen 4.x und 5.x im folgenden als KOS bezeichnet. Bitte beachten Sie auch die Datei KOS.INF Ihrer Systemdiskette. Sie finden dort weitere wichtige Informationen.

Der vorliegende Teil des PSI80-technischen Handbuchs gibt sehr detaillierte Hinweise über das Betriebssystem KOS, die für Anwender nützlich sind, die die volle Leistung dieses Betriebssystems aus ihren Anwenderprogrammen heraus nutzen wollen.

Für alle anderen Anwendungsfälle ist das PSI80-Bedienungshandbuch zusammen mit dem Abschnitt 'KOS-Systemkommandos' in dieser PSI80-Technischen Beschreibung voll ausreichend.

INHALTSVERZEICHNIS

1. Übersicht

Das Betriebssystem KOS (Kontron-Operating-System) ist in allen diskbasierenden PSI80-Versionen implementiert. Es ist als Anwendungs-orientiertes transparantes Betriebssystem konzipiert.

KOS ist medienunabhängig: externe Massenspeicher werden über logische Kanäle mit wahlfreier Zuordnung verwendet. Für die Medienverwaltung gilt:

- Kapazität pro Medium bis 64 MByte
- automatische Dateisuche auf allen aktiven Medien
- Dateigröße von 0 bis 8 MByte
- Variables Inhaltsverzeichnis mit einer nur durch die Kapazität des Mediums begrenzten Anzahl von Dateien.
- Sequentieller und wahlfreier Zugriff auf alle Sätze einer Datei
- benutzerdefinierbare Dateieigenschaften (file properties). wie Benutzerkennzeichen, schreibgeschützt, löschgeschützt, geheim etc.

Eine komfortable Ein-/Ausgabeverwaltung erlaubt die beliebige Zuordnung von logischen Ein-/Ausgabekanälen zu symbolischen Gerätekennzeichnungen. Damit ist der Datentransfer von beliebigen Quellen zu beliebigen Zielen von allen Programmen aus möglich, ohne diese selbst zu ändern.

KOS unterstützt den quasisimultanen Ablauf eines Vordergrundprogramms mit bis zu zehn Hintergrundprogrammen (Tasks).

Über frei aktivierbare Umsetzermodule wird die Aufwärtskompatibilität zu anderen Betriebssystemen erreicht. Implementiert ist z.Zt. das Anpassungsmodul zu CP/M 1.4 und CP/M 2.0.

2. Aufbau von KOS

Das Betriebssystem KOS ist ein Dienstleistungssystem, das Kommandos vom Benutzer oder externen Dateien (Kommandodateien) erhält und diese ausführt. KOS besteht im wesentlichen aus folgenden sieben Teilen:

- dem Systemverwalter
- dem Kommandoentschlüssler
- dem Dateiverwalter
- dem Ein-/Ausgabeverwalter
- dem Speicherverwalter
- dem Taskverwalter
- den internen Systemprogrammen

Der Systemverwalter ist das übergeordnete Softwaremodul des Betriebssystems. Die Hauptfunktion des Systemverwalters ist die Koordination der Abläufe in den übrigen Teilen von KOS.

Der Kommandoentschlüssler unterscheidet KOS-interne von externen (medienresidenten) Systemkommandos und bereitet gleichzeitig den Kommandostring zu einem Parameterblock auf. Dieser wird von anderen Teilen des Betriebssystems weiter verwendet.

Der Dateiverwalter besorgt die Zuordnung zwischen symbolischen und logischen Adressen, z.B. Name <---> Satznummer von Informationsaufzeichnungen auf Medien (externen Datenträgern).

Die wesentliche Funktion der E/A-Verwaltung ist die Verwaltung und Verzweigung von und zu logischen Datenübertragungskanälen. KOS und die Dienstprogramme verwenden definierte logische Kanäle zur Ein-/Ausgabe. Logische Kanäle sind gewöhnlich einem Ein-/Ausgabetreiber zugeordnet. Diese Zuordnung ist beliebig und kann jederzeit geändert werden (siehe EAK-Kommando). Damit ergibt sich die Möglichkeit, die Ein-/Ausgaben aller Programme auf beliebige Kanäle zu dirigieren, ohne die Programme selbst ändern zu müssen.

KOS zeichnet sich unter anderem durch eine leistungsfähige Speicherverwaltung aus. Diese teilt den gesamten verfügbaren Speicher von 64 kByte in Segmente zu je 128 Byte ein. Pro Segment wird in einer Tabelle von 64 Byte ein Bit geführt, das gesetzt ist, falls das dazugehörige Segment belegt ist. Im umgekehrten Fall ist es rückgesetzt. Immer dann, wenn Programme Speicherplatz benötigen, wird die Speicherverwaltung beauftragt, den entsprechenden Speicherbereich zu belegen. Ist dieser bereits belegt, so erfolgt eine Fehlermeldung. Vor jeder Ausführung eines Systemprogramms (wie ASM, EDIT, COPY etc.) reserviert KOS auch einen 100H Byte großen Speicherbereich für den Stack des Programms im obersten noch freien Speicher.

Der Taskverwalter von KOS verwaltet bis zu zehn benutzerdefinierbare Tasks, also Programme, die neben der eigentlichen Benutzertask (Editor, Assembler etc.) im Hintergrund ablaufen.

3. Speicherorganisation von KOS

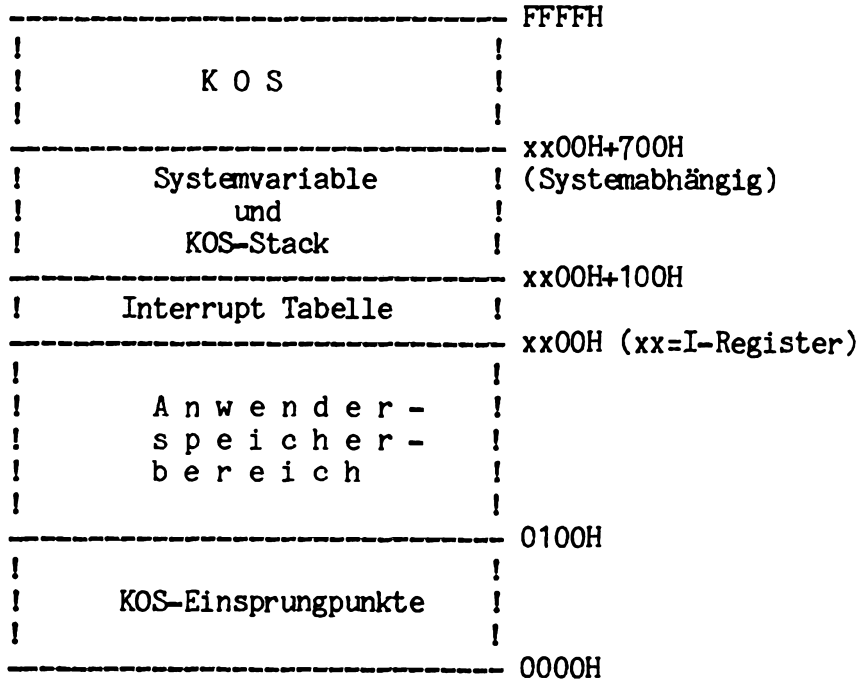
Das Betriebssystem KOS benötigt etwa 12 kByte und residiert grundsätzlich im obersten zur Verfügung stehenden Speicherbereich. Beim Kaltstart von KOS wird folgende Speicherorganisation aufgebaut:

- Der PROM-Bereich der PSI80-Zentralplatine wird abgeschaltet, so daß nur Schreib-/Lesespeicher vorhanden ist.

- Der Adreßbereich 0 - FFH (256 Byte) wird für Systemparameter reserviert und enthält auf der Adresse 8 den Systemeinsprungpunkt KOSCAL; dort erfolgt die Verzweigung zu den einzelnen Funktionen des Betriebssystems aufgrund einer Funktionsnummer 'n'.

- Für das Betriebssystem wird der Bereich von xx00H bis FFFFH reserviert.

KOS-Speicherorganisation



Es wird empfohlen, den Wert xx grundsätzlich aus dem I-Register der CPU zu entnehmen. Dies garantiert Kompatibilität bei Änderungen von xx.

Wichtige Systemadressen: (Hexwerte)

00 - 02	KOS Warmstart
03	KDT-Statusbyte
04	reserviert für CP/M-KOS Translator ($CPM)
05	CP/M-Systemcall Entry Point (RST 10H)
06 - 07	MEMTOP
08 - 0A	KOS-Systemcall Einsprung
0B - 0D	KOS-Fehler Einsprung
0E - 0F	reserviert für Datum
10 - 12	Einsprungpunkt nach $CPM
13	reserviert für $DSK0/$DSK1
14 - 15	reserviert für $MON/$GRAP
16 - 17	reserviert für PSI/PASCAL
18 - 1A	frei für RST 18H Einsprung
1B	reserviert für $GRAP
1C - 1E	reserviert für Uhrzeit (Sec/Min/Std)
1F	20ms Intervall Zähler
20 - 22	frei für RST 20H Einsprung
23 - 27	reserviert für KDM
28 - 2A	frei für RST 28H Einsprung
2B - 2C	reserviert für KDM
2D - 2F	frei
30 - 32	frei für RST 30H Einsprung
33 - 37	frei
38 - 3A	RST 38H Einsprung (KDM Haltepunkt)
3B - 3F	frei
40 - 4F	frei
50 - 5F	DSB 1 (KOS Parameteraufbereitung)
60 - 6F	DSB 2 (KOS Parameteraufbereitung)
70 - 7F	DSB 3 (KOS Parameteraufbereitung)
100	Anwenderspeicherbereich (bis Interrupttabelle)

a) WSTART
Der Warmstart Einsprungpunkt (WSTART) bietet eine einfache Möglichkeit zur Rückkehr von Anwenderprogrammen in das Betriebssystem KOS. Der momentane Stand des CPU-Stackpointers spielt keine Rolle, da dieser in der WSTART-Routine neu definiert wird. KOS führt nach einem 'Warmstart' weitere Kommandos einer Kommandozeile aus oder wartet auf neue Benutzereingaben.

b) KOSCAL
Der Aufruf von beliebigen Funktionen des Betriebssystems durch System- oder Anwenderprogramme erfolgt unabhängig von der Lage des Betriebssystems mit dem Einbyte-Callbefehl RST8.

c) KERROR
Der Fehlereinsprungpunkt von KOS (Adresse: 000BH) kann als Rückkehradresse im Parametervektor eines KOS-Aufrufs angegeben werden (siehe auch allgemeine Systemfunktionen: Funktion 3).

Der Effekt ist, daß dann bei jedem KOS-Aufruf (KOSCAL) im Fehlerfalle zunächst die entsprechende KOS-Standardfehlermeldung ausgegeben und anschließend ein Warmstart durchgeführt wird.

d) MEMTOP
In den Speicherstellen 6 und 7 wird die Adresse des ersten nicht überschreibbaren Speicherplatzes eingetragen (MEMTOP). Anwender- und Dienstprogramme, welche die KOS-Speicherverwaltung nicht verwenden wollen, ersehen daraus den ihnen zur Verfügung stehenden Speicherplatz (100H bis MEMTOP-1). MEMTOP ist ein dynamischer Wert, der vor jeder Programmausführung vom Speicherverwalter des Betriebssystems neu errechnet wird.

4. KOS-Systemfunktionen - Übersicht

Alle Systemfunktionen (Ein-/Ausgaben, Datei-, Task-, Speicherverwaltung) sind über den gemeinsamen Systemeinsprungpunkt auf Adresse 8 (RST 8-Befehl) erreichbar. Dieser Einsprungpunkt ist reentrant, d.h., daß Systemfunktionen sich selbst oder andere Systemfunktionen aufrufen können. Dies bedeutet auch, daß eine Systemfunktion an beliebiger Stelle per Interrupt unterbrechbar ist und, daß dieselbe Systemfunktion während der Abarbeitung einer Interrupt Service Routine mit den gleichen oder aber anderen Eingangsparametern wieder aufgerufen werden kann.

Einige Systemfunktionen dürfen an bestimmten Stellen nicht unterbrochen werden. Falls sie trotzdem unterbrochen und erneut aufgerufen werden, so antworten diese Funktionen mit dem Fehlercode 42H (Funktion ist 'busy').

4.1. ParameterÜbergabe - allgemeine Konvention

Der Aufruf von KOS-Funktionen durch ein Anwender- oder Systemprogramm mittels KOSCAL erfolgt mit dem IX-Register der CPU als Parametervektor. Dieser zeigt auf einen beliebigen Speicherbereich mit folgenden Informationen:

(IX + 0)	-	logische Kanalnummer
(IX + 1)	-	Funktionsnummer
(IX + 2)	-	funktionsabhängig
(IX + 3)	-	funktionsabhängig
(IX + 4)	-	funktionsabhängig
(IX + 5)	-	Fehlercode (Rückmeldung)
(IX + 6)	-	Rückkehradresse im Fehlerfall (Low Byte)
(IX + 7)	-	Rückkehradresse im Fehlerfall (High Byte)

Bei jedem Systemaufruf werden sämtliche Register des Hauptregistersatzes in den Stack gerettet.

Gleichzeitig wird das IY-Register mit dem Wert des Stackpointers geladen. In diesem 'IY-Stack' stehen somit einer Systemfunktion (z.B. einem E/A-Treiber) die ursprünglichen Werte aller CPU-Register zur Verfügung. Nach dem Einsprung enthält:

IY - 6	Returnadresse (Low Byte)
IY - 5	Returnadresse (High Byte)
IY - 4	frei, mit Null vorbelegt
IY - 3	frei, mit Null vorbelegt
IY - 2	frei, mit Null vorbelegt
IY - 1	frei, mit Null vorbelegt
IY + 0	das L-Register
IY + 1	das H-Register
IY + 2	das E-Register
IY + 3	das D-Register
IY + 4	das C-Register
IY + 5	das B-Register
IY + 6	das F-Register
IY + 7	das A-Register
IY + 8	das IY-Register (low byte)
IY + 9	das IY-Register (high byte)

Die Speicherstellen (IY-4) bis (IY-1) sind frei verwendbar und mit Null vorbelegt.

Hinweis:
Der zweite Registersatz der CPU wird vom Floppy Disk Treiber benötigt und deshalb bei allen Systemaufrufen verändert, die Floppy Disk Zugriffe erfordern.

Aus Kompatibilitätsgründen zu früheren KOS-Versionen bleiben die Register A und HL beim Einsprung in eine Systemfunktion unverändert. Die Inhalte der Registerpaare DE und BC sind zu diesem Zeitpunkt undefiniert, jedoch über den IY-Stack jederzeit rekonstruierbar.

Systemfunktionen, die Rückmeldungen an das aufrufende Programm in bestimmten Registern liefern, **müssen** dies über den IY-Stack tun.

Achtung:
An dieser Stelle besteht ein wesentlicher Unterschied zu früheren KOS-Versionen. Insbesondere existierende Eingabetreiber müssen i.a. entsprechend abgeändert werden.

Beispiel:

Ein Eingabetreiber liefert den eingelesenen Wert im Register A zurück.

```
.
call INPUT    ;Zeichen in Reg.A einlesen
ld (iy+7),a   ;Reg. A im IY-Stack ablegen
ret
```

Erläuterung des Vektorinhalts:

(IY+0) - logische Kanalnummer für E/A-Funktionen

(IX+1) - Nummer 'n' der gewünschten Funktion mit der Zuordnung:

0 < n < 40H ---> allgemeine Systemfunktionen
40H < n < 80H ---> Dateiverwaltungsfunktionen
80H < n < C0H ---> Ein-/Ausgabefunktionen

(IX+2) - funktionsabhängig
(IX+3) - funktionsabhängig
(IX+4) - funktionsabhängig

(IX+5) - Rückmeldungscode an das aufrufende Programm. Dieser enthält im Fehlerfalle einen Wert **ungleich 0** mit Bit 7 gesetzt. KOS verwendet folgende Werte:
(siehe auch Anhang A)

80H - unerlaubter Parameter (z.B. Funktion 'n' nicht implementiert)
81H - unerlaubte Funktionsnummer
82H - Gerät (Kanal) nicht bereit
83H - Kanal nicht aktiviert
84H - Datenübertragungsfehler
85H - logischer Satz nicht gefunden oder nicht vorhanden
86H - Medium schreibgeschützt (mechanisch)
87H - unerlaubte Datenrichtung
88H - Medium voll
89H - Speicherbelegungskonflikt
8AH - DSB Liste voll (mehr als 16 Dateien eröffnet)
8BH - Inhaltsverzeichnis-Format Fehler
8CH - Datei schreibgeschützt
8DH - Datei löschgeschützt
8EH - Blockallokation eines Mediums inkonsistent
8FH - reserviert

(IX+6) - Rücksprungadresse für den Fehlerfall
(IX+7) (Rückmeldungscode >80H); wird allerdings nur berücksichtigt, falls ungleich 0, ansonsten erfolgt die Rückkehr zum aufrufenden Programm.

Neben den Rückmeldungscodes >80H, die auf einen 'irreparablen' Fehler hindeuten, sind folgende Codes im Bereich 40H-4FH möglich:

(IX+5) - 40H - Anfang einer Datenübertragung
41H - Ende einer Datenübertragung
42H - Funktion kurzfristig nicht ausführbar (Funktion busy)
43H - Datei bereits eröffnet
44H - Speicher bleibt geschützt (allokiert)
Kehrt ein Anwender- oder Systemprogramm (Task) mit dem Wert 44H unter (IX+5) in das Betriebssystem zurück, so bleibt der für dieses Programm reservierte Speicherbereich allokiert.
45H - Rückmeldung des DV-Aufrufs 'OPEN-FILE', falls eine Datei mit Benutzerkennzeichen logisch nicht vorhanden ist, da ein falsches Benutzerkennzeichen eingegeben wurde.

5. Ein-/Ausgabefunktionen

5.1 Übersicht

Aufruf: RST 8 mit 80H < (IX+1) < COH

Die funktionsabhängigen Parameter (IX+2) bis (IX+4) werden folgendermaßen interpretiert:

(IX+2) - log. Satznummer (lrn) für Medien Ein-/Ausgabe (low byte)
(IX+3) - log. Satznummer (lrn) für Medien Ein-/Ausgabe (medium byte)
(IX+4) - log. Satznummer (lrn) für Medien Ein-/Ausgabe (high byte)

Die folgende Tabelle enthält eine Aufstellung der möglichen E/A-Funktionen und die jeweiligen Ein-/Ausgabeparameter.

Alle komplexen E/A-Funktionen (z.B. BUFIN, ACCOUT, STRING, STOP? etc.) werden auf die Basisfunktionen INPUT, OUTPUT bzw. *STATUS zurückgeführt. In E/A-Treibern erübrigt sich somit die Implementierung anderer Funktionen als INPUT, OUTPUT, OSTATUS (Status for Output) und ISTATUS (Status for Input).

Beim Aufruf einer nicht implementierten Funktion antwortet KOS mit dem Fehlercode 81H.

FNKT.# (IX+1)	Eingangs-parameter A	Eingangs-parameter HL	Ausgangs-parameter A	Ausgangs-parameter HL	Kurzbeschreibung
80H:	nicht implementiert				
81H:OSTATUS	cd	asc	-	-	Status des Ausgabekanals A-n mit Übertragung falls Kanal bereit und cd # 0
82H:ISTATUS	cd	-	-	-	Status des Eingabekanals E-n mit Übertragung falls Zeichen bereit und cd # 0
83H:STOP?	-	-	-	-	Ausgabesteuerung aller KOS Programme Z-Flag = 0 ---> Abbruch-bedingung erkannt (ESC-Taste)
84H:INPUT	-	-	byte	-	Dateneingabe eines Bytes über Kanal E-n
85H:BUFIN	x	adr	y	adr+2	Einlesen von 'x' Zeichen in den Speicher ab 'adr' über Eingabekanal E-n
86H:OUTPUT	byte	-	-	-	Datenausgabe des Bytes 'byte' über Kanal A-n
87H:STRING	-	adr	-	-	Textausgabe (String muß mit 0 enden) auf Kanal A-n
88H:ACCOUT	hex	-	-	-	Ausgabe des Akkus im Hex-Code als zwei ASCII-Zeichen auf Kanal A-n
89H:	nicht implementiert				

FNKT.# (IX+1)	Eingangs-parameter A	Eingangs-parameter HL	Ausgangs-parameter A	Ausgangs-parameter HL	Kurzbeschreibung
8AH:LRREAD	(IX+2)	adr	-	-	Lesen eines log. Satzes von Medium M-n in den Pufferspeicher 'adr'
	(IX+3)				
	(IX+4)				
8BH:LRWRITE	(IX+2)	adr	-	-	Schreiben eines log. Satzes auf Medium M-n aus dem Pufferspeicher 'adr'
	(IX+3)				
	(IX+4)				
8CH:IOCINP	cd		-	-	Ausführung der INIT. OPEN oder CLOSE-Routine eines Ein- (8CH) oder Ausgabetreibers (8DH) entsprechend des Codes 'cd'
8DH:IOCOUTP	cd		-	-	
8EH:GETCRS	-	-	-	crsadr	Abfrage des momentanen Cursorstandes (relative Bildwiederholspeicheradr.)
8FH:GETSCR	-	-	-	scradr	Abfrage der momentanen Scrolladresse (relative Bildwiederholspeicheradr.)
90H:PUTCRS	-	crsadr	-	-	Setzen der Cursoradresse
91H:PUTSCR	-	scradr	-		Setzen der Scrolladresse
92H:GETPNT	-	-	-	eatab	Liefert einen Zeiger auf E/A-Tabellen der E/A-Verwaltung
93H:ASSIGN	An En On	eatname	-	stadr	Zuordnung der Kanalnummern an einen E/A- oder Medientreiber
AO::LSREAD	-	adr	-	-	Lesen eines log. Sektors von Medium M-n in den Pufferspeicher 'adr'
A1:LSWRITE	-	adr	-	-	Schreiben eines log. Sektors auf Medium M-n aus dem Pufferspeicher 'adr'

5.2 Beschreibung der E/A-Funktionen

Funktion 81: OSTATUS

Abfrage des Status von Ausgabekanal A-n. Bei dieser Funktion wird gleichzeitig ein zu übertragendes Zeichen im Register L übergeben, falls Register A einen Wert # 0 enthält. Als Ergebnis zeigt das Zero-Flag der CPU, ob der Ausgabekanal bereit ist, ein Byte zu empfangen. Ist der Kanal bereit, ein Zeichen (Byte) zu empfangen, so muß der Ausgabetreiber das Zeichen im L-Register übertragen (A # 0).

Rückmeldung:
- Z-Flag = 1 ---> Treiber (Kanal) nicht bereit; in diesem Fall enthält (IX+5) den Wert 42H.
- Z-Flag = 0 ---> Treiber (Kanal) bereit; in diesem Fall wurde das Zeichen im L-Register übertragen

Die Funktion 81 ist eine Basisfunktion, die in einem Ausgabetreiber implementiert sein muß. Sie wird von der Printertask 'SPOOL' verwendet.

Funktion 82: ISTATUS

Abfrage des Status von Eingabekanal E-n. Bei dieser Funktion wird ein vorliegendes Zeichen im Register A übergeben, falls A beim Einsprung einen Wert # 0 enthält. Als Ergebnis zeigt das Zero-Flag der CPU, ob ein Zeichen vorliegt oder nicht.

- Z-Flag = 0 ---> ein Zeichen liegt vor in diesem Fall enthält Register A das Zeichen
- Z-Flag = 1 ---> kein Zeichen liegt vor

Die Funktion 82 ist eine Basisfunktion, die in Eingabetreibern implementiert sein muß. Sie wird insbesondere von der Empfängertask 'RTASK' verwendet (PSI80 Modemsoftware).

Funktion 83: STOP?

Steuerung der Textausgabe auf den Sichtschirm des PSI80-Computers. Diese Funktion ermöglicht neben dem Anhalten, Verlangsamen (CNTRL-S wurde gedrückt) und Abbrechen einer Textausgabe auch das 'Hin- und Herblättern' in den 8 Seiten des Bildwiederholspeichers (CNTRL-P war gedrückt).

Als Ergebnis von Funktion 83 zeigt das Zero-Flag der CPU, ob die ESCAPE-Taste gedrückt war oder nicht.

```
Z-Flag = 0 ---> ESCAPE-Taste war gedrückt
Z-Flag = 1 ---> keine oder eine beliebige
                NICHT-ESCAPE-Taste war gedrückt
```

Funktion 83 wird auf die Funktionen 82, 84 und 86 zurückgeführt.

Funktion 84: INPUT

Einlesen eines Zeichens über den logischen Kanal 'n' (Kanalzuordnung siehe 'EAK LIST'-Kommando). Diese Funktion wartet in einer Schleife bis ein Zeichen vorliegt und kehrt dann mit diesem Zeichen im Register A zurück. Der Wert für 'n' (0 < 'n' <9) steht unter (IX+0).

Ein-/Ausgabetreiber müssen das eingelesene Zeichen im IY-Stack (IY+7) abliefern (siehe Abschnitt: Parameterübergabe).

Funktion 85: BUFIN

Einlesen von maximal 'x' Zeichen über den Eingabekanal E-n in einen Pufferspeicher beginnend bei Adresse 'adr+2'. Hierbei kann mit den Tasten CNTRL-H und RUBOUT korrigiert werden. Alle Zeichen werden automatisch an den Ausgabekanal A-n übertragen. Die Rückkehr erfolgt nach 'RETURN' oder der Eingabe von 'x' Zeichen. Register A enthält dann die Anzahl 'y' der tatsächlich eingegebenen Zeichen; Registerpaar HL zeigt auf die Adresse des ersten Zeichens (= 'adr' + 2). Die Funktion 85 verwendet die Speicherstellen 'adr' und 'adr'+1 zur Ablage temporärer Werte. Der Pufferspeicher muß x+3 Byte groß sein.

Die Funktion 85 wird auf die Funktionen 84 (INPUT) und 86 (OUTPUT) zurückgeführt.

Funktion 86: OUTPUT

Ausgabe eines Bytes auf den logischen Kanal 'n' (Kanalzuordnung siehe 'EAK LIST'-Kommando). Der Wert für 'n' (0 < 'n' < 9) steht unter (IX+0).

Funktion 87: STRING

Ausgabe einer mit 0 abgeschlossenen Zeichenkette (= 'String') auf den Kanal A-n. Die Zeichenkette steht in einem Pufferspeicher ab Adresse 'adr'. Funktion 87 wird auf die Funktion 86 (OUTPUT) zurückgeführt.

Funktion 88: ACCOUT

Ausgabe der Hexadezimalzahl im Register A als zwei ASCII-Zeichen auf Kanal A-n. Funktion 88 wird auf die Funktion 86 (OUTPUT) zurückgeführt.

Funktion 89: nicht implementiert

Funktion 8A: LRREAD (logical record read)

Lesen eines logischen Satzes des Mediums M-n in einen Pufferspeicher ab Adresse 'adr'. Ein log. Satz enthält grundsätzlich 128 Byte. Die Funktion 8A liefert deshalb in jedem Fall 128 Byte, auch dann, wenn dies nicht der kleinsten adressierbaren Einheit eines Mediums entspricht. Es enthält:

HL	Pufferadresse 'adr'
(IX+0)	Mediennummer
(IX+2)	low byte der log. Satznummer
(IX+3)	medium byte der log. Satznummer
(IX+4)	high byte der log. Satznummer

Funktion 8B: LRWRITE (logical record write)

Beschreiben des log. Satzes von Medium M-n mit den Daten des Pufferspeichers ab Adresse 'adr'. Es werden 128 Byte geschrieben (Parameterübergabe wie Funktion 8A).

Hinweis:

Bei Medien mit einer physikalischen Satzlänge >128 (z.B. double density Diskette) führt der einmalige Aufruf von Funktion 8B (z.B.: durch das SR-Kommando in KDM) zu keinem Zugriff auf das Medium, da der logische Satz im Treiber zwischengespeichert wird. Der Schreibvorgang erfolgt erst beim nächsten Zugriff auf das Medium über Funktion 8A oder 8B.

Funktion 8C/8D: ICINP/IOCOUTP (INIT, OPEN, CLOSE)

Ausführung der 'INIT, OPEN oder CLOSE-Routine' eines E/A-Treibers. Hierbei wird im Register A ein Code 'cd' übergeben, der eine der drei Routinen auswählt.

A = 0 ---> INIT-ROUTINE
A = 1 ---> OPEN-Routine
A = 2 ---> CLOSE-Routine

Diese beiden Funktionen sind Basis-Routinen für E/A-Treiber und müssen dort implementiert sein. Die KOS Utility Diskette enthält ein Beispiel für einen Treiber.

Funktion 8E/8F: GETCRS/GETSCR

Abfrage des momentanen Standes der Cursor- bzw. Scroll-Adresse. Diese Funktion liefert eine relative Adresse im Bereich von 0 bis 47DOH für die Cursoradresse ('crsadr') bzw. im Bereich 0 bis 3FFFH für die Scrolladresse ('scradr') im Registerpaar HL.

Funktion 90/91: PUTCRS/PUTSCR

Setzen der Cursor- oder Scroll-Adresse ('crsadr' bzw. 'scradr'). Die Werte hierfür liegen in dem für die Funktionen 8E/8F möglichen Bereich.

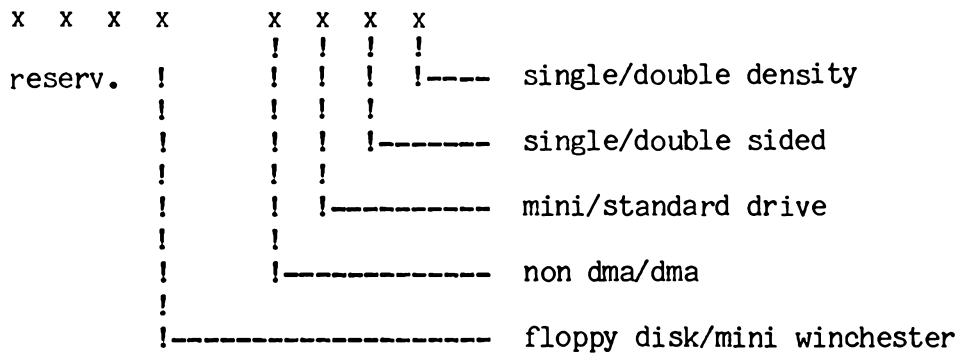
Die Funktionen 8E bis 91 sind kanalunabhängig und adressieren grundsätzlich den Monitortreiber $MON.

Funktion 92H: GETPNT

Funktion 92H liefert eine Tabellenadresse der KOS-E/A-Verwaltung mit Zeigern (jeweils 2 Byte) auf folgende Parameter:

1. Namenstabelle für E/A- und Medienkanäle (Erklärung a)
2. Sprungtabelle für E/A- und Medienkanäle (Erklärung b)
3. wie Punkt 1. (Erklärung a)
4. Sprungtabelle für Medienkanäle (Erklärung b)
5. Motor off Konstante für Minilaufwerke (Erklärung c)
6. Betriebssystemversion mit (neu)

Byte 1 = Versionsnummer (hex) z.B. 53H
Byte 2 = Sprache (ASCII) z.B. 'G'
Byte 3 = Laufwerk Identifikation mit (Bit7 - Bit0)



a) Organisation der Namenstabelle

Diese Tabelle enthält pro Treiber 8 Byte und hat insgesamt Platz für 20 Einträge.

Byte 1	Anzahl der vom Treiber belegten Speichersegmente
Byte 2-3	Startadresse des Treibers
Byte 4	Treiberidentifikation
Byte 5-8	Name des Treibers (ASCII-Code)

b) Organisation der Sprungtabelle

2-Byte-Zeiger auf Sprungtabelle für Ein-, Aus- und Medienkanäle

```
JP   E-0        ; Eingabekanäle
.
.
JP   E-9        ; Ausgabekanäle
JP   A-0
.
.
JP   A-9        ; Medienkanäle
JP   M-0
.
.
JP   M-9
```

c) Motor off Konstante

Die Motoren der Floppy Disk Laufwerke schalten nach etwa 10 sec (= 200H x 20 ms) ab. Die Motor off Konstante (200H) darf vom Anwender verändert werden.

Funktion 93: ASSIGN

Zuweisung der logischen Kanalnummer 'n' (1 < 'n' < 9) an einen E/A- oder Medientreiber. Register A enthält hierbei im unteren Halbbyte den Wert 'n' und im oberen Halbbyte den hexadezimalen Wert 0 für Medientreiber, 'E' für Eingabetreiber bzw. 'A' für Ausgabetreiber. Das Registerpaar HL zeigt auf den Namen 'eatname' des Treibers (maximal vierstelliger Name mit Leerzeichen auf den nichtbesetzten Stellen. Nach der Rückkehr enthält das HL-Registerpaar die Startadresse 'stadr' des Treibers 'eatname', falls dieser aktiviert war. War dies nicht der Fall, so enthält (IX+5) den Fehlercode 83H. Medientreibern kann auch der Kanal 0 zugewiesen werden.

Funktion 94 bis 9F: nicht implementiert

Funktion A0: LSREAD (logical sector read)

Lesen eines logischen Sektors von Medium M-n in einen Pufferspeicher ab Adresse 'adr'.
Ein logischer Sektor ist die kleinste adressierbare Einheit eines Mediums und enthält immer ein 2 hoch n-faches von 128 Byte. Die Anzahl der gelesenen Bytes hängt ab von der Größe der kleinsten adressierbaren Einheit eines Mediums. Ist diese beispielsweise 100H /256), so liefert die Funktion A0 256 Bytes. Die Parameterübergabe entspricht der Funktion 8A (LRREAD).

Funktion A1: LSWRITE (logical sector write)

Beschreiben eines log. Sektors von Medium M-n mit den Daten des Pufferspeichers ab Adresse 'adr'. Ansonsten wie Funktion A0.

Hinweis: Die Funktionen A0/A1 werden weder von KOS noch von irgendeinem standardmäßigen Dienstprogramm verwendet. Sie sind jedoch in allen Medientreibern implementiert ($DSK0/$WIN0 etc.).

5.3 Programmbeispiele

a) periodische Ausgabe eines Textes auf den Sichtschirm mit der Ausgabesteuerung (Funktion 83).

```
BEGIN:
        LD IX, VECTOR       ;IX-Vektor laden
        LD (IX+1), STRING   ;Funktionsnr. für Stringausgabe laden
        LD HL, TEXT         ;Zeiger auf String
        RST 8               ;KOSCAL
        LD (IX+1), STOP?    ;Funktionsnr. für STOP? laden
        RST 8
        JR Z,BEGIN          ;Abbruch falls Z-Flag=0 (ESC-Taste war
                            ;gedrückt
        RET

VECTOR: DEFW 0              ;Kanalnummer 0 (Monitor/Keyboard)
        DEFW 0
        DEFW 0
        DEFW 0

TEXT:   DEFW CRLF
        DEFM 'Testprogramm für KOSCALs - hier die E/A-Aufrufe
                                         83/87'
        DEFB 0

;EQUATES

CRLF    EQU     0AODH
STRING  EQU     87H
STOP?   EQU     83H

        END BEGIN
```

b) Einlesen von ASCII-Zeichen von der Tastatur mit anschließender Ausgabe des zugehörigen hexadezimalen Codes auf den Sichtschirm.

```
START:
        LD IX,VECTOR          ; IX-Vektor laden
START1:
        LD (IX+1),INPUT       ; Funktionsnummer laden
        RST 8                 ; KOSCAL: Zeichen einlesen
        CP ESCAPE             ; Abbruch mit 'ESC'-Taste
        JR Z,ENDEN
        LD (IX+1),ACCOUT      ; Funktionsnummer laden
        RST 8                 ; KOSCAL: Zeichen ausgeben
        JR START1             ; Sprung auf Anfang

ENDEN:
        RET

VEKTOR:
        DEFB 0                ; Kanal 0 (immer Tastatur bzw.
        DEFS 1                ; Monitor)
        DEFW 0
        DEFW 0
        DEFW 0

ESCAPE  EQU  1BH
ACCOUT  EQU  88H
INPUT   EQU  84H

        END START
```

Mit diesem Programm kann z.B. die Tastenbelegung der Tastatur (oder eines anderen Eingabegerätes bei Änderung der Kanalnummer) auf dem Sichtschirm dargestellt werden.

c) Zuordnung der Kanalnummer 5 an den Eingabetreiber $EIN unter Verwendung der Fehlerrückkehradresse im Parametervektor.

```
START:
        LD IX,VECTOR            ;IX-Vektor laden
        LD HL,EATPNT            ;Zeiger auf EAT-Name
        LD A,0E5H               ;E für Eingabe - 5 für Kanalnummer
                                ;an dieser Stelle müßte das Register
                                ;A mit dem Wert 0A5H für Ausgabetreiber
                                ;und 05H für Medientreiber geladen werde
        LD (IX+1),93H           ;E/A-Funktion: ASSIGN
        RST 8                   ;KOSCAL
        RET

EATPNT:
        DEFM 'EIN '             ;EATN-String (leere Stellen müssen mit
                                ;Leerzeichen aufgefüllt sein, maximal
                                ;vierstelliger Name)

VECTOR:
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW ERROR              ;Fehlerrückkehradresse

ERROR:
        LD A,(IX+5)             ;Fehlercode in ACCU laden
        LD (IX+1),3             ;allgemeine Systemfunktion 3 aufrufen
        RST 8
        RST 0                   ;KOS Warmstart (WSTART)

        END START
```

Tritt beim Aufruf der Funktion 93 ein Fehler auf ($EIN ist beispielsweise nicht aktiviert), so kehrt KOS zur Adresse 'ERROR' zurück (Fehlerrückkehradresse).

In obigem Beispiel wird im Fehlerfall die KOS-Funktion 3 aufgerufen, welche eine entsprechende Fehlermeldung ausgibt.

Derselbe Effekt wird erreicht, falls die Fehlerrückkehradresse im Parametervektor den Wert 'KERROR' (=000BH) enthält. In diesem Fall wird automatisch ein KOS Warmstart ausgeführt.

6. Allgemeine Systemfunktionen

6.1 Übersicht

Aufruf: RST 8 mit 0 < (IX+1) < 40H

Zusammenstellung der allgemeinen Systemfunktionen:

Nummer (IX+1)	Eingangsparameter A	HL	DE	Bedeutung
1H:KMDINT	-	adr	-	Ausführung des Kommando-strings ab Adresse 'adr'
2H:DSBGEN	-	dsb	str	Aufbereitung des Strings ab Adresse 'str' zu einem DSB ab Adresse 'dsb'
3H:EMSGOUT	cd	-	-	Ausdruck einer KOS-Fehler- oder Systemmeldung entsprechend des Codes 'cd'
4H:MEMGR	cd	adr	n	Belegung oder Freigabe von 'n' Speichersegmenten ab Adresse 'adr' ('cd' = 1 ---> Belegung) ('cd' = 2 ---> Freigabe) ('cd' = 3 ---> Belegung im höchstmöglichen Speicherbereich)
5H:KOSTAB	-	-	-	Liefert einen Zeiger auf verschiedene Tabellen von KOS
6H:ATASK	-	tcb	-	Aktivierung einer Task beschrieben durch den Task-controlblock ('tcb')
7H:DTASK	n	-	-	Deaktivierung der Task 'n' (0<n<9)
8H:PTASK	-	tname	-	Parameterübertragung an die Task auf deren Namen (HL) zeigt

6.2 Funktionsbeschreibung

Funktion 1: KMDINT

Aufruf des Kommandointerpreters von KOS. Das HL-Registerpaar zeigt hierbei auf einen String von ASCII-Zeichen, der mit (binär) 0 abgeschlossen ist. Funktion 1 führt das (oder die) Kommando(s) des Kommandostrings aus.

Anwenderprogrammen bietet sich hiermit die Möglichkeit, KOS-interne oder externe Systemprogramme aufzurufen.

Beispiel:

Ausführung der Kommandos 'STATUS' und 'IL' von einem Anwenderprogramm aus.

```
START:
        LD HL,CMDSTR            ; Kommandostring-Zeiger
        LD IX,VECTOR            ; IX-Vektor laden
        LD (IX+1),1             ; Funktion 1 aufrufen
        RST 8                   ; KOSCAL
        RET                     ; Rückkehr ins Betriebssystem

CMDSTR:
        DEFM 'STATUS;IL'
        DEFB 0                  ; String Begrenzer

VECTOR:
        DEFW 0
        DEFW 0
        DEFW 0
        DEFW 0

        END START
```

Hinweis:

Das Anwenderprogramm muß im allgemeinen dafür sorgen, daß der für die aufgerufenen Programme notwendige Speicher (ab 100H) frei ist. Es ist deshalb auf eine geeignete Startadresse zu linken:

```
LINK TEST/N,TEST/P:adr/E
```

Der Wert 'adr' bestimmt die Startadresse der Datei TEST.COM. Das Programm TEST kann auch als Objektmodul über das RLOAD-Kommando geladen werden:

```
RLOAD TEST
```

Funktion 2: DSBGEN

Generiert die ersten 16 Byte eines Dateispezifikationsblocks (DSB) aus einem mit (binär) 0 abgeschlossenen String. Folgende Eingangsparameter sind erforderlich:

HL ---> Zeiger auf den Speicherbereich (dsb), in dem der Aufbau des DSB erfolgen soll (16H Byte groß)

DE ---> Zeiger auf den ASCII-String (str)

Nach der Aufbereitung enthält das erste Byte des DSB das sogenannte DSB-Statusbyte (siehe Abschnitt 8).

Beispiel:

Aufbereitung einer Benutzereingabe über die E/A-Funktion 85H zu einem Dateispezifikationsblock mit anschließender Dateieröffnung.

```
START:
        LD IX,KOSVEC          ; IX-Vektor laden
        LD HL,BUFFER          ; Eingabepuffer definieren
        LD A,20               ; Anzahl der Zeichen definieren
        RST 8                 ; Funktion 'BUFIN' aufrufen
                              ; BUFIN meldet zurück:
                              ; A = Anzahl der eingegebenen Zeichen
                              ; HL= Zeiger auf erstes Zeichen
                              ; (BUFFER+2)
        AND A
        JP Z,ZERO             ; Kein Zeichen wurde eingegeben
        LD (IX+1),2           ; Funktionsnummer laden
        EX DE,HL              ; DE ---> String
        LD HL,DSBBHF          : Speicher für DSB (21H Byte)
        RST 8                 ; Funktion 2 aufrufen
                              ; Nach der Rückkehr steht im Bereich
                              ; DSBBUF der Dateispezifikationsblock
        .                     ; weiter im Programm
        .                     ; z.B. DSB-Statusbyte checken
        LD A,(DSBBUF)
        AND 7                 ; Bits 3 bis 7 ausblenden
        LD (DSBBUF),A
        LD (IX+1),62H         ; Funktionsnummer OPEN-FILE
        LD HL,DSBBUF
        RST 8                 ; Funktion 62H aufrufen
        CP OFFH               ; Datei vorhanden?
        JP Z,NEIN             ; falls nicht vorhanden
        .                     ; weiter im Programm, an dieser Stelle
                              ; zeigt HL auf den KOS-kompatiblen DSB
NEIN:   ...
ZERO:   ...                   ; Fehlerroutine
```

```
BUFFER:
        DEFS 23             ; Muß um 3 größer sein als die Anzahl
                            ; der zugelassenen Zeichen
DSBBUF:
        DEFS 21H            ; DSB-Bereich
KOSVEC:
        DEFB 0
        DEFB 85H            ; Funktionsnummer für E/A-Funktion BUFIN
        DEFW 0
        DEFW 0
        DEFW 11             ; KERROR (=000BH)
        END START
```

Zu beachten ist, daß vor dem Aufruf einer Dateiverwaltungsfunktion die obersten 5 Bits des DSB-Statusbytes auf Null zu setzen sind.

Funktion 3: EMSGOUT

Diese Funktion ruft den auf Ausgabekanal A-3 aktiven Treiber zur Ausgabe von Systemmeldungen auf.

Beim Aufruf des Ausgabekanals A-3 kennzeichnet der Wert des Registers A den auszugebenden Text. Es gilt folgende Zuordnung:

A=00 bis 0FH Systemfehler 80H bis 8FH

Hierunter fallen alle Fehlermeldungen, die aufgrund eines Error Return Codes im Parametervektor eines KOS-Aufrufs entstehen (Anhang: A).

Die restlichen Codes werden von KOS für die verschiedenen Systemmeldungen verwendet. Die Zuordnung ist ebenfalls aus Anhang A zu ersehen.

Im Treiber nicht belegte Codes führen zur Fehlermeldung 'Unerlaubter Parameter'.

Standardmäßig werden die Treiber $KSM und $KSML mitgeliefert; $KSML liefert ausführliche Texte. Durch Editierung des Treibers $KSML sind benutzerspezifische Botschaften generierbar.

Der in KOS implementierte Standardtreiber ($KSM - KOS System-Messages) blendet das höherwertige Bit des A-Registers aus. Funktion 3 kann deshalb unmittelbar mit dem Fehlercode im Register A zur Ausgabe einer entsprechenden Fehlermeldung aufgerufen werden.

Funktion 4: MEMMGR

Aufruf der KOS-Speicherverwaltung mit der Möglichkeit 'n' Segmente zu belegen (allocate) oder freizugeben (deallocate). Als Eingangsparameter sind erforderlich:

```
HL ---> Adresse 'adr' des ersten Segments
DE ---> Anzahl 'n' der Segmente
A  ---> Code 'cd' für Allocation/Deallocation mit
        A = 1 ---> Allocation (Belegung)
        A = 2 ---> Deallocation (Freigabe)
```

Bei der Rückkehr von einer Speicherbelegung zeigt das Carry-Flag der CPU an, ob der Speicher frei war oder nicht.

```
C-Flag = 1 ---> Speicher war bereits belegt,
                die geforderte Belegung wurde
                nicht durchgeführt

C-Flag = 0 ---> Speicher wurde korrekt belegt
```

Zusätzlich wird (IX+5) mit dem Wert 89H beschrieben, falls eine Belegung nicht durchgeführt werden konnte. Bei der Freigabe von Speichersegmenten können keine Fehlerbedingungen auftreten.

Die Speicherverwaltung bietet zusätzlich die Möglichkeit 'n' Speichersegmente im höchstmöglichen Adreßbereich zu belegen. Die erforderlichen Eingangsparameter sind:

```
  A = 3 --->  Funktionscode 'cd' für Memory Manager
 DE = n --->  Anzahl 'n' der Segmente mit n<100H
```

Nach dem Rücksprung enthält das HL-Registerpaar die Anfangsadresse des belegten 'n' Segmente großen Speicherbereichs. Bei Speicherbelegungskonflikten enthält (IX+5) den Wert 89H und das Carry Flag der CPU ist gesetzt.

Beispiel:

Verwendung der KOS-Speicherverwaltung. Im folgenden wird der oberste noch freie 2 kByte Speicherbereich gesucht. Dies sind 16 Segmente je 128 Byte.

```
START:
        LD IX,KOSVEC        ; IX-Vektor laden
        LD DE,16            ; Anzahl der Segmente
        LD A,3              ; Code für Suchen
        RST 8
        JR C,NOMEM          ; falls kein Speicher frei
MEMOK:
        .                   ; an dieser Stelle zeigt HL
        .                   ; auf den belegen 2k-Block
        .
        .
NOMEM:
        .                   ; Fehlerbehandlung: kein Speicher frei
        .
        .
KOSVEC:
        DEFB 0.4.0.0.0.0.0.0

        END START
```

Funktion 5: KOSTAB

Diese Funktion liefert im HL-Registerpaar einen Zeiger auf die Adressen verschiedener Pufferbereiche und Tabellen in KOS.

```
Wort 1:    BUFPNT     ; Buffer pointer (Eingabepuffer)
Wort 2;    MMGMAP     ; Tabelle der Speicherverwaltung
Wort 3:    JOBTAB     ; Tabelle der Taskverwaltung
```

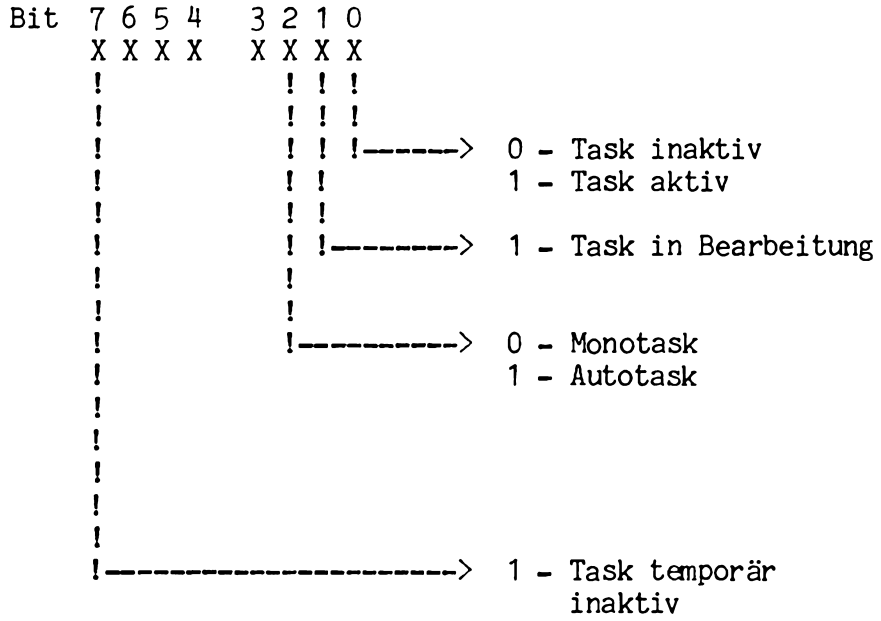
Bedeutung:

BUFPNT - Zeiger auf den momentan in Verwendung befindlichen Kommandoeingabepuffer
MMGMAP - Bitmap der Speicherverwaltung (64 Byte)
JOBTAB - Zeiger auf die Tabelle des Taskcontrolblocks der Taskverwaltung (10 x 16 Byte).

6.3 Hintergrundverarbeitung (Backgroundtasks)

KOS enthält standardmäßig einen sogenannten 'Task scheduler', ein Softwaremodule, das in regelmäßigen Abständen (hier alle 20 ms) eine Taskliste überprüft und aktive Tasks gegebenenfalls ausführt (siehe dazu auch Dienstprogramm: 'TASK'). Eine Task ist durch einen 16 byte großen Parameterblock (Taskcontrolblock 'tcb') bestimmt. Die Bedeutung der einzelnen Bytes ist wie folgt:

Byte 0: **Statusbyte** einer Task mit folgender Bitzuordnung



Alle übrigen Bits sind für spätere Verwendung reserviert.

Bei der Deaktivierung einer Task wird Byte 0 des zugehörigen 'tcb' auf Null gesetzt.

Definitionen:

Monotask: Eine Task wird nur einmal ausgeführt und anschließend automatisch deaktiviert.

Autotask: Eine Task wird periodisch ausgeführt.

Byte 1 = AMS Anzahl der für die Task belegten Speicher-Segmente. Bei der Deaktivierung einer Task werden ab der Adrese TLA AMS Speichersegmente deallokiert.

Bytes 2/3 = PCNT Preset Counter (16 bit). Dieser Zähler bestimmt die Anzahl der 20ms-Perioden, nach der eine Task ausgeführt wird. Ist PCNT beispielsweise 50, so wird die entsprechende Task nach jeder Sekunde (50 x 20 ms = 1 s) ausgeführt.

Bytes 4/5 = DCNT Down Counter (16 bit). Der 'Task Scheduler' verwendet diese beiden Speicherstellen als Rückwärtszähler, beginnend vom Wert PCNT. Eine Task wird nach dem Erreichen des Zählerstandes 0 ausgeführt. Anschließend erhält DCNT den Wert von PCNT, falls Bit 2 des Task Statusbytes (Byte 0) gesetzt ist. Andernfalls wird die Task automatisch deaktiviert.

Bytes 6/7 = TEP Task Entry Point. Einsprungpunkt einer Task. Eine Task darf alle Register des Hauptregistersatzes der CPU verwenden, ohne sie vorher in den Stack zu retten.

Byte 8-13 Name der Task, bestehend aus 6 ASCII-Zeichen mit Blanks auf den nicht besetzten Stellen.

Bytes 14/15 = TLA Task Load Address. Dies ist gewöhnlich die Startadresse des Programmes, das die eigentliche Task definiert. TLA wird zur Speicherverwaltung benötigt und bestimmt die Anfangsadresse der AMS-Speichersegmente, die bei der Deaktivierung einer Task automatisch deallokiert werden.

Zur Verwaltung von Tasks stehen drei Systemfunktionen zur Verfügung:

Funktion 6 : Aktivierung einer Task
Funktion 7 : Setzen des Statusbytes einer Task
Funktion 8 : Parameterübertragung an eine Task

Funktion 6: ATASK

Eingangsparameter: HL ---> 16 byte Parameterblock zur Taskbeschreibung (tcb)

Ausgangsparameter: A ---> Nummer (0<A<9), die dieser Task zugeordnet wurde
(IX+5) = 89H, falls bereits 10 Tasks aktiv.

Die durch den Parameterblock beschriebene Task wird aktiviert. Der Parameterblock wird in eine Tabelle des 'Task Schedulers' eingetragen. Der Eintrag erfolgt an der ersten freien Stelle dieser Tabelle. Ist dort kein freier Platz mehr vorhanden (sind also bereits 10 Tasks aktiv), so antwortet Funktion 6 mit dem Code 89H unter (IX+5). Die Reihenfolge der Taskaktivierung bestimmt gleichzeitig die Priorität der einzelnen Tasks. Eine spätere Prioritätsänderung ist mit dem KOS-Kommando 'TASK' möglich. Funktion 6 liefert im Register A die Nummer 'n' zurück, die der Task zugeordnet wurde.

Funktion 7: DTASK

Eingangsparameter: A ---> Nummer der adressierten Task (0<A<9)
L ---> neues Statusbyte

Ausgangsparameter: keine
(IX+5) = 81H, falls A>10

Das Statusbyte der Task mit der Nummer 'n' wird gesetzt. Ist dieses gleich 0, so wird der entsprechende Eintrag aus der Liste der aktiven Tasks gestrichen (deaktiviert). Ist der Wert 'AMS' ungleich 0, so erfolgt automatisch die Deallokation der durch die Task belegten Speichersegmente.

Funktion 8: PTASK

Eingangsparameter: HL ---> Zeiger auf Taskname
DE ---> Zeiger auf Parameterblock

Ausgangsparameter: keine
(IX+5) = 82H, falls Task nicht aktiv

Diese Funktion dient dazu, beliebig geartete Parameter an eine Task zu übergeben (z.B. Kommandostrings etc.). Hierfür wird ebenfalls der Einsprungpunkt TEP (Task Entry Point) verwendet. Zur Unterscheidung, ob beim Einsprung über TEP die Task ausgeführt werden soll, oder nur Parameter übergeben werden sollen, dient der Wert von Register A.

A = 0 Ausführung der Task
A = 1 Parameterübergabe

Welche Parameter übergeben werden, hängt im Einzelfall von der Task und dem aufrufenden Programm ab.

Taskverwaltung

KOS erlaubt die parallele Bearbeitung mehrerer Tasks, ist jedoch nicht im Sinne eines konventionellen Real-Time-Multitask-Systems architekturiert. Dies hat zur Folge, daß ein Teil des Verwaltungsaufwands von einer Task selbst durchgeführt werden muß, allerdings nur dann, wenn die betreffende Task Systemaufrufe verwendet. In diesem Fall kann es nämlich vorkommen, daß die aufgerufene Systemfunktion momentan nicht in der Lage ist, den Auftrag auszuführen. Sie kehrt deshalb mit dem Returncode 42H unter (IX+5) sofort wieder zurück. In diesem Fall hat die Task selbst dafür zu sorgen, daß der entsprechende Systemaufruf beim nächsten Aufruf der Task nochmals ausgeführt wird.

Benötigt eine Task pro Aufruf mehr als 20 ms, so wird sie automatisch vom 'Task Scheduler' solange unterbrochen, bis alle höher priorisierten Tasks ausgeführt wurden.

Beispiel einer einfachen Task

Eine Task solle alle 10 s den akustischen Ausgang (Ausgabe von CNTRL-G) des PSI80D aktivieren.

Laden des Programms 'BEL'

Das Programm kann entweder als COM-Datei oder über den Relocater des RLOAD-Kommandos als OBJ-Datei geladen und ausgeführt werden. Der vom Programm BEL beanspruchte Speicherbereich wird nach der Rückkehr ins Betriebssystem nicht wieder freigegeben, da (IX+5) den Wert 44H enthält.

```
TLA:      ld ix,vector          ; ix-Vector laden
          ld hl,ttable          ; tcb-Zeiger laden
          rst 8                 ; KOS-Call
          ld (IX+5),44h         ; Return code 44H: KOS gibt
          ret                   ; den Speicher für dieses
                                ; Programm nach der Rückkehr
                                ; nicht frei

AMS       EQU (ende-TLA)/128+1

vector:   deb 0,6,0,0,0,0,0,0

ttable:   defb 5                ; Statusbyte
          defb AMS              ; Anzahl der belegten Speicher-
                                ; segmente
          defw 500              ; Preset Counter (PCNT)
          defw 500              ; Down Counter   (DCNT)
          defw TEP              ; Entry Point der Task (TEP)
          defm 'SOUND '
          defw TLA

TEP:      ld ix, vect1
          ld a, CNTRLG
          rst 8                 ; Funktion 86 aufrufen
          ret

vect 1:   defb 1, 86H, 0,0,0,0,0,0

CNTRLG    equ 07

ende:     END TLA
```

Anmerkung:

Der Einsprungpunkt TEP wird sowohl von Funktion 8 (PTASK), als auch bei der eigentlichen Taskausführung verwendet. Soll hier eine Verzweigung erfolgen, so muß dort zunächst Register A abgefragt werden.

```
TEP:      and a
          jr nz, ptask  ; a>0 ---> Parameterübergabe
          .
          .
          .
ptask:                  ; an dieser Stelle enthält das Registerpaar
                        ; DE direkt oder indirekt den oder die zu
                        ; übergebenden Parameter. Die Art des
                        ; Parameters hängt im Einzelfall von der Task
                        ; und dem Programm ab, das die Funktion 8
                        ; aufruft.
```

7. Dateiverwaltung

7.1. Übersicht

Die Dateiverwaltung von KOS arbeitet grundsätzlich auf logischer Ebene und ist deshalb unabhängig von den spezifischen Eigenschaften verschiedener Medien. Individuelle Treiber bilden die Schnittstelle zwischen logischer und physikalischer Ebene. Jedem Treiber ist eine logische Kanalnummer im Bereich von 0 bis 9 zugeordnet, über die die Dateiverwaltung ein bestimmtes Medium adressiert.

Das Basisdatenelement eines Mediums ist der Satz. Dieser umfaßt grundsätzlich 128 Byte an Daten und ist eine logische Größe, die von der Dateiverwaltung über eine logische Satznummer adressiert wird. Die Satznummer umfaßt 19 bit (0 bis 524288). Dies ergibt eine Maximalkapazität pro Medium von 64 Megabyte. Der einem Medium zugeordnete Treiber konvertiert die logische Größe 'Satznummer' in die für das jeweilige Medium relevanten physikalischen Größen (z.B.: Sektor- und Spurnummern bei Floppy Disk Treibern).

Im allgemeinen Fall entspricht der Satz nicht der kleinsten auf physikalischer Ebene adressierbaren Einheit (z.B. Sektor bei double density FD).

Die Dateiverwaltung benötigt von einem Medium lediglich zwei Informationen, nämlich:

- den Wert cd (Medienidentifikation)
- den Wert 'maxrec' der auf einem Medium speicherbaren Sätze

Somit ist auch das Medium selbst ein für die Dateiverwaltung abstraktes Gebilde, das in der Lage ist, 'maxrec' mal 128 Byte zu speichern. Als abstraktes Gebilde kann es völlig unterschiedlich von einem Medium im herkömmlichen Sinne des Wortes (Floppy Disk, Winchester, Band etc.) sein. Beispielsweise sind virtuelle Medien möglich, die einen Teil des Systemspeichers für temporäre Dateien mit extrem kurzer Zugriffszeit verwenden (siehe: $VMED-Treiber auf der KOS-Systemdiskette).

KOS setzt von allen Medientreibern volle Randomfähigkeit voraus. Es ist Aufgabe des Medientreibers, die erforderliche Verwaltung durchzuführen. Zur Unterstützung führt KOS nach allen Directory-Schreiboperationen automatisch eine Directory-Leseoperation aus.

7.2 Die Organisation eines Mediums

Die Dateiverwaltung von KOS teilt ein Medium in logische Blöcke ein. Ein derartiger Block besteht aus 8 aufeinanderfolgenden Sätzen und stellt den minimalen Platzbedarf einer nicht leeren Datei dar. Die einer Datei zugeordneten Blocknummern sind im Inhaltsverzeichnis gespeichert. Es gilt folgende Zuordnung:

Blocknummer	Satznummern (hexadezimal)
0000	0000-0007
0001	0008-000F
0002	0010-0017
bn	bn x 8 - bn x 8+7 (allgemein)

Das Inhaltsverzeichnis ist prinzipiell eine Datei wie jede andere, mit zwei Ausnahmen:

- es beginnt grundsätzlich mit Block 0
- der Typ der Datei 'Inhaltsverzeichnis' muß 'DIR' lauten.

Der erste Eintrag im Inhaltsverzeichnis weist auf das Inhaltsverzeichnis selbst hin. Im Inhaltsverzeichnis wird pro Datei mindestens ein Satz benötigt (128 Byte). Ein 'Directory record' enthält folgende Informationen:

Byte	Bedeutung
0	00 --> Satz enthält Dateieintrag E5 --> Satz enthält keinen Dateieintrag XX --> alle anderen Werte in Byte 0 führen beim Lesen des Directorys zu einem 'Directory Format Fehler'
1 - 8	Name der Datei im ASCII-Code
9 - 11	Typ der Datei im ASCII-Code
12	Erweiterungszähler (extension counter) Ein 'Directory Satz' bietet Platz für 48 Blocknummern (entsprechend einer Datei von 48 kByte). Ist eine Datei größer als 48 kByte, so ist pro 48 kByte ein eigener Directory Eintrag erforderlich. Byte 12 enthält die Nummer eines 48k Segments (Extension).
13 - 14	Startadresse für Dateien mit ablauffähigen Maschinenprogrammen. Dieser Wert ist nur in der 'nullten' Dateierweiterung enthalten. In allen folgenden Extensions steht in den Bytes 13/14 ein Zeiger auf den vorherigen Directory Eintrag der Datei (Backpointer). Bei Medienkapazitäten größer 8 MByte umfassen For- und Backpointer 19 Bit. In diesen Fällen enthält Byte 27 in den Bitstellen 4-6 die höherwertigen drei Bits des Backpointers und in den Bitstellen 1-3 die höherwertigen drei Bits des Forpointers.

Byte	Bedeutung
15	Dateieigenschaften (file properties) mit der Bedeutung: Bit 0 - Systemdatei Bit 1 - Datei schreibgeschützt Bit 2 - Datei löschgeschützt Bit 3 - Properties gesperrt Bit 4 - nicht verwendet (wird später definiert) Bit 5 - Directory Datei Bit 6 - Datei hat Benutzerkennzeichen Bit 7 - Datei ist 'verborgen'
16 - 17	Datum, an dem die Datei generiert wurde
18 - 19	Benutzerkennzeichen
20 - 25	Backup Flags
26 - 27	Anzahl der Sätze in der Datei(erweiterung) (9 Bit) Die Bits 1-6 von Byte 27 werden bei Medien größer 8 MByte für For- und Backpointer mitverwendet (siehe Byte 13-14. 28-29).
28 - 29	Satznummer des nächsten Eintrags im Inhaltsverzeichnis (Forpointer) (Bit 1-3 von Byte 27 bei Medien größer 8 MByte)
30 - 31	Satzzähler für Schreib/Lese-Zugriffe
32 - 33	Nummer des 1. Blocks der Datei
33 - 34	Nummer des 2. Blocks der Datei (falls vorhanden)
.	
.	
.	
127 - 128	Nummer des 48. Blocks der Datei (falls vorhanden).

Ist eine Datei oder Dateierweiterung kleiner als 48 kByte, so bestimmt der Wert des Satzzählers (Bytes 26/27) die Anzahl der gültigen Einträge im Bereich der Bytes 32 bis 128.

7.3 Die Datei 'Inhaltsverzeichnis'

Die Datei 'Inhaltsverzeichnis' ist die einzige Datei, die auf einem noch jungfräulichen Medium existieren muß. Sie enthält am Anfang gewöhnlich nur einen Eintrag, nämlich den Verweis auf das Inhaltsverzeichnis selbst. Dieser Eintrag wird beim Formatiervorgang automatisch generiert, wobei dem Inhaltsverzeichnis ein beliebiger maximal 8-stelliger Name zugewiesen werden kann. Dieser Name wird bei der Initialisierung eines Mediums zusammen mit dem Belegungsplan des Mediums im Systemspeicher abgelegt.

Der Belegungsplan spiegelt belegte und freie Blöcke eines Mediums wieder. Bei allen Dateieröffnungen wird überprüft, ob der Mediumname noch mit dem ursprünglichen Namen übereinstimmt. Ist das nicht der Fall, so erfolgt eine automatische Neuinitialisierung des betreffenden Mediums. Auf diese Art werden Fehlfunktionen aufgrund gewechselter Disketten etc. vermieden.

Dieser Automatismus kann nur bei unterschiedlichen Directorynamen funktionieren. Bei gleichen Namen muß die Neuinitialisierung durch das entsprechende Systemkommando (N) veranlaßt werden.

Da das Inhaltsverzeichnis als ersten Eintrag immer den Verweis auf sich selbst enthält und mindestens den Block 0 (Satz 0 bis 7) umfaßt erwartet die Dateiverwaltung beim Lesen der entsprechenden Sätze die entsprechenden Informationen. Ist dies nicht der Fall, so resultiert daraus ein 'Directory Format Fehler'.

Beispiel des Formats des ersten Directory Satzes: (logical record 0)

Byte	Wert (hex)	
0	00	
1	XX	)
2	XX	)
3	XX	)
4	XX	)
5	XX	)- Directoryname (ASCII-Zeichen)
6	XX	) wird beim Formatieren eingetragen
7	XX	)
8	XX	)
9	44	)
10	49	)- Directorytyp (DIR)
11	52	)
12	00	
13	00	
14	00	
15	20	- Dateieigenschaften
16	00	)- Systemdatum, wird beim Laden von KOS
17	00	)- in die Speicherstellen OEH/OFH geschrieben
18	00	
19	00	
20	00	)
21	00	)
22	00	)- Backup-flags
23	00	)
24	00	)
25	00	)
26	20	)- Sätze in der Datei, sagt aus, daß
27	00	) vier Blöcke belegt sind
28	00	
29	00	
30	00	
31	00	
32	00	)
33	00	)
34	01	)
35	00	)- Diese Blöcke sind von der Datei
36	02	) 'Directory' belegt
37	00	)
38	03	) (Sätze: 0000 - 001F)
39	00	)
40	00	)- irrelevant, da nur 4 Blöcke belegt sind,
.	00	) sollte allerdings mit Null vorbelegt
.		) sein.
128	00	)

7.4 Dateiverwaltungsfunktionen

Die Dateiverwaltungsfunktionen umfassen das Eröffnen, Lesen, Schreiben oder Schließen von Dateien. Bei allen Aufrufen dieser Funktionen muß das aufrufende Programm einen Parameterblock bereitstellen, der die zu bearbeitende Datei eindeutig beschreibt. Dieser Parameterblock, im folgende DSB - Dateispezifikationsblock - genannt, ist sowohl in einer KOS 3.x, als auch KOS-kompatiblen Form möglich. Aus diesem Grund sind alle Dateiverwaltungsfunktionen über zwei verschiedene Funktionsnummern zugänglich:

a) KOS 3.x-kompatible Aufrufe: 40H < (IX+1) < 5FH

b) KOS 4/5 kompatible Aufrufe: 60H < (IX+1) < 7FH

Die in KOS implementierte Aufwärtskompatibilität zu früheren Versionen garantiert in den meisten Fällen die Ablauffähigkeit bereits existierender Programme unter KOS ohne jegliche Änderungen. Bei **Neuentwicklungen wird empfohlen, KOS-kompatible Aufrufe zu verwenden.**

Alle Dateiverwaltungsfunktionen ab Nummer 42/62 liefern im Register A den Wert 0, falls die Operation erfolgreich beendet wurde und ansonsten den Wert FF. Das Auftreten des Wertes FF kann eventuell seine Ursachen auf tieferen Ebenen als der Dateiverwaltung haben. In diesem Fall enhält (IX+5) des Parametervektors einen entsprechenden Fehlercode.

Beispiel:

Eine Datei soll auf Medium 6, welches nicht aktiviert ist, eröffnet werden. Nach der Rückkehr enthält A den Wert FF (die Datei ist logisch nicht vorhanden) und (IX+5) den Fehlercode 83H (Kanal 6 ist nicht aktiv).

7.4.1 Dateispezifikationsblock

Der Dateispezifikationsblock - DSB kennzeichnet in eindeutiger Weise die zu bearbeitende Datei oder Dateigruppe. Der Aufbau des DSB ist im folgenden erläutert.

a) der KOS 3.x (und auch CP/M 1.4) kompatible DSB

Dieser Parameterblock umfaßt 33 Byte. Er beschreibt 16 Kbyte einer Datei(erweiterung) und liegt grundsätzlich im Speicherbereich des aufrufenden Programms. Die Bedeutung der einzelnen Bytes ist wie folgt:

Byte 0	Mediennummer mit 0 ---> derzeitiges Mastermedium 1 ---> Medium 0 A ---> Medium 9
Byte 1 - 8	Dateiname (ASCII-Code mit Leerzeichen auf den nicht besetzten Stellen)
Byte 9 - 11	Dateityp (ASCII-Code mit Leerzeichen auf den nicht besetzten Stellen)
Byte 12	Dateierweiterungszähler (Extension Counter)
Byte 13 - 14	Ladeadresse für ablauffähige Maschinenprogramme, falls ungleich 0, ansonsten gilt 100H als Voreinstellung.
Byte 15	in KOS 3.x die Anzahl der Sektoren in der Datei(erweiterung)

Achtung: Dieser Wert wird in KOS zur Kennzeichnung der Dateieigenschaften (file properties) verwendet.

Vorsicht ist an dieser Stelle geboten, falls existierende Anwenderprogramme Byte 15 des DSB verwenden, um daraus die Dateilänge abzuleiten.

Byte 16 - 31	in KOS 3.x von der Dateiverwaltung als Belegungstabelle verwendet, für Anwenderprogramme gewöhnlich bedeutungslos.

Achtung: KOS verwendet diese Stellen anderweitig. Die Belegungstabelle ist nur im KOS kompatiblen DSB vorhanden.

Byte 32	Nummer des nächsten Satzes bei Schreib-/Lese-Zugriffen (NR: Next Record Counter)

Alle existierenden unter KOS 3.x erzeugten Programme sind bezüglich der DV-Funktionen unverändert unter KOS ablauffähig, sofern sie Byte 15 des DSB nicht verwenden.

b) der KOS 4/5-kompatible DSB

Der KOS kompatible DSB umfaßt 128 Byte und entspricht exakt einem Eintrag im Inhaltsverzeichnis eines Mediums. Beim Eröffnen einer Datei muß das aufrufende Programm lediglich die ersten 16 Byte (Mediennummer, Name, Typ, Extension) bereitstellen. Die Dateiverwaltung von KOS sucht daraufhin automatisch ein freies Speichersegment im Systemspeicher und generiert dort den gesamten 128 Byte DSB. Einer Datei wird beim Eröffnen eine Nummer zugewiesen (0 bis 15), die im oberen Halbbyte des ersten Bytes im DSB abgelegt wird.

Alle Dateiverwaltungsfunktionen, die Dateien manipulieren, liefern als Ergebnis einen Zeiger auf den von KOS generierten DSB. Die Bedeutung der einzelnen Bytes des KOS-kompatiblen DSB ist wie folgt:

Byte 0 — Mediennummer (niederwertiges Halbbyte)
log. Dateinummer (höherwertiges Halbbyte).
Diese wird von der Dateiverwaltung beim Eröffnen eingetraen.

Beide werden nicht auf dem Medium abgelegt. Dort steht an dieser Stelle immer Null, mit Ausnahme bei gelöschten Dateien, wo an dieser Stelle der Wert E5 steht.

Byte 1 - 8 — Name der Datei (8-stellig mit Leerzeichen auf den nicht besetzten Stellen).

Byte 9 - 11 — Typ der Datei (3-stellig mit Leerzeichen auf den nicht besetzten Stellen).

Byte 12 — Dateierweiterungszähler (Extension Counter)

Der DSB von KOS beschreibt 48 kByte einer Datei. Bei größeren Dateien ist pro 48 kByte Segment ein eigener Eintrag im Inhaltsverzeichnis erforderlich.
Byte 12 enthält die Nummer des Segments.

Byte 13 - 14 — Ladeadresse für ablauffähige Maschinenprogramme (nur für das erste 48 kByte Segment abgelegt).

Bei Dateien größer 48 kByte enthält der Inhaltsverzeichnis-Eintrag für das n-te Segment in den Bytes 13/14 die log. Satznummer des Inhaltsverzeichnis-Eintrags für das (n-1)-te Segment (Backpointer).
Bei Medien mit einer Gesamtkapazität von mehr als 8 MByte umfaßt der Backpointer 19 Bit.
Die höherwertigen 3 Bits sind in den Bitstellen 4-6 von Byte 27 enthalten.

Byte 15 Dateieigenschaften (Properties)

Bit 0 - Systemdatei
Bit 1 - Datei schreibgeschützt
Bit 2 - Datei löschgeschützt
Bit 3 - Properties gesperrt
Bit 4 - nicht verwendet (wird später definiert)
Bit 5 - Directory Datei
Bit 6 - Datei hat Benutzerkennzeichen
Bit 7 - Datei ist 'verborgen'

Byte 16 - 17 Datum, an dem die Datei generiert wurde

Byte 18 - 19 Benutzerkennzeichen

Byte 20 - 25 Backup Flags

Byte 26 - 27 Anzahl der Sätze in der Datei(erweiterung)
Dies ist ein Wert zwischen 0 und 180H.
Der Wert 180H (9 Bit) deutet darauf hin, daß ein weiteres 48 kByte Segment vorhanden ist.
Die Bits 1-6 von Byte 27 enthalten die höherwertigen 3 Bits des For- und Backpointers.

Byte 28 - 29 Satznummer des nächsten Eintrags im Inhaltsverzeichnis.
Ist eine Datei(erweiterung) größer als 48 kByte. so enthalten diese Bytes die log. Satznummer des nächsten Directorysatzes für diese Datei (Forpointer).
Bei Medien mit einer Gesamtkapazität von mehr als 8 MByte umfaßt der Forpointer 19 Bit.
Die höherwertigen 3 Bits sind in den Bitstellen 1-3 von Byte 27 enthalten.

Byte 30 - 31 Satzzähler für Schreib/Lesezugriffe.
Dieser bestimmt die Nummer des Satzes, der bei Schreib/Lese-Zugriffen verwendet wird.
Dies ist ein Wert zwischen 0 und 180H bei sequentiellen Dateizugriffen (Funktionen 67/68). Bei wahlfreien Dateizugriffen (Funktionen: 77/78) ist dies ein beliebiger Wert von 0 bis FFFF.

Byte 32 - 127 log. Blocknummern die dieser Datei zugeordnet sind (1 Block enthält 8 Sätze).
Ist eine Datei(erweiterung) kleiner als 48 kByte, so bestimmt der Satzzähler (Byte 26 - 27) die Anzahl der relevanten Einträge im Bereich der Bytes 32 - 127.

Die Blocknummer umfaßt 2 Byte. KOS kann somit Medien bis 65535 kByte (entsprechend 64 Megabyte) verwalten.

7.4.2 Beschreibung der Dateiverwaltungsfunktionen

Zusammenstelung der unter KOS definierten Dateiverwaltungsfunktionen (die Funktionen 60H...6CH entsprechen den KOS 3.x-Funktionen 40H...4CH und sind auch über diese Aufrufe für KOS 3.x-Programme zugänglich):

Aufruf: **RST8 mit 40H < (IX+1) < 80H**

Nummer (IX+1)	Eingangs-parameter A	Eingangs-parameter HL	Ausgangsparameter A	Ausgangsparameter HL
60H:INIT	-	-	y	y
61H:DEFMAS	n	-	y	y
62H:OPEN	-	DSB	FF: nicht vorhanden 0 : Datei vorhanden	x DSB5
63H:CLOSEW	-	DSB5	FF: nicht vorhanden 0 : Datei geschl.	x x
64H:SEARCH	-	DSB	FF: nicht gefunden 0 : Datei gefunden	x DSB5
65H:SEARCHN	-	DSB	FF: keine Datei mehr vorhanden 0 : Datei vorhanden	x DSB5
66H:DELETE	-	DSB	FF: Datei nicht gelöscht 0 : Datei gelöscht	x x
67H:READS	-	DSB5	0 : Ende der Datei 0 : erfolgreich gelesen	y y
68H:WRITES	-	DSB	FF: Inhaltsverz. oder Diskette voll 0 : erfolgreich geschrieben	y
69H:MAKE	-	DSB	FF: Inhaltsverz. voll 0 : Datei eingetragen	x DSB5
6AH:RENAME	-	DSB	FF: alte Datei nicht vorhanden 0 : Name geändert	x x
6BH:SETADR	-	adr	y	y
6CH:MASTER?	-	-	n	y

Nummer (IX+1)	Eingangs-parameter A	Eingangs-parameter HL	Ausgangsparameter A	Ausgangsparameter HL
6DH:CLOSER	-	DSB5	0 : Datei geschlossen FF: Datei nicht geschlossen	DSB5 x
6EH:CLOSEA	-	-	y : Schließen aller geöffneten Dateien	x
6FH:	nicht implementiert			
70H:DVMTAB	-	-	y	pnt
71H:DISPAR	n	pnt	y	y
77H:RD-RANDOM	-	DSB5	0 : log. Satz gelesen F : log. Satz nicht gelesen	DSB5 x
78H:WR-RANDOM	-	DSB5	0 : log. Satz geschrieben FF: log. Satz nicht geschrieben	DSB5 x
79H:EOF	-	DSB5	EX	RC

Es bedeuten: x - Wert ist undefiniert
y - Wert bleibt unverändert
DSB - Dateispezifikationsblock vor Aufruf (16 Byte)
DSB5 - Dateispezifikationsblock nach Aufruf
EX - Extension Counter
RC - Record Counter

Funktion 40/60: INIT-DV

Initialisierung der Dateiverwaltung. Bei dieser Gelegenheit werden je nach Eingangsparameter alle aktiven oder nur ein bestimmtes Medium neu initialisiert. Bei der Initialisierung eines Mediums wird das Inhaltsverzeichnis gelesen und daraus der Gesamtbelegungsplan des Mediums errechnet und im Systemspeicher zusammen mit dem Namen der Datei 'Inhaltsverzeichnis' abgelegt. Dieser Name dient der Dateiverwaltung als Mediumidentifikation. Außerdem werden alle geöffneten Dateien dieses Mediums geschlossen.

Die Initialisierung eines Mediums erfolgt automatisch, wenn beim Eröffnen einer Datei ein Wechsel der Mediumidentifikation des adressierten Mediums festgestellt wird. Eine Neuinitialisierung (N-Komando) von Disketten nach einem Diskettenwechsel ist deshalb nicht erforderlich, sofern diese unterschiedliche Namen aufweisen.

Es wird dringend empfohlen, die Verwendung von gleichnamigen Disketten (siehe FORMAT-Kommando) zu vermeiden.

Eingangsparameter: A = '*' alle aktiven Medien werden initialisiert
A = n Medium n (0<n<9) wird initialisiert

INIT-DV schließt alle Dateien des jeweiligen Mediums.

Funktion 41/61: DEFINE-MASTER

Das Medium 'n' wird zum Mastermedium deklariert. Der zulässige Bereich von 'n' liegt zwischen 0 und 9 und wird im Register A übergeben.

Das Mastermedium ist definitionsgemäß das Medium, auf dem bei Dateieröffnungen ohne Mediennummer als erstes gesucht wird. Noch nicht existierende Dateien werden auf dem Mastermedium eröffnet.

Funktion 42/62: OPEN-FILE

Eröffnet die durch den DSB bestimmte Datei. Falls keine Mediennummer angegeben ist (Byte 0 des DSB = 0), wird eine automatische Dateisuchsequenz auf allen aktiven Medien (beginnend mit dem Mastermedium) gestartet. Ist die Datei vorhanden, so wird der entsprechende Eintrag des Inhaltsverzeichnis in ein freies Speichersegment des Systems geladen. Nach der Rückkehr zum aufrufenden Programm zeigt das Registerpaar HL auf das erste Byte des 128 Byte DSB.

Byte 0 enthält hierbei in den Bits D0 bis D3 die Nummer des Mediums, auf dem die Datei eröffnet wurde. Die Bits D4 bis D7 werden von der Dateiverwaltung für interne Zwecke verwendet. **Byte 0 des DSB darf vom Anwenderprogramm nach dem Eröffnen nicht verändert werden.**

Alle Dateiverwaltungsfunktionen mit Ausnahme von SEARCH und SEARCH-NEXT erfordern das vorausgehende Eröffnen der gewünschten Datei. Da die Dateiverwaltung für jede geöffnete Datei Speicher belegt, ist am Ende einer Dateibearbeitung in jedem Fall die Funktion CLOSEW oder CLOSER erforderlich (close after write bzw. close after read).

Das Eröffnen einer Datei über Funktion 62 erfordert lediglich einen 16 Byte großen Parameterblock entsprechend den ersten 16 Byte des KOS kompatiblen DSB. Nach der Rückkehr enthält Register A den Wert 0, falls die Datei vorhanden war, ansonsten den Wert FF.

Ausgangsparameter:	A = 00	Datei vorhanden
	A = FF	Datei nicht vorhanden
	HL	Zeiger auf KOS kompatiblen DSB, falls Datei vorhanden, sonst undefiniert

Hat eine Datei ein Benutzerkennzeichen, so führt der Versuch, eine Datei zu eröffnen zur Aufforderung, das Benutzerkennzeichen einzugeben. Die Datei ist logisch nicht vorhanden, falls das eingegebene Benutzerkennzeichen mit dem gespeicherten nicht übereinstimmt. In diesem Fall enthält (IX+5) den Wert 45H.

Funktion 43/63: CLOSEWFILE (close after write)

Schließt die durch den DSB bestimmte Datei, sofern diese nicht schreibgeschützt ist. Bei dieser Gelegenheit erfolgt der Eintrag des KOS kompatiblen DSB in das Inhaltsverzeichnis eines Mediums. Außerdem wird die geschlossene Datei aus der Liste der geöffneten Dateien gestrichen und das durch den DSB belegte Speichersegment freigegeben. Die Funktion CLOSEW ist am Ende eines Dateischreibvorgangs (Funktionen 47/67) notwendig, nicht aber am Ende eines Lesevorgangs. Das logische Schließen einer Datei nach einem Lesevorgang erfolgt mit der Funktion 71 (CLOSER-FILE - close after read).

Ausgangsparameter:	A = 00	Datei geschlossen
	A = FF	Datei nicht geschlossen
	HL	undefiniert

Funktion 44/64: SEARCH-FILE

Sucht die durch den Dateispezifikationsblock bestimmte Datei. Hierbei darf der DSB im Bereich des Namens/Typ Fragezeichen (ASCII-Code: 3F) als Repräsentanten eines beliebigen Zeichens enthalten. Funktion 44/64 findet den ersten Eintrag des Inhaltsverzeichnis, der dem DSB entspricht. In diesem Fall enthält das Register A den Wert 0. HL zeigt auf den DSB der gefundenen Datei. Wurde keine Datei gefunden, so enthält Register A nach der Rückkehr den Wert FF.

Ausgangsparameter:	A = 00	Datei gefunden
	A = FF	Datei nicht gefunden
	HL	Zeiger auf DSB, falls Datei gefunden, sonst undefiniert

Funktion 45/65: SEARCH-NEXT

Diese Funktion ist nur nach Funktion 44/64 möglich und sucht den nächsten Eintrag des Inhaltsverzeichnis eines Mediums, der dem DSB entspricht. Wurde keine Datei mehr gefunden, so enthält Register A den Wert FF, ansonsten den Wert 0. Bei allen SEARCH-Funktionen ist es nicht erforderlich, eine Datei zu eröffnen.

Ausgangsparameter:	A = 00	Datei gefunden
	A = FF	keine Datei mehr gefunden
	HL	Zeiger auf DSB, falls Datei gefunden, sonst undefiniert

Die SEARCH-Funktionen 44/45/64/65 berücksichtigen die Eigenschaften einer Datei (file properties). Dies hat zur Folge, daß beim Aufruf einer dieser Funktionen Bedingungen bezüglich der Properties spezifiziert werden können, so daß nur Dateien mit bestimmten Properties oder Kombinationen von Properties gefunden werden. Die Bedingung wird in Form einer Maske in Byte 15 des DSB erwartet. Steht dort der Wert 0, so wird eine Datei nur dann gefunden, wenn sie nicht 'verborgen' ist. Durch das Setzen entsprechender Bits können Dateien, die nicht mindestens die in der Maske spezifizierten Properties haben, ausgeklammert werden.

Funktion 46/66: DELETE-FILE

Löscht die durch den Dateispezifikationsblock bestimmte Datei. Hierbei wird im Inhaltsverzeichnis eines Mediums der Wert E5H an die erste Stelle des entsprechenden Dateieintrags gschrieben. Die DELETE-Funktion erfordert nicht das vorherige Eröffnen einer Datei. Eine bereits geöffnete Datei wird durch die DELETE-Funktion automatisch geschlossen.

Ausgangsparameter: A = 00 Datei gelöscht
A = FF Datei nicht gelöscht
HL definiert

Die DELETE-Funktion wird nicht ausgeführt bei Dateien, die schreib- oder löschgeschützt sind (Bit 1 und/oder 2 der Properties).

Funktion 47/67: READ-RECORD

Lesen des n-ten Satzes der durch den DSB bestimmten Datei. Bei Funktion 47 wird der Wert n aus den Größen NS (Byte 3 des KOS 3.2 kompatiblen DSB: Next record counter) und EX (Byte 12 des DSB: Extension Counter) errechnet.

Für Funktion 67, die mit einem Zeiger auf einen KOS-kompatiblen DSB aufzurufen ist, steht der Recordzähler auf den Bytestellen 30/31. In jedem Fall wird nach dem Lesen der entsprechende Recordcounter automatisch um eins erhöht. Falls erforderlich, eröffnet KOS ebenfalls automatisch die nächste Erweiterung einer Datei (immer dann, wenn eine 48k-Grenze überschritten wird). READ-RECORD liest in die durch Funktion 4B/6B bestimmte Adresse, falls (IX+2) und (IX+3) den Wert 0 haben. Andernfalls bestimmen (IX+2/3) die Pufferadresse.

Funktion 67 ermöglicht sequentiellen Zugriff auf die einzelnen Sätze einer Datei.

Ausgangsparameter: A = 00 Record gelesen
A = FF Ende der Datei (oder Übertragungsfehler)
HL unverändert

Wahlfreier Zugriff ist in beschränktem Umfang durch entsprechendes Setzen des Recordzählers (Byte 30/31) möglich. Da der Wert des Recordzählers 180H nicht überschreiten darf, ist random access mit Funktion 47/67 nur noch innerhalb eines 48k-Segments möglich.

Hinweis:
Mit den Funktionen 77/78 bietet KOS den uneingeschränkten wahlfreien Zugriff auf die einzelnen Sätze einer bis zu 8 MByte großen Datei.

Funktion 48/68: WRITE-RECORD

Für diese Funktion gilt sinngemäß Funktion 47/67.
WRITE-RECORD wird nicht ausgeführt bei schreibgeschützten Dateien.

Ausgangsparameter: A = 00 Record geschrieben
A = FF Record nicht geschrieben
(Medium voll oder Hardwarefehler).
Im zweiten Fall enthält (IX+5)
den entsprechenden Fehlercode.
HL unverändert

Funktion 49/69: MAKE-FILE

Generierung der durch den DSB bestimmten Datei. Nur die ersten 16 Byte des DSB relevant. MAKE-FILE bewirkt automatisch ein Eröffnen der Datei. Dieser wird zunächst kein Speicherplatz auf dem Medium zugewiesen. Allerdings wird ein Eintrag im Inhaltsverzeichnis des adressierten Mediums erzeugt, wobei die Bytes 32 bis 127 Null gesetzt sind. Das Systemdatum wird automatisch in die Bytes 16/17 eingetragen.

Ausgangsparameter: A = 00 Datei generiert
A = FF Datei nicht generiert
HL Zeiger auf DSB, falls Datei generiert,
ansonsten undefiniert

Funktion 4A/6A: RENAME-FILE

Umbenennung einer Datei. Hierfür ist ein spezieller 32 Byte DSB erforderlich, der in den Bytes 0 bis 11 den alten Dateinamen und in den Bytes 16 bis 27 den neuen Dateinamen beschreibt. Die Bytes 12 bis 15, sowie 28 bis 31 sind irrelevant.

Ausgangsparameter: A = 00 Datei umbenannt
A = FF Datei nicht umbenannt
HL undefiniert

Funktion 4B/6B: SET-IOADR

Definition der Adresse des Pufferspeichers mit dem der Datentransfer bei Schreib-/Lese-Zugriffen (Funktionen 47/67 bzw. 48/68) stattfindet. Diese Adresse kann nur durch die Funktion 4B/6B verändert werden und muß vor jedem READ/WRITE-RECORD entsprechend gesetzt werden.

Eingangsparameter: HL Pufferadresse

Achtung:
Ab KOS 4.3/5.3 kann die Pufferadresse direkt im Aufrufvektor unter (IX+2) bzw. (IX+3) angegeben werden. Enthalten (IX+2/3) den Wert Null, so wird die durch Funktion 4B/6B zuletzt definierte Adresse verwendet.

Funktion 4C/6C: MASTER?

Frage nach der Nummer des derzeitigen Mastermediums; kehrt mit dem Binärwert 'n' der Nummer des derzeitigen Mastermediums im Register A zurück. Der Wert für 'n' liegt zwischen 0 und 9.

Ausgangsparameter: A = n Mediennummer

Funktion —/6D: CLOSER-FILE (close after read)

Diese Funktion schließt eine Datei am Ende eines Lesevorgangs. Dabei wird die angegebene Datei aus der Liste der geöffneten Dateien gestrichen und das von der Dateiverwaltung für den DSB belegte Speichersegment freigegeben. Da die Anzahl der gleichzeitig geöffneten Dateien beschränkt ist, wird dringend empfohlen, nach dem Lesen einer Datei Funktion 6D zu verwenden.

Ausgangsparameter: A = 00 Datei geschlossen
A = FF Datei war nicht geöffnet
HL unverändert

Funktion —/6E: CLOSE-ALL

Schließt alle geöffneten Dateien und wird vom KOS-internen Kommando F aufgerufen.

Funktion 6F: nicht implementiert

Funktion —/70: KPOINTER

Diese Funktion liefert einen Zeiger auf folgende Pointertabelle (jeweils 2 Byte) der Dateiverwaltung.

a) Zeiger auf die Tabelle der DSB5 der momentan geöffneten Dateien. Die Einträge sind nach der logischen Dateinummer geordnet. Hat ein Eintrag den Wert 0, so ist die entsprechende logische Dateinummer frei. Es sind maximal 16 Einträge zu je 2 Byte (= 16 geöffnete Dateien) möglich.

b) Zeiger auf die Tabelle der Anfangsadressen der Medienbelegungspläne. Die Einträge sind nach folgendem Schema angeordnet:

1. Eintrag: reserviert (2 Byte)
2. Eintrag: Adresse des Belegungsplans von Medium 0 (2 Byte)
3. Eintrag: Adresse des Belegungsplans von Medium 1 (2 Byte)
.
.
.
11. Eintrag: Adresse des Belegungsplans von Medium 9 (2 Byte)

Die ersten 8 Byte des Belegungsplans enthalten den Namen der Datei 'Inhaltsverzeichnis' des entsprechenden Mediums. Pro verwendeter Blocknummer wird im Belegungsplan ein Bit gesetzt. Die Länge des Belegungsplans errechnet sich aus der Kapazität des Mediums.

c) Zeiger auf die Tabelle, die die Kapazität bzw. die Anzahl der logischen Sätze (1 Satz = 128 Byte) eines Speichermediums enthält. Hierfür werden jeweils 3 Byte benötigt und nach folgendem Schema eingetragen.

1. Eintrag: reserviert (3 Byte)
2. Eintrag: Kapazität Medium 0 (3 Byte)
3. Eintrag: Kapazität Medium 1 (3 Byte)
.
.
.
11. Eintrag: Kapazität Medium 9 (3 Byte)

d) Zeiger auf die Tabelle der Identifikationsbytes der Speichermedien. Die physikalische Satzlänge eines Speichermediums kann ein Vielfaches der logischen Satzlänge betragen. Der Wert n steht im unteren Halbbyte des 1 Byte langen Eintrages. Das obere Halbbyte ist reserviert. Die Reihenfolge der Einträge ist:

1. Eintrag: reserviert (1 Byte)
2. Eintrag: Identifikation Medium 0 (1 Byte)
.
.
.
11. Eintrag: Identifikation Medium 9 (1 Byte)

Die Medienidentifikation wird derzeit nur vom STATUS-Kommando verwendet.

e) Zeiger auf die Tabelle, die die Satznummer des DSB der momentan offenen Datei enthält. Die Einträge sind nach der logischen Dateinummer geordnet und benötigen jeweils 3 Byte.

Alle Tabellen dürfen von Anwenderprogrammen nicht verändert werden!

Funktion --/71: DISKPAR

Definiert die für ein Medium spezifischen Parameter.

cd - Treiberidentifikation (bis KOS 4.2/5.2 die Satzlänge n)
maxrec - Anzahl der verfügbaren log. Sätze (128 Byte Blöcke) eines Mediums.

Beim Aufruf der Funktion 71 zeigt HL auf einen 4 Byte-Parameterblock. Register A enthält die Nummer des Mediums, dem diese Parameter zugeordnet werden sollen.

Parameterblock:	cd	-	Treiberidentifikation
	maxrec	-	Anzahl der log. Sätze auf dem Medium (3 byte)

Dieser Parameterblock wird von der STATUS-Routine eines Medientreibers bereitgestellt (siehe Abschnitt: Medientreiber).

Treiberidentifikation:

niederwertiges Halbbyte:	physikalische Satzlänge (1 = 128, 2 = 256 etc.)
höherwertiges Halbbyte:	0 (reserviert für zukünftige KOS-Versionen)

Funktion -/77: READ-RANDOM
-/78: WRITE-RANDOM

Diese beiden Funktionen ermöglichen den wahlfreien Zugriff auf alle Sätze einer Datei. Hierzu muß lediglich der Satzzähler (Byte 30/31 des DSB) auf den gewünschten Wert (0<RC<FFFF) gesetzt werden. KOS eröffnet automatisch die richtige Dateiextension.

Achtung:
Wahlfreier Schreibzugriff ist nur auf existierende Sätze einer Datei möglich.

Eingangsparameter:	(IX+2)	Pufferadresse (low byte)
	(IX+3)	Pufferadresse (high byte)
Ausgangsparameter:	A = 00	Record gelesen/geschrieben
	A = FF	Record nicht gelesen/geschrieben (Dateiende überschritten)
	HL	unverändert

Als Eingangsparameter muß HL auf einen vollständigen 128-byte DSB zeigen. Der Recordcounter wird automatisch um 1 erhöht.

Achtung:
Im Gegensatz zum sequentiellen Schreib-/Lesezugriff (Funktion: 67/68 bzw. 47/48) erfordern die Random-Funktionen 77/78 die Pufferadresse des Speichers, mit dem der Datentransfer stattfindet grundsätzlich unter (IX+2) und (IX+3).

Funktion —/79: EOF-FILE

Diese Funktion setzt den Extension- und Recordcounter der durch einen vollständigen 128-byte DSB spezifizierten Datei auf das Dateiende. Nach dieser Funktion kann eine Datei unmittelbar mit der Funktion 68 (WRITE-RECORD) erweitert werden.

Ausgangsparameter: A = EX Nummer der Dateierweiterung
HL = RC Recordcounter für die letzte Dateierweiterung

7.4.3 Programmbeispiel

Aus diesem Beispiel ist auch die Verwendung der KOS-Speicherverwaltung ersichtlich; es lädt die Datei TEST.ABC in den Arbeitsspeicher ab Adresse 4000H.

```
START:
        LD IX,VECTOR        ; IX-Vektor laden
        CALL OPEN           ; Datei eröffnen
        CP OFFH             ; vorhanden?
        JR Z,NOTFND         ; falls nicht gefunden
        PUSH HL
        POP IY              ; IY zeigt nun auf den DSB
        LD,E (IY+26)
        LD,D (IY+27)        ; Länge der Datei ermitteln
        CALL ALLOCM         ; Speicher belegen
        JR C,MERROR         ; falls Speicher belegt
        LD B,E
        LD HL,4000H
RLOOP:
        CALL SETADR         ; Pufferadresse definieren
                            ; in diesen Puffer wird der
                            ; Sektor geladen

        PUSH HL             ;
        CALL READ
        POP HL              ;
        CP   0
        JR NZ,RERROR        ; falls Lesefehler
        LD DE,128
        ADD HL,DE           ; Pufferadresse erhöhen
        DJNZ RLOOP          ; B ist Sektorzähler
        .                   ; weiter im Programm
        .
OPEN:
        LD (IX+1), 62H      ; Code für Datei eröffnen
        LD HL,DSB           ; Zeiger auf DSB
        RST 8               ; KOSCAL: OPEN-FILE
        RET
ALLOCM:
        LD (IX+1), 4        ; Funktionsnummer für Speicher-
                            ; verwaltung
        LD HL,4000H
        LD A,1              ; DE enthält die Anzahl der
                            ; Segmente für Allocation
        RST 8               ; KOSCAL: MEMMGR
        RET
SETADR:
        LD (IX+1), 6BH      ; Funktionsnummer laden
        RST 8               ; KOSCAL: SET-IOADR
        RET
READ:
        PUSH IY
        POP HL
        LD (IX+1), 67H      ; Funktionsnummer laden
                            ; Zeiger auf DSB
        RST 8               ; KOSCAL: READ-SECTOR
        RET
```

```
NOTFND:   ......              ; Datei nicht vorhanden
SERROR:   ......              ;
MERROR:   ......              ; Fehlerroutinen
RERROR:   ......              ;

DSB:
          DEFB 0              ; Masterlaufwerk
          DEFM 'TEST    '     ; Dateiname
          DEFM 'ABC'          ; Dateityp
          DEFB 0,0,0,0        ; EZ-0
VECTOR:
          DEFW 0
          DEFW 1
          DEFW 3
          DEFW 0

          END START
```

8. Hinweise zur Erstellung von Anwenderprogrammen

8.1 KOS-Parameteraufbereitung

Anwenderprogramme werden unmittelbar von KOS aus geladen und ausgeführt. Der folgende Abschnitt zeigt auf, wie man auf die nach dessen Namen stehenden Eingaben des Parameterfeldes zurückgreifen kann. Zur Illustration dient der Aufruf eines fiktiven Programms TESTPROG, das drei Parameter im P-Feld benötigt.

Diese Parameter heißen:

```
1:DATEI1.ABC
  DAT*.??
  $EATN
```

Der erste Parameter ist vom Typ EDA (eindeutige Dateiadresse), der zweite vom Typ MDA (mehrdeutige Dateiadresse), der dritte Parameter schließlich kennzeichnet einen E/A-Treiber.

Eingabe: TESTPROG DATEI1.ABC DAT*.?? $EATN<---

Vor dem Einsprung in das Anwenderprogramm 'TESTPROG' bereitet KOS automatisch die ersten drei Parameter nach dem Kommandonamen zu Dateispezifikationsblöcken (DSB's) auf. Die Ablage der DSB's erfolgt in jeweils 16 Byte großen Bereichen ab den Adressen:

```
50H : Parameter 1
60H : Parameter 2
70H : Parameter 3
```

Ein jeder dieser 16 Byte großen Blöcke enthält folgende Informationen und entspricht den ersten 16 Byte des für die Dateiverwaltung benötigten DSB.

0	1 . . . 8	9 . . . 11	12	13	14	15
Status-Byte	Name	Typ	0	0	0	0

Das DSB-Statusbyte (Byte 0) beinhaltet in den Bitstellen:

D2-D0: Information über die Mediennummer des Parameters
D4-D3: Information über den Namen/Typ des Parameters
D6-D5: nicht verwendet (immer Null)
D7 : E/A-Treiberkennzeichnung (dem Parameter war ein $-Zeichen vorgestellt) falls gesetzt.

Statusbyteaufschlüsselung:

D2	D1	D0	
0	0	0	keine Mediennummer angegeben (Mastermedium)
0	0	I	Medium 0
0	I	0	Medium 1
0	I	I	Medium 2
I	0	0	Medium 3
I	0	I	-
I	I	0	-
I	I	I	ungültige Mediennummer (hier sind aus Kompatibilitätsgründen zu früheren KOS-Versionen nur Mediennummern von 0 bis 6 zulässig).

D4	D3	
0	0	kein Name/Typ angegeben
0	I	EDA
I	0	MDA
I	I	unzulässige Zeichen in Name oder Typ

Das oben erwähnte Beispiel erzeugt im Bereich 50H - 7FH folgende Einträge: (jeweils hex und ASCII-Code)

```
50H:    09 44 41 54 45 49 31 20 20  41 42 43  0  0  0  0
        .  D  A  T  E  I  1        A  B  C  .  .  .  .

60H:    10 44 41 54 3F 3F 3F 3F 3F  44 3F 20  0  0  0  0
        .  D  A  T  ?  ?  ?  ?  ?  ?  ?     .  .  .  .

70H:    88 45 41 54 4E 20 20 20 20  20 20 20  0  0  0  0
        .  E  A  T  N                       .  .  .  .
```

Fehlende Zeichen im Namen/Typ-Bereich (jeweils Byte 1-11) werden mit Leerzeichen (ASCII-Code: 20H) aufgefüllt. Zusätzlich zu dieser Parameteraufbereitung legt KOS den gesamten Kommandostring nach dem Kommandonamen beginnend mit dem ersten nicht Trennzeichen (im Beispiel: 1) ab Adrese 81H ab (maximal 127 Zeichen). Auf Adresse 80H steht dabei die Anzahl der abgelegten Zeichen. Anwenderprogramme mit mehr als drei Eingabeparametern müssen auf die Information ab Adresse 80H zurückgreifen.

8.2 Programmbeispiel

Ein Anwenderprogramm 'TEST' benötigt einen eindeutig definierten Dateinamen als Parameter. Das folgende Beispiel zeigt die Aufbereitung des DSB aus dem vorbereiteten Parameterblock auf Adresse 50H.

```
START:
       LD HL,50H              ; Adresse des KOS-DSB
       LD A,(HL)              ; Statusbyte lesen
       AND A
       JR Z,NONAME            ; Fehler: kein Name angegeben
       Bit 4,A
       JR NZ,NOEDA            ; Fehler: kein eindeutiger Dateiname
       AND 7                  ; Bits 3 bis 7 ausblenden
       LD (HL),A
       CP 7
       JR Z,INVDN             ; Fehler: Mediennummer unzulässig
                              ; an dieser Stelle ist die Eingabe
                              ; in Ordnung
       .
       .                      ; weiter im Hauptprogramm
       .                      ; Datei eröffnen, etc.
       .
NONAME:...
NOEDA: ...                    ; Fehlerroutinen
INVDN: ...

        END START
```

PSI ψ 80-HARDWARE

Version: 5.3

Stand: 15. August 1981

Das vorliegende Handbuch beschreibt Aufbau und Funktion der Computersysteme der PSI80D-Serie. Die darin enthaltene Information ist für den Anwender vor allem in folgenden Fällen wesentlich:

* Anschluß nicht standardmäßiger Peripherie
* Nachrüsten von zusätzlichen PROM-residenten Programmen (z.B. für Diagnose-Zwecke)
* Erweiterung des Geräts um optionale Hardware (z.B. ECB-Zusatzkarten)
* Erstellung von ASSEMBLER-Software, die direkten Zugriff auf Hardwaregegebenheiten nimmt (z.B. spezielle Ein-/Ausgabe-Treiber, Echtzeitaufgaben, Memory Mapping usw.).

Für andere, nicht hardwarespezifische Aufgaben ist die Verwendung des PSI80-Bedienungshandbuchs voll ausreichend.

INHALTSVERZEICHNIS

1. Übersicht

Diese technische Beschreibung der zentralen Hardware der Computersysteme der Serien PSI80 und PSI80D gilt der Architektur, den Funktionen und den Ausbaumöglichkeiten eines modernen leistungsfähigen Kompaktcomputers.

Es wird von Fall zu Fall auf Schalt- und Bestückungspläne verwiesen, die zum Lieferumfang eines jeden Systems gehören. Außerdem werden Datenblätter und Beschreibungen folgender großintegrierter Schaltkreise verwendet:

ZILOG - Z80A-CPU*	ZILOG - Z80A-SIO*
ZILOG - Z80A-PIO*	ZILOG - Z80A-CTC*
NEC - uPD765	MOTOROLA - MC6845

Diese Bauteile werden im Text zumindest kurz charakterisiert. Im allgemeinen ist eine direkte Programmierung dieser Komponenten nicht notwendig, da sie durch die Betriebs-Software bereits entsprechend ihrer Verwendung im PSI80-System initialisiert sind.

Die Datenblätter dieser Bausteine gehören nicht zum Lieferumfang und sind im Bedarfsfall direkt vom Lieferanten zu beziehen (im Falle der Z80-Bausteine von KONTRON).

Die PSI80-Hardware ist auf einer zentralen Platine aufgebaut. Unterschiedliche Ausbaustufen sind im allgemeinen mit unterschiedlicher Bestückung verbunden. Außerdem können Unterschiede in der Auslegung und Anzahl der peripheren Einheiten mit den unterschiedlichen Versionen verbunden sein. Es wird, soweit erforderlich, in der Beschreibung auf diese Varianten Rücksicht genommen.

Maßgeblich ist in jedem Fall die gültige Produkt-Spezifikation.

PSI80-Systeme sind vom FTZ geprüft:

Zulassungsnummer: C-089/80
Funkentstörgrad: N nach VDE 0875
Modemzulassung: FTZ 02013D PSI80

2. Architektur der Rechnerbaugruppe

Basis der PSI80-Computerserie ist eine hochintegrierte Computerbaugruppe, die eine vollständige Zentraleinheit mit weitreichendem Ausbau umfaßt. Auf dieser Platine sind in der maximalen Ausbaustufe möglich:

- Z80A-CPU* Zentralprozessor,
- Programm- und Datenspeicher (max. 16 kByte PROM, 64 kByte RAM),
- Bildschirmprozessor mit 16 kByte Bildwiederholspeicher,
- Tastaturanschluß,
- Controller für 2 ins Gehäuse integrierte Floppy-Disk-Laufwerke,
- Parallel- und Serien-Ein-/Ausgabe sowie eine
- ECB-Busschnittstelle.

Zusätzliche Ausbaumöglichkeiten bietet ein in der Reihe PSI80(D)/Mx verfügbarer Einschubrahmen für anwendungsorientierte Baugruppen im ECB-Bus-Standard, die Ein-/Ausgabe-bezogen sein können (Z80A-ECB/AE16, Z80A-ECB/O, ...), Funktionserweiterungen darstellen (Z80A-ECB/A), Speichererweiterungen bieten (Z80A-ECB/D32), oder als autonome Subsysteme arbeiten können.

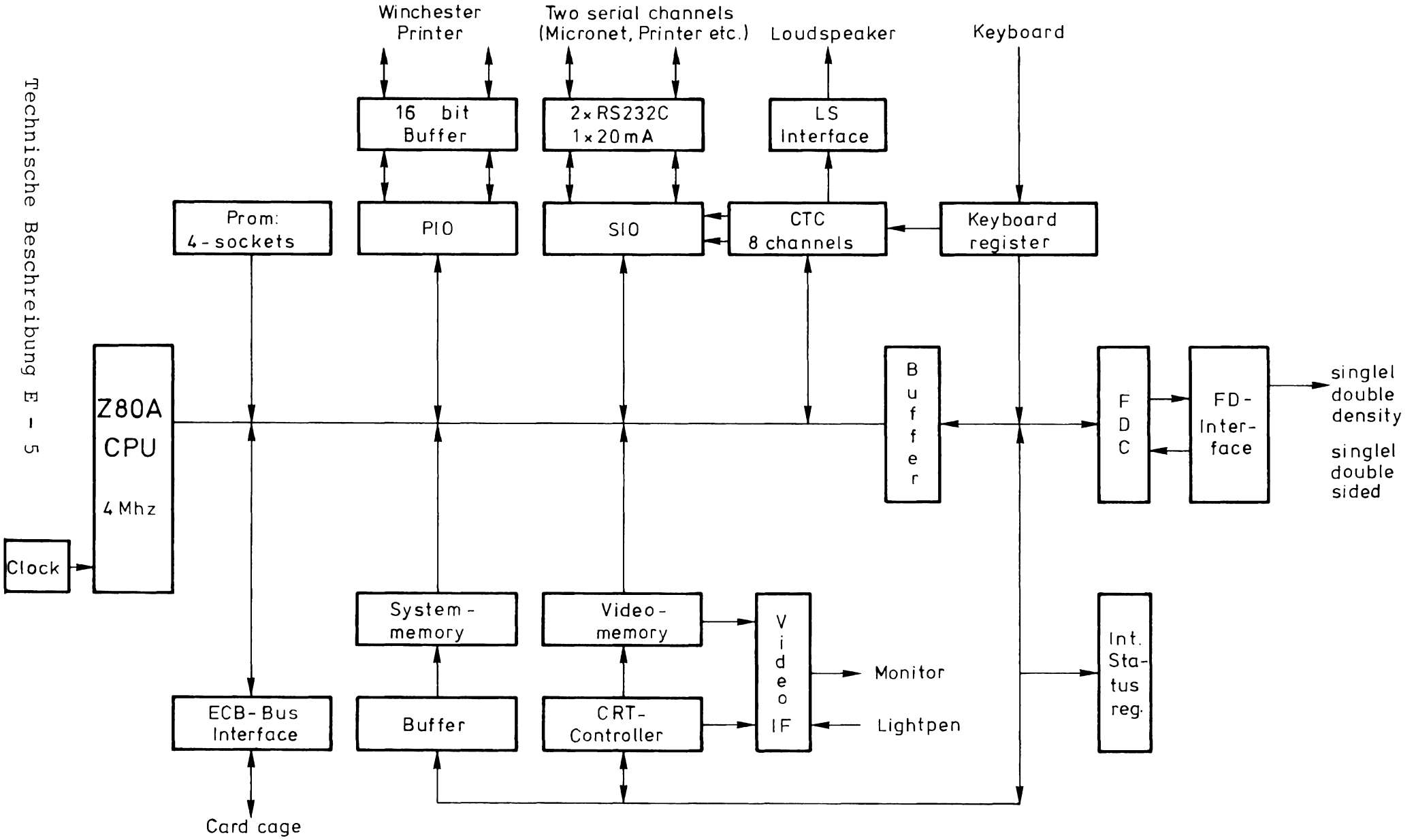
Die Architektur des Rechners entspricht langjähriger Erfahrung auf dem Gebiet der Schaltungsentwicklung von Mikrocomputersystemen. Die Einheiten auf der Grundplatine wurden nach funktionalen, logischen und elektrischen Gesichtspunkten so angeordnet, daß höchste Betriebssicherheit gewährleistet wird. So wurde beispielsweise strikt darauf geachtet, die Busstruktur des Prozessors (Adreß-/Datenbus) auch im Layout aufrechtzuerhalten, um Störeinflüsse wie Übersprechen etc. von vornherein zu minimieren.

Die Aufteilung in einen 'inneren' und 'äußeren' Datenbus begrenzt die kapazitive Belastung in den einzelnen Bereichen und erhöht somit wiederum die Betriebssicherheit. Der 'innere' Datenbus verbindet alle Z80A-Peripheriebausteine und die Ausgänge der Speicher mit der CPU. Störsichere Schmitt-Trigger-Puffer trennen diesen vom 'äußeren' Datenbus, der alle übrigen Peripherieeinheiten und die Eingänge der Schreib-/Lesespeicher bedient. Diese Anordnung erübrigt einerseits Aufwand zur Bussteuerung für Z80A-E/A-Bausteine und ermöglicht andererseits die Verwendung von Speichern mit einer Zugriffszeit nahe der theoretisch oberen Grenze von 375 ns.

Von diesen beiden Bussystemen elektrisch durch störsichere Schmitt-Trigger-Puffer getrennt ist der 'externe Erweiterungsbus' im ECB-Standard (Reihe "Mx").

Die folgende Seite zeigt das Blockschaltbild der zentralen Rechnerbaugruppe.

* Trademarks of ZILOG Inc., Cupertino/CA



Blockdiagram KDT Rev.5

3. Hardwarebeschreibung

3.1 Zentraler Rechnerteil

Der Rechnerteil ist aus folgenden funktionale Blöcken aufgebaut:

- CPU mit Buspuffern
- Festwertspeicher
- Schreib-/Lesespeicher
- Ein-/Ausgabeschnittstellen
- Bildprozessor mit Bildwiederholspeicher
- Floppy Disk-Prozessor *

Die folgenden Abschnitte beschreiben Einzelheiten der Baugruppe.

* nicht bei PSI80T, PSI80/MO

3.1.1 Zentraleinheit

Die CPU besitzt drei verschiedene Bussysteme: Adreß-, Steuer- und Datenbus. Die beiden erstgenannten sind unidirektional, d.h. sie gehen immer von der CPU aus und führen von dort im Prinzip zu allen auf der Baugruppe vorhandenen Funktionsgruppen.

Der Datenbus ist bidirektional, d.h. der Informationsstrom kann sowohl von außen zur CPU (Lesevorgang), als auch umgekehrt (Schreibvorgang) fließen. Zur Steuerung der entsprechenden Richtung und um zu verhindern, daß mehrere Datenquellen gleichzeitig den Bus beanspruchen, ist eine Bussteuerung vorhanden.

Die PSI80-Zentralbaugruppe hat einen zweigeteilten Datenbus. Der 'innere Datenbus' (ID0...ID7) verbindet alle Z80A-Peripheriebausteine, den PROM-Bereich und die Ausgänge sämtlicher Speichereinheiten mit der CPU. Der 'äußere Datenbus' (AD0...AD7) schließlich bedient die Eingänge der Schreib-/Lesespeicher und die übrigen Peripherieeinheiten.

In den Reihen 'Mx' der PSI80-Computerkarte steht für Erweiterungen mit ECB- oder S100-Baugruppen auch ein externer Erweiterungsbus zur Verfügung. Dieser ist über einen eigenen bidirektionalen Puffer mit dem 'inneren' Datenbus verbunden. Der externe Adreß- und Steuerbus ist identisch mit dem internen. Für Erweiterungen steht ein FAN-OUT von etwa 7 TTL-Lasten zur Verfügung (ca. 30 LS TTL-Lasten).

3.1.2 Speicherbereiche

Auf der PSI80-Zentralbaugruppe sind 3 verschiedene Speicherbereiche, sowie die Steuerlogik für einen weiteren externen Speicherbereich (über ECB- oder S100-Bus) mit bis zu sechs Speicherbänken vorhanden.

	Bezeichnung	Kapazität	Speichertyp
1.	Festwertspeicher	max. 16 kByte	EProm (5V Versorgung)
2.	System Schreib-/ Lesespeicher	64 kByte	RAM dynamisch
3.	Bildwiederhol-speicher	16 kByte	RAM dynamisch
4.	externer Speicher	max.192 kByte	beliebig

Zur Auswahl der einzelnen Bereiche stehen über einen Statusport drei Steuersignale zur Verfügung. Folgende Adreßbereiche sind belegt (gültig für Prom **P7-Standardprogrammierung**):

POFF	MAP1	MAP0	J6	PROM	SYSRAM	VID.RAM	Ext.RAM
0	0	0	ein	0-1FFF	2000-FFFF	-	-
			aus	0-3FFF	4000-FFFF		
0	0	1	ein	0-1FFF	2000-7FFF C000-FFFF	8000-BFFF	-
			aus	0-3FFF	4000-7FFF C000-FFFF	8000-BFFF	-
0	1	0	ein	0-1FFF	2000-3FFF	-	4000-BFFF
			aus	0-3FFF	-		
			x		C000-FFFF		
0	1	1	ein	0-1FFF	2000-3FFF	8000-BFFF	4000-7FFF
			aus	0-3FFF	-		
			x		C000-FFFF		
1	0	0	x	-	0000-FFFF	-	-
1	0	1	x	-	0000-7FFF C000-FFFF	8000-BFFF	-
1	1	0	x	-	0000-3FFF	-	4000-BFFF
1	1	1	x	-	0000-3FFF	8000-BFFF	4000-7FFF

Speicheradreßbereiche (x --> beliebig)

Das Verfahren zur Programmierung des Statusports ist im Abschnitt 3.2.2 beschrieben.

3.1.2.1 Festwertspeicher

Die Baugruppe enthält vier Steckplätze für folgende EProm-Typen mit 5Volt Versorgungsspannung:

	J6	J5
i2716	A	A
TMS2532	B	B

Der Adreßbereich reicht in Abhängigkeit von Jumper J6 von 0-1FFFH (8 kByte) bzw. von 0-3FFFH (16 kByte) und ist als 'Nur-Lesespeicher' realisiert. Der Systemspeicher (RAM) im selben Adreßbereich ist als 'Nur-Schreibbereich' zugänglich, während der Festwertspeicher selektiert ist.

3.1.2.2 Schreib-/Lesespeicher (System RAM)

Dieser Bereich umfaßt 64 kByte (4 Bänke je 16 kByte) und ist mit dynamischen RAMs (16 kBit x 1) realisiert. In Abhängigkeit der Statusleitungen POFF, MAP1 und MAP0 stehen unterschiedliche Adreßbereiche als System RAM zur Verfügung. Bei PSI80/M0 ist nur Bank 2 implementiert (16 kByte RAM).

3.1.2.3 Bildwiederholspeicher

Dieser Bereich umfaßt 16 kByte und ist ebenfalls mit 16k x 1 Bit dynamischen RAMs realisiert. Der Adreßbereich liegt zwischen 8000H und BFFFH, falls das Bit MAP0 den Wert 1 hat.

Der Bildwiederholspeicher ist für CPU-Zugriffe transparent. Dies bedeutet, daß Bildprozessor (CRTC 6845) und CPU virtuell gleichzeitig auf diesen Speicherbereich zugreifen können. Zugriffe auf den Bildwiederholspeicher erfordern das entsprechende Setzen der Statusleitung MAP0 (siehe Abschnitt 3.2.2).

3.1.2.4 Externer Speicher (1)

Über den ECB-Bus können bis zu 6 zusätzliche Speicherbänke angeschlossen werden. Zu diesem Zweck werden auf der zentralen Baugruppe 6 verschiedene MBS-Signale (Memory bank select) erzeugt und auf den ECB-Bus geführt.

Externe Speicherbaugruppen müssen eines dieser Signale (MBS0 bis MBS5) als Card select-Signal verwenden. Der Adreßbereich externer Speicher hängt ab von den Statusleitungen MAP0 und MAP1.

Die Kapazität einer einzelnen externen Speicherbaugruppe darf bis zu 32 kByte betragen. Dies entspricht der Standardprogrammierung von Prom 7 (HM 7603).

Busbelegung der MBS-Signale: (2)

MBS0	ECB BUS: 10c
MBS1	12c
MBS2	13c
MBS3	14a
MBS4	23c
MBS5	19c

Diese Signale sind nur dann aktiv, wenn tatsächlich ein Zugriff auf einen externen Speicherbereich stattfindet. Welches der 6 MBS-Signale in einem solchen Fall aktiv wird, hängt vom Inhalt des Mapper Ports (IC13: 74LS137) ab. Dieser kann durch Port-Write Befehle (Adresse: 1AH) entsprechend gesetzt werden.

D7.....D3	D2	D1	D0	Signal
x	0	0	0	MBS0
x	0	0	1	MBS1
x	0	1	0	MBS2
x	0	1	1	MBS3
x	1	0	0	MBS4
x	1	0	1	MBS5
x	1	1	0	MBS6

x ---> beliebig

(1) nur bei PSI80(D)/Mx-Serie
(2) Diese Signale sind bei PSI80(D)-Systemen in der eingebauten Busplatine verdrahtet (siehe Abschnitt: Einschubrahmen). Die Verdrahtung des Card-Select-Signals der 'externen' Speicherkarte auf die gewünschte MBS-Leitung ist Anwender-seitig vorzunehmen.

3.2 Ein-/Ausgabekanäle

3.2.1 PSI80-Ein-/Ausgabebausteine

Die Baugruppe enthält folgende E/A-Bausteine (1):

Adresse (hex)	Baustein	Port	implementiert bei
00	Z80A-DMA	DMA-Port	) DMA-
01	AM 2918	Control Register	) Option
02	-		
03	-		
04	Z80A-SIO/0(/9)	Port A-Data	) PSI80(D)/xx
05	"	Port B-Data	) (1)
06	"	Port A-Control	) PSI80(D)/xx
07	"	Port B-Control	) (1)
08	Z80A-CTC-1	Kanal 0	)
09	"	Kanal 1	) PSI80(D)/xx
0A	"	Kanal 2	)
0B	"	Kanal 3	)
0C	Z80A-PIO	Port A-Data	)
0D	"	Port B-Data	) PSI80(D)/Mx
0E	"	Port A-Control	)
0F	"	Port B-Control	)
10	Z80A-CTC-2	Kanal 0	)
11	"	Kanal 1	) PSI80(D)/xx
12	"	Kanal 2	)
13	"	Kanal 3	)
14	FDC 765	Main Status Reg.	) PSI80(D)/M2, M1
15	"	Data Reg.	)
16	-	-	) PSI80(D)/S2, S1
17	-	-	)
18	CRTC-6845	Adreßregister	) PSI80(D)/xx
19	"	Registerfile	)
1A	IC13 : Mapper Port Strobe		) PSI80(D)/Mx
1B	Prom5 : Software protection PROM Strobe		) optional)
1C	IC40 : Statusport Strobe		) PSI80(D)/xx
1D	IC44 : Keyboard Input Port Strobe		) PSI80(D)/xx
1E	FDC 765: DACK (Data Acknowledge)		) PSI80(D)/M2,M1,
1F	" : TC (Terminal Count)		) S2, S1

(1) zur Beachtung: nicht bei allen PSI80-Versionen implementiert
(siehe Produktspezifikation)

3.2.2 Statusport

Über den Statusport (Baustein IC40: 74LS273) werden alle Steuerleitungen der Baugruppe bedient. Der Statusport ist über die I/O-Adresse 1CH erreichbar.

Achtung:
Dieser Port kann und darf nicht gelesen werden. Der momentane Inhalt des Statusports ist im Betriebssystem KOS in der Speicherstelle 3 abgelegt. Ändern des Statusports durch Anwenderprogramme nur unter Interruptsperre. Gleichen Inhalt in Port und Speicherstelle 3 schreiben! Interruptroutinen des Betriebssystems beeinflussen ebenfalls beide Inhalte.

Bitzuordnung:

ST0	-	MAP 0
ST1	-	MAP 1
ST2	-	Sound Trigger
ST3	-	Video Invert
ST4	-	Alpha/Graphik
ST5	-	Prom off
ST6	-	Standard/Mini
ST7	-	Motor on

Die Funktion der Signale wird in der folgenden Signalbeschreibung dargestellt.

Signalbeschreibung:

MAP0/MAP1 Memory map Signale zur Auswahl interner und externer Speicherbereiche (siehe auch Abschnitt: Speicher)

MAP1	MAP0	System	Video	Extern
0	0	0000-FFFF	-	-
0	1	0000-7FFF C000-FFFF	8000-BFFF	-
1	0	0000-3FFF C000-FFFF	-	4000-BFFF
1	1	0000-3FFF C000-FFFF	8000-BFFF	4000-7FFF

ST2: Sound Trigger — Eine negative Flanke auf dieser Leitung triggert ein Monoflop (IC36: 74LS123) und aktiviert für die Dauer von etwa 0.5 s den Lautsprecherausgang 'SOUND OUT'. Das Monoflop ist nachtriggerbar. Die Frequenz des Tons hängt von Kanal 0 des CTC-2 ab.

ST3: Video Invert — Invertiert den Videoausgang

ST3=0 ---> Hintergrund dunkel
ST3=1 ---> Hintergrund hell

ST4: Alpha/Graph — Umschaltung zwischen alphanumerischer und graphischer Betriebsart

ST4=0 ---> graphische Betriebsart
ST4=1 ---> alphanumerische Betriebsart

ST5: Prom off — Ein-/Ausschaltung des Prombereichs

ST5=0 ---> Prombereich eingeschaltet
ST5=1 ---> Prombereich ausgeschaltet

ST6: STD/Mini — Einstellung des Diskcontrollers auf Standard- oder Minilaufwerke

ST6=0 ---> 8"-Laufwerk
ST6=1 ---> 5 1/4"-Laufwerk

ST7: Motor on — 'Motor On'-Leitung für 5 1/4" Laufwerke

ST7=0 ---> Motor abgeschaltet
ST7=1 ---> Motor eingeschaltet

3.2.3 Tastaturanschluß

Der Tastaturanschluß erfolgt über ein 8bit TTL-Register (IC44: 74LS274) mit Tristate-Ausgängen. Dieses Register kann als Port (Adresse: 1DH) gelesen werden.

Der Takteingang des Keyboard Input Registers wird durch das Strobe Signal der Tastaturelektronik gesteuert. Dieses geht zur Generierung eines Interrupts gleichzeitig an Kanal 1 von CTC-2. Der Anschluß der Tastaturelektronik erfolgt an Stecker ST-E (26-polig).

Signalbezeichnung		Anschluß ST-E	Pegel
DATA	KB0	E-9	TTL-input
DATA	KB1	E-8	"
DATA	KB2	E-7	"
DATA	KB3	E-6	"
DATA	KB4	E-5	"
DATA	KB5	E-4	"
DATA	KB6	E-3	"
DATA	KB7	E-2	"
STROBE	(invertiert)	E-11	"
VCC	+5V	E-13/26	
GND	0V	E-1/14	

Belegung von Stecker ST-E (Tastaturanschluß).

ST-E ist 1:1 verdrahtet auf dem untersten 25-poligen Stecker der Rückwand.

3.2.4 Serienschnittstellen

Die Baugruppe enthält im Vollausbau zwei unabhängige Serienschnittstellen (Kanal A und B) zum Anschluß beliebiger serieller Sender-/Empfängergeräte (Drucker, Terminals, MICRONET/PSI etc.). Beide Schnittstellen werden von einem Z80A-SIO/0 bedient. Folgende Norm-Schnittstellen sind realisiert:

KANAL	Adresse	RS232-C	20mA	RS423	RS422
A	04	ja	ja	optional	optional
B	05	ja	nein	optional	optional

Sämtliche Signale der beiden Schnittstellen sind auf je einem 25-poligen Normstecker herausgeführt (optional 37-polig nach RS 449).

Kanal A ---> ST-F
Kanal B ---> ST-G

Steckerbelegung Kanal A:

Anschluß SIO	Anschluß ST-F	Signalbez.	Pegel
TxDA	F-3 output	Send data	RS232-C
TxDA	F-16 output	Send data	20mA (+)
-	F-10 outp. return	Send data	20mA (-)
DTRA	F-5 output	Data terminal ready	RS232-C
RTSA	F-6 output	Request to send	RS232-C
RxDA (J2-A)	F-2 input	Receive data	RS232-C
RxDA (J2-B)	F-14 input	Receive data	20mA (+)
-	F-9 inp. return	Receive data	20mA (-)
CTSA	F-4 input	Clear to send	RS232-C
DCDA	F-20 input	Data set ready	RS232-C
TxCA (J3-B)	F-15 input	Ext. transmitter clock	RS232-C(*)
TxCA (J4-B)	F-17 input	Ext. receiver clock	RS232-C(*)
-	F-7	Signal Ground	0 Volt
-	F-1	Chassis Ground	0 Volt

Kanal A ist empfängerseitig standardmäßig für RS232C-Pegel ausgelegt (Jumper J2 in Stellung A). Die Auswahl der 20mA-Schnittstelle erfolgt durch Umstecken vom Jumper J2 auf Stellung B.
ST-F ist 1:1 verdrahtet auf dem zweiten (von unten) 25-poligen Stecker der Rückwand.

Steckerbelegung Kanal B:

Anschluß SIO	Anschluß ST-G	Signalbez.	Pegel
TxDB	G-3 output	Send data	RS232C
DTRB	G-5 output	Data terminal ready	RS232C
RTSB	G-6 output	Request to send	RS232C
RxDB	G-2 input	Receive data	RS232C
CTSB	G-4 input	Clear to send	RS232C
DCDA	G-20 input	Data set ready	RS232C
TxRxCB (J1-B)	G-17 input	Ext. rec./trans. clock	RS232C(*)
TxRxCB	G-24 output	Int. rec./trans. clock	RS232C(*)
-	G-7	Signal Ground	0 Volt
-	G-1	Chassis Ground	0 Volt

(*) siehe Abschnitt: Einstellung der Baudrate

ST-G ist 1:1 auf dem dritten (von unten) 25-poligen Stecker verdrahtet.

Durch die Option 'Modemschnittstelle' wird Kanal B als Datenendgerät-Schnittstelle mit folgender Belegung implementiert:

Anschluß SIO	Anschluß (Rückwand)	Signalbezeichnung	Norm
TxD	2	Sendedaten	D1
RxD	3	Empfangsdaten	D2
DTR	20	DE-Einrichtung betriebsbereit	S1.2
RTS	4	Sendeteil einschalten	S2
+12V	23	Hohe Übertragungsgeschwindigkeit einschalten	S4
SYNC (DCD)	6	Betriebsbereitschaft	M1
CTS	5	Sendebereitschaft	M2
DCD (SYNC)	8	Empfangssignalpegel	M5
GND	7	Betriebserde	E2
	1	Schutzerde	E1

Einstellung der Baudraten

Die Baudrate für Kanal A des Z80A-SIO/0 wird entweder von Kanal 2 des Z80A-CTC-1 oder, bei synchronen Übertragungsverfahren von einem extern zugeführten Takt bestimmt. Die Wahl zwischen internem und externem Takt erfolgt mit den Jumpern J3 und J4.

TxCA	RxCA	J3	J4
intern	intern	A	A
extern	intern	B	A
intern	extern	A	B
extern	extern	B	B

Wird interner Takt verwendet, so ist für Kanal 2 von CTC-1 untenstehende Programmiertabelle maßgebend.

Kanal B des SIO-Bausteins erhält den Übertragungstakt entweder von Kanal 1 des CTC-1 (siehe Tabelle), oder von extern für synchrone Anwendungen. Bei Kanal B sind Sender- und Empfängertakt auf dem SIO-Baustein zusammengefaßt.

TxRxCB	J1
intern	A
extern	B

SIO-Takt: CTC-Betr.Art	x16 Zähler	x32 Zähler	x64 Zähler	x16 Zeitgeber
BAUDRATE	CTC-Teilerfaktor			
9600	13	--	--	--
4800	26	13	--	--
2400	52	26	13	--
1200	104	52	26	--
600	208	104	52	--
300	--	208	104	--
150	--	--	208	--
110	--	--	--	142
75	--	--	--	208

Programmiertabelle zur Einstellung verschiedener Baudraten über einen CTC-Kanal.

3.2.5 Parallelschnittstellen und Festplattenanschluß

Die Baugruppe stellt über einen Baustein Z80A-PIO 16 fest verdrahtete Ein-/Ausgänge, sowie 4 Handshake Leitungen zur Verfügung (nur PSI80(D)/M-Reihe).

Die Adressen des PIOs sind:

PORT A - 0CH
PORT B - 0DH

Alle 16 Datenleitungen sind mit nicht invertierenden Schmitt-Trigger Bausteinen gepuffert. Sie sind so angeordnet, daß damit Drucker mit Parallelschnittstelle (Centronic-Schnittstelle), sowie Winchester Laufwerke PSI/WINS angeschlossen werden können. Der Abgriff der Signale erfolgt über den 26-poligen Stecker ST-C. Dieser ist 1:1 auf dem vierten (von unten) 25-poligen Stecker der Rückwand verdrahtet.

PIO-Anschluß	Richtung	Signalname	Steckerbelegung
PA0	Output	STROBE	C-12
PA1	Output	IPRIME	C-11
PA2	Input	FAULT	C-23
PA3	Input	EMPTY	C-22
PA4	Input	BUSY	C-21
PA5	Input	SELECT	C-20
PA6	Output	SPARE	C-25
PA7	Output	direction Port B	C-10
ARDY	Output		C-7
ASTR	Input		C-8
BSTR	Input		C-24
BRDY	Output		C-5
PB0	bidirectional	DATA 0	C-19
PB1	"	DATA 1	C-17
PB2	"	DATA 2	C-18
PB3	"	DATA 3	C-16
PB4	"	DATA 4	C-4
PB5	"	DATA 5	C-3
PB6	"	DATA 6	C-2
PB7	"	DATA 7	C-15
-	-	Vcc (5Volt)	C-13/26
-	-	GND (0Volt)	C-1/14

Die Signalbezeichnung entspricht der weitverbreiteten CENTRONIC-Schnittstelle.

Port A und Port B können durch Ersetzen der Treiberbausteine Anwender-seitig umdefiniert werden.
Port B ist mit bidirektionalen Puffern (IC1: 74LS245) versehen. Ihre Richtung wird durch Bit 7 von Port A festgelegt.

Port A - Bit 7	Richtung IC1
0	Input
1	Output

Alle unidirektionalen Eingänge sind mit 1 kOhm Pull Up-Widerständen versehen.

Die Treiberkapazität aller Ausgänge beträgt ca. 24 mA bei log. Null Pegel.

Die Strobe-Leitung für PORT B ist auch an den CPU/DMA-Sockel (Pin 41) der Baugruppe geführt. Sie dient dort als Ready-Signal für DMA-gesteuerte Datenübertragungen über die Parallelschnittstelle.

Festplattenanschluß (PSI/WINS)

Der Anschluß einer Festplatte an das PSI80(D)-System erfolgt ohne Hardwareänderungen über die Parallelschnittstelle der Zentralbaugruppe. Folgende Leitungen werden verwendet:

PIO-Bit	PSI/WINS-Anschluß	ST-C Anschluß
PA0	Strobe	C-12
PA1	Reset	C-11
PA2	IF Active	C-23
PA3	Ready	C-22
PB0	DATA0	C-19
PB1	DATA1	C-17
PB2	DATA2	C-18
PB3	DATA3	C-16
PB4	DATA4	C-4
PB5	DATA5	C-3
PB6	DATA6	C-2
PB7	DATA7	C-15
	GND	C-1/14

Bitte beachten Sie die mit PSI/WINS zusätzlich mitgelieferte Anschluß-, Inbetriebnahme- und Service-Dokumentation.

3.2.6 Zähler-/Zeitgeber-Kanäle

Die PSI80(D)-Zentralbaugruppe enthält zwei Z80A-CTC Bausteine

CTC-1 - Basisadresse: 08H
CTC-2 - Basisadresse: 10H

Verwendung der Kanäle:

CTC-1 Kanal 0 - Interrupt FD-Controller uP 765
Kanal 1 - Baudrate SIO-Port A
Kanal 2 - Baudrate SIO-Port B
Kanal 3 - frei für Anwender-Verwendung

CTC-2 Kanal 0 - Tongenerator
Kanal 1 - Tastatur Interrupt
Kanal 2 - Vsync Interrupt (Systemtakt für Multitasking)
Kanal 3 - frei für Anwender-Verwendung

3.2.7 Floppy Disk Controller

Die Floppy Disk Controller Schaltung der Baugruppe ermöglicht den Betrieb von Mini- und/oder Standardlaufwerken mit einfacher und/oder doppelter Schreibdichte (single/double density). Die entsprechende hard- und softwaremäßige Festlegung erfolgt fabrikseitig entsprechend der bestellten PSI80(D)-Version.

Das Herz des Controllers ist der Baustein NEC uP 765, der als intelligenter Peripherieprozessor alle wesentlichen Aufgaben der FD-Ansteuerung übernimmt. Vier Laufwerke mit 'single' und/oder 'double head' Ausrüstung können angeschlossen werden.

Dem Baustein uP 765 sind vier E/A-Adressen zugeordnet:

14H - Main Status Register
15H - Data Register
1EH - Data Acknowledge (für DMA basierende Datentransfers)
1FH - Terminal Count

Über Kanal 0 von CTC-1 ist der uP 765 im Z80A-System interruptfähig. Die Leitung DRQ (Data Request) ist zur Steuerung von DMA-gesteuerten Datenübertragungen zwischen uP 765 und Speicher an den CPU/DMA-Sockel der Baugruppe geführt (Pin 43).

Zur Steuerung der Disk Controller Hardware dienen folgende Bits des Statusports:

ST-6 Laufwerkstyp (Umschaltung der Controller Hardware):
0 ---> 8"-Laufwerke
1 ---> 5 1/4"-Laufwerk

ST-7 Motor On (für 5 1/4"-Laufwerke):
0 ---> Motor ausgeschaltet
1 ---> Motor eingeschaltet

Anschluß von Laufwerken

Der Anschluß erfolgt am 34-poligen Stecker ST-D über eine 1:1 Verbindung.

Anschluß ST-D		Signalbezeichnung	Bemerkung
D-2	output	Head Load	
D-4	input	Index	(8"-Laufwerke)
D-6	input	Ready	(falls nicht vorhanden: J11 geschlossen)
D-8	input	Index	(5 1/4"-Laufwerke)
D-10	output	Drive Select 1	
D-12	output	Drive Select 2	
D-14	output	Drive Select 3	
D-16	output	Motor On	
D-18	output	Direction	
D-20	output	Step	
D-22	output	Write data	
D-24	output	Write gate	
D-26	input	Track 0	
D-28	input	Write protect	
D-30	input	Read Data	
D-32	output	Head Select	(für double sided drives)
D-34	output	Drive Select 4	

Alle Ausgänge zum Laufwerk werden von Open Collector Puffern getrieben. Alle Eingänge vom Laufwerk sind mit 150 Ohm Pull Up-Widerständen versehen. Die ungeraden Anschlüsse von ST-D liegen auf Masse.

3.2.8 Video Controller

Der Videocontroller der Baugruppe ist um den Controller CRTC-6845 aufgebaut. Diesem Baustein sind zwei I/O-Adressen zugeordnet.

18H - Adreßregister
19H - Registerfile

Folgende Bits des Statusports dienen zur Steuerung der Video Controller Hardware:

ST4 - Umschaltung zwischen alphanumerischer und graphischer Betriebsart

ST4=0 --> Graphische Betriebsart
ST4=1 --> Alphanumerische Betriebsart

ST3 - Invertierung des gesamten Bildes

ST3=0 --> Hintergrund dunkel
ST4=1 --> Hintergrund hell

Die Invertierung einzelner Zeichen erfolgt im alphanumerischen Modus bei gesetztem achten Bit im ASCII-Code des abzubildeten Zeichens.

Bildwiederholspeicher

Der Bildwiederholspeicher liegt im Adreßbereich von 8000H - BFFFH und kann von der CPU aus durch Memory mapping erreicht werden (siehe Kapitel Speicher).

Der Bildwiederholspeicher ist für CPU-Zugriffe transparent. Dies ermöglicht virtuell einen gleichzeitigen Zugriff von CPU und CRTC. Die Synchronisation von CPU-Zugriffen erfolgt über die WAIT-Leitung der CPU.

Zeichensatz

Der Zeichensatz ist durch PROM8 festgelegt. Einem Zeichen sind 16 Bytes zugeordnet, von denen die ersten 10 jeweils das Bitmuster einer Bildschirmzeile (Matrix 10x8) beschreiben. Die ASCII-Zeichen unter 20H werden vom Bildschirmtreiber $MON ausgewertet. Im zugehörigen PROM-Bereich sind Sonder- und Semigraphik-Zeichen enthalten, die über Codes > 80H angesprochen werden können.

Monitoranschluß

Alle benötigten Signale sind an dem 26-poligen Stecker ST-K herausgeführt.

Anschluß ST-K	Signalbezeichnung
K-2 output	Video
K-3 output	VSTPA (*)
K-4 input	Light Pen Strobe
K-5 output	Composite Video
K-6 input	VSTOP (*)
K-7 output	Horizontal Sync (HSYNC)
K-8 output	Vertical Sync (VSYNC)
K-9 input	ext. Clock in (*)

Alle anderen Anschlüsse liegen auf Masse.

Alle Ein-/Ausgänge mit Ausnahme des Composite Video Signals sind TTL-kompatibel.

Das Signal LIGHT PEN STROBE kann über Kanal 3 von CTC-1 einen Interrupt generieren.

Das composite Video-Signal ist auf eine 75 Ohm BNC-Buchse geführt ('Mx'-Reihe).

(*) nicht verwendet.

3.2.9 Interruptpriorität

Auf der Baugruppe ist folgende Interruptpriorität festgelegt:

1. DMA (falls bestückt)
2. CTC-1
3. SIO
4. CTC-2
5. PIO
6. Einschubrahmen

In der Standardversion ist Kanal 0 von CTC-1 (Disk Interrupt) die höchste Priorität zugeordnet.

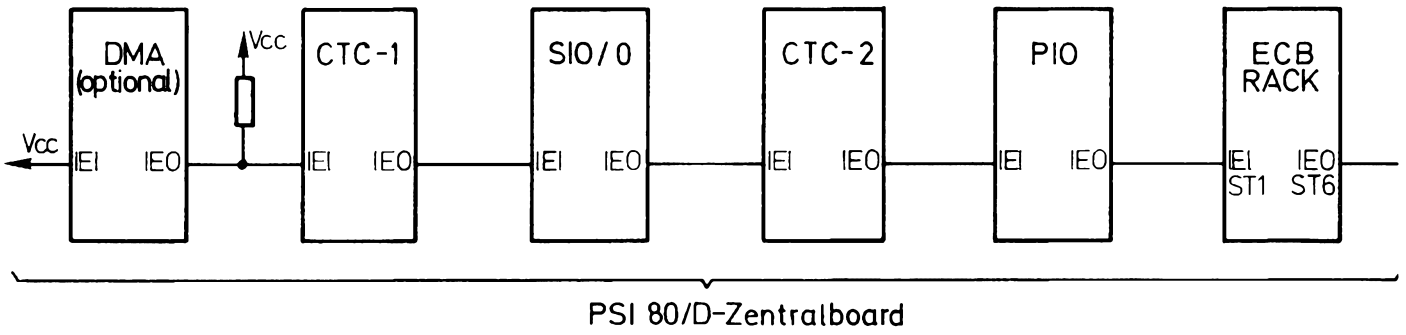
Die Z80A-CPU ermöglicht vektorisierte Interrupts. Dazu stellt die CPU selbst im I-Register die frei in Schritten von 256 Byte wählbare Basisadresse xx00H der aktuellen Interrupttabelle zur Verfügung. Von den Peripheriebausteinen wird in 8 Bits (Format: yyyyyyy0B) einer aus den 128 möglichen Einsprüngen in die Interrupttabelle definiert.

Dazu ist das Vektorregister der Z80A-Peripheriebausteine über Steuerbefehle entsprechend zu setzen.
Die Einträge in der Interrupttabelle werden als Anfangsadressen der zu jedem Interrupt zugehörigen Beantwortungsroutinen (ISR = Interrupt Service Routine) aufgefaßt. Vom Betriebssystem KOS ist der Speicherbereich von

xx00H bis xxFFH

für die Interrupttabelle reserviert. Der Wert xx kann aus dem I-Register der CPU abgelesen werden (KDM-Kommando). Innerhalb der Interrupttabelle sind die Einträge von xxE0H bis xxFFH für das Betriebssystem reserviert.

Bild 2: Interruptpriorität im PSI80/D-System



3.2.10 Lautsprecher- und Resetanschluß

Stecker ST-J dient zum Anschluß des Lautsprechers und des Resetschalters.

J-1	GND	J-6	GND
J-2	GND	J-7	LAUTSTÄRKE POTI (Empfohlener
J-3	RESET-SCHALTER		Wert: 220-470 Ohm)
J-4	GND	J-8	GND
J-5	LAUTSPRECHER	J-9	GND
		J-10	GND

Ein optionales externes Lautsprecherpotentiometer wird dem internen (R 82) parallelgeschaltet.
Bei verschiedenen PSI80-Reihen ist der Lautsprecher ersetzt durch einen Summerbaustein (siehe Produktspezifikation).

4. Zusatzbaugruppen

4.1 CPU/DMA-Adapter

Die CPU/DMA-Adapterplatine (Option PSI/DMA) dient zur Erweiterung der PSI80(D)-Zentralplatine um einen Z80A-DMA Baustein.

Die Platine wird als Piggy Back Baugruppe anstelle der CPU in den 48-poligen CPU/DMA-Sockel der Baugruppe eingesetzt. Hierbei ist bei nachträglicher Montage darauf zu achten, daß der überstehende Teil der Adapterplatine in Richtung Z80A-PIO der zentralen Rechnerplatine zeigt.

Die Baugruppe enthält neben CPU und DMA (I/O-Adresse: 0) einen Multiplexer (74LS153), mit dem 4 verschiedene Eingangssignale auf den Ready Eingang des DMA-Bausteins geschaltet werden können. Der Multiplexer ist softwaremäßig über ein Register (AM 2918) einstellbar. Dieses Register kann als I/O-Port beschrieben und gelesen werden (E/A-Adresse: 01; x = beliebiger Wert):

D7.....D0	DMA-Ready	von
xxxx xx 00	FDC-DRQ	Floppy Disk Controller 765
xxxx xx 01	DMA-Ready	von ECB-Bus Pin 22a
xxxx xx 10	STR-B	von PIO Port B-Strobe
xxxx xx 11	Wait/Ready A	von SIO Wait/Ready Port A

Achtung:
Die Verwendung des DMA-Bausteins wird vom Standardbetriebssystem KOS 5.x nicht unterstützt. Es ist also Sache eines Anwenderprogramms, welche Funktion der DMA-Baustein im System erfüllt.

4.2 RS 422/423-Option

Diese Option für die Serienschnittstellen ist in Vorbereitung. Die genauen Spezifikationen und Anschlußbelegungen sind einem separaten Zusatz zur PSI80-Hardwarebeschreibung zu entnehmen.

4.3 Modem-Option

Diese Option zur PSI80(D)-Serienschnittstelle B setzt die SIO-Schnittstelle auf die Datenendeinrichtungs-Schnittstelle um. Betriebsart: manuell, Betriebsweise duplex, Gleichlaufverfahren asynchron, Übertragungsgeschwindigkeit 300 bit/s (1200 bit/s in Vorbereitung). Zulassungsnummer FTZ 02013D PSI80.

5. Beschreibung des Einschubrahmens

In den Versionen der PSI80(Y)/Mx ist ein zusätzlicher Einschubrahmen im Inneren des Gehäuses hinter dem Sichtschirm vorhanden.

Er ermöglicht den Einsatz von ECB-Karten im Europaformat (160x100 mm). Jeweils 2 nebeneinanderliegende Steckplätze können stattdessen auch eine Karte im Doppel-Europa-Format (230x160 mm) oder - mit dem Busadapterzusatz PSI/S100 - eine Karte im S100-Format aufnehmen.

Träger dieser Karte ist eine gedruckte Schaltung, die den KONTRON ECB-Bus realisiert. Über ein 64-poliges Kabel ist diese Platine mit der zentralen Computerbaugruppe verbunden.

5.1 KONTRON's ECB-Bus Standard

Zu diesem Bus sind Karten für die Realisierung der vielfältigsten Aufgaben von vielen Herstellern aus Westdeutschland, Österreich, Schweiz, Schweden, Niederlande und Frankreich verfügbar.

Wir weisen darauf hin, daß im PSI80 auch dieser Bus mit 4 MHz Taktfrequenz betrieben wird: Es sind somit Karten der Z80A-ECB-Reihe zu verwenden.

Bei externer Speichererweiterung genügt die Klasse -3 (Zugriffszeit 250 ns) für RAM's. Bei PROM's sind im allgemeinen Typen mit einer garantierten Zugriffszeit von 350 ns ausreichend, wenn sichergestellt ist, daß die Umgebungstemperatur außerhalb des Gehäuses 35 Grad C nicht übersteigt.

Über Einzelheiten der ECB-Reihe informieren KONTRON's ECB-Handbücher.

ECB-Bus Pin-Belegung:

	Benennung	Stecker Pin	Bezeichnung
Adreßbus:	A0	5c	Adresse 0
	A1	7c	Adresse 1
	A2	6a	Adresse 2
	A3	6c	Adresse 3
	A4	7a	Adresse 4
	A5	8a	Adresse 5
	A6	9a	Adresse 6
	A7	9c	Adresse 7
	A8	8c	Adresse 8
	A9	30a	Adresse 9
	A10	18c	Adresse 10
	A11	17c	Adresse 11
	A12	27c	Adresse 12
	A13	29a	Adresse 13
	A14	18a	Adresse 14
	A15	28c	Adresse 15
Datenbus:	D0	2c	Data 0
	D1	14c	Data 1
	D2	4c	Data 2
	D3	4a	Data 3
	D4	5a	Data 4
	D5	2a	Data 5
	D6	3a	Data 6
	D7	3c	Data 7
Bank Select:	-MBS0	10c	Memory Bank Select 0
	-MBS1	12c	Memory Bank Select 1
	-MBS2	13c	Memory Bank Select 2
	-MBS3	14a	Memory Bank Select 3
	-MBS4	23c	Memory Bank Select 4
	-MBS5	19c	Memory Bank Select 5

Hinweis: Aktiv-Low-Signale sind mit einem Minuszeichen gekennzeichnet.
Alle Bussignale dürfen mit ca. 7 TTL-Eingängen belastet werden (ca. 30 LS TTL-Lasten).

	Benennung	Stecker Pin	Bezeichnung	
Steuerbus:	-M1	20a	Maschinenzyklus 1	
	-MRQ	30c	Memory Request	
	-IORQ	27a	IN/OUT Request	
	-RD	24c	Read	
	-WR	22c	Write	
	-RFRSH	28a	Refresh	
	-HLT	25c	Halt	
Verschiedenes:	-WAIT	10c	Wait	
	-INT	21c	Interrupt	
	-NMI	20c	non maskable Int.	
	-RESET	31c	Reset (Ausgang)	
	IEI	11c	Int. enable in	
	IEO	16c	Int. enable out	
	-PWRCL	26c	Power on clear	
	CLK	29c	Clock 4.0 MHz (MOS-Pegel)	
	2 x CLK	16a	2 x Clock	(1)
	n x CLK	25a	n x CLOCK	(1) (3)
	-BUSRQ	11a	Busrequest	
	-BUSAK	31a	Busacknowledge	
	BAI	12a	Busprioritäts-steuerung Ein	
	BAO	17a	Busprioritäts-steuerung Aus	
	WRITE EN	26a	Write Enable	
	DPR	23a		
	+5	1a,c	+ 5V	
	GND	32a,c	Ground	
	+12	13a	+ 12V für EPROMs	(2)
	- 5	15a	- 5V für EPROMs	(2)
	+15	19a	+ 15V für V24 und	(2)
	-15	15c	- 15V für AD-Wandler	(2)
	VCMOS	24a	(Notstrom)	

(1) wird von der PSI80-Zentralplatine nicht bereitgestellt.

(2) Nicht mit PSI80-Zentralplatine verbunden, Versorgung vom Netzteil direkt über Lötbrücken auf der Einschubplatine

(3) nicht durchverbunden

5.2 Adressierung im ECB-Bus

Der 'Externe Erweiterungsbus' des Einschubrahmens ist elektrisch ständig an die Busse der PSI80D-Zentralplatine angekoppelt. Baugruppen des Einschubrahmens dürfen bei folgenden Gelegenheiten Daten auf den Bus schalten:

a) Die CPU liest von Ein-/Ausgabeadressen größer oder gleich 20H. Dementsprechend sind auf den Erweiterungskarten Bausteine nur mit Adressen größer/gleich 20H adressierbar.

b) Die CPU liest den Interruptvektor eines externen Peripheriebausteins (siehe Interruptpriorität).

c) Die CPU liest von einem externen Speicher (siehe hierzu Abschnitt: Speicher)

Dies gilt nicht für den Betrieb eines autonomen Subsystems (z.B. MICRONET/PSI). Bei diesem Aufbau entfällt das 64-polige Verbindungskabel zur Zentralplatine und das Subsystem wird z.B. über eine Serienschnittstelle angekoppelt.

Beim Einsatz von ECB-Karten im PSI80 sind folgende Port-Adressen standardmäßig verwendet:

Platine	E/A-Adresse	fest/variabel
Z80A-ECB/A	C0H...C3H, 30H...4FH	fest
Z80A-ECB/B	20H...2FH	fest (für PSI/BASIC)
Z80A-ECB/I	A0H	variabel
Z80A-ECB/X	D0H	variabel
Z80A-ECB/O	E0H	variabel
Z80A-ECB/AE16	8CH	variabel
Z80A-ECB/AA4	40H	variabel

Die E/A-Adressen der ECB/A und ECB/B sind fest eingestellt. Die Baugruppen-Adressen der restlichen Karten wird bei Konfiguration im Werk auf obigen Wert eingestellt.

Bei mehr als 2 Karten gleichen Typs werden die darauf folgenden Adressen eingestellt, sofern dies mit anderen eingebauten Karten vereinbar ist (z.B. 4xZ80A-ECB/I: A0H, B0H, C0H, D0H, falls keine ECB/A zusätzlich eingebaut ist).

5.3 Interruptsteuerung im Einschubrahmen

In Z80A-Systemen sind Interrupts hardwaremäßig durch die fest verdrahtete Daisy chain Kette in ihrer Priorität festgelegt.

Die Priorität im Einschubrahmen ist wie folgt:

- Steckplatz 1 im Einschubrahmen höchste Priorität
- Steckplatz 2 "
- Steckplatz 3 "
- Steckplatz 4 "
- Steckplatz 5 "
- Steckplatz 6 " niedrigste Priorität

Beim Anordnen von interruptfähigen Karten im Einschubrahmen ist darauf zu achten, daß die Durchschleifung von der in der Priorität niedrigsten Platine bis zum Anschlußkabel ununterbrochen ist. Die Durchschleifung auf der Platine des Einschubrahmens ist also auf den eingesteckten Platinen so fortzusetzen, daß alle interruptfähigen Bausteine einbezogen sind.

Die Durchschleifung der Signale IEI/IEO ist zeitkritisch: wie auf der Zentralplatine (siehe Schaltplan) ist bei längeren Ketten eine Look-ahead Logik vorzusehen.

Weitere Informationen sind in Applikationsschriften zur Interruptbehandlung von KONTRON enthalten.

5.4 Stromversorgung des Einschubrahmens

Über einen 7-poligen Stecker werden dem Einschubrahmen alle Spannungen des PSI80-Netzteils zugeführt.

Die Spannungen sind wie folgt belastbar:

	PSI80/M2	PSI80/M1	PSI80/M0
+ 5V	3.0 A	3.3 A	3.6 A
+ 12V	0.2 A	0.7 A	1.2 A
+ 15V	0.2 A	0.2 A	0.2 A
- 15V	0.2 A	0.2 A	0.2 A

Die 5V-Versorgung erfolgt direkt. Die anderen Spannungen sind mit auf der Platine des Einschubrahmens zu schließenden Lötbrücken geführt. Durch Einsetzen eines Festspannungsreglers des Typs 7905 kann aus -15V die für manche Speicherplatinen notwendige -5V-Versorgung realisiert werden. Der Platz dafür ist vorgesehen.

5.5 S100-Adaptermodul

5.5.1 Schaltungsbeschreibung

Die S100-Adapterbaugruppe (Platinennummer: 245) ermöglicht den Einsatz von S100-Baugruppen im Einschubrahmen des PSI80-Systems. Eine S100-Adapterbaugruppe benötigt zwei nebeneinanderliegende Steckplätze für Einfacheuropakarten. Bei nachträglichem Einbau müssen deshalb die beiden mittleren Führungsschienen eines Steckplatzpaares ausgebaut werden.

Neben Adreß- und Datenleitungen werden folgende Steuersignale entsprechend dem S100-Standard verwendet:

S100-Pin	Signal (-=invertiertes Signal)
75	-PRESET
99	-POC
43	SMEMR
45	SOUT
46	SINP
68	MWRITE
78	PDBIN
72	PREADY
76	PSYNC

PSYNC ist durch eine ODER-Verknüpfung aus den Signalen -MRQ und -IORQ abgeleitet.

5.5.2 Speicher- und E/A-Adressierung auf S100-Karten

a) Speicheradressierung

Speicherbereiche auf S100-Karten werden vom System als externer Speicher adressiert (siehe Abschnitt: Speicher), angezeigt durch die Aktivierung eines Memory Bank Select Signals.

Über die Jumper J1 bis J6 kann ausgewählt werden, welches von 6 MBS-Signalen zur Adressierung verwendet werden soll. In jedem Fall liegt der Adreßbereich zwischen 4000H und BFFFH bei ausgeschaltetem Videobereich bzw. zwischen 4000H und 7FFFH bei eingeschaltetem Videobereich. Standardmäßig ist Jumper J1 geschlossen.

b) E/A-Adressierung

E/A-Bausteine auf S100-Karten dürfen Adressen >=20H verwenden. Es ist zu beachten, daß interruptgesteuerter Betrieb von E/A-Bausteinen auf S100-Karten gewöhnlich nicht möglich ist.

5.5.3 Stromversorgung von S100-Karten

S100-Karten werden gewöhnlich mit ungeregelten Spannungen versorgt, da jede Karte mit entsprechenden Gleichspannungsreglern ausgerüstet ist. Ungeregelte Spannungen stehen im PSI80-System nicht zur Verfügung. Es ist deshalb erforderlich, die Spannungsregler einer S100-Karte zu überbrücken und die geregelten Spannungen des PSI80-Systems direkt zu verwenden.

Achtung:

Stecken Sie niemals eine S100-Karte bei eingeschaltetem Gerät ein oder aus.

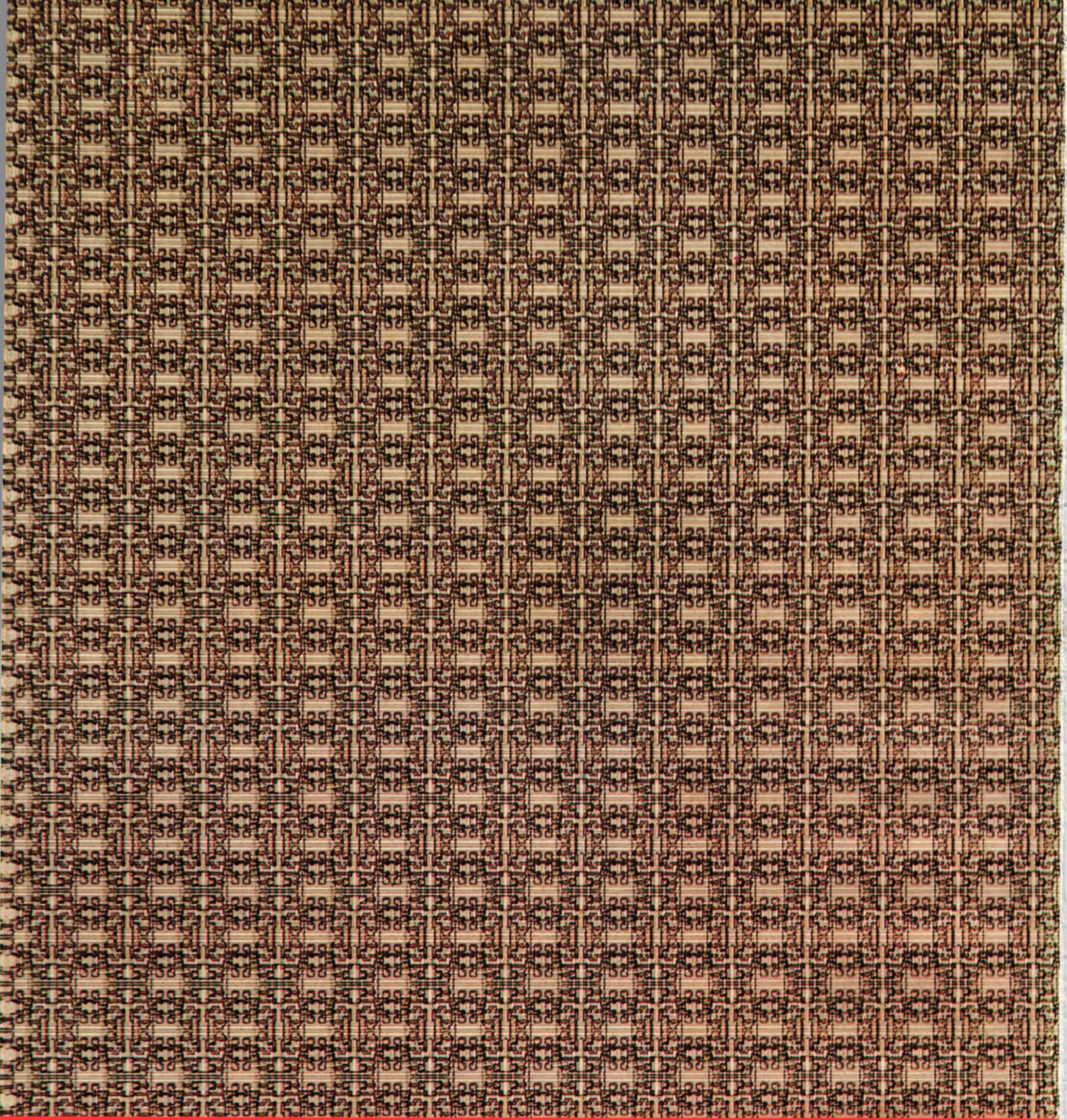
6. Netzteil

Das Netzteil der PSI80-Systeme ist sekundär getaktet (+5V, +12V). Die Nebenspannungen (+15V, -15V, -12V) werden durch Festspannungsregler erzeugt.

Jede Baugruppe im PSI80-System ist über ein eigenes Kabel versorgt.

Die zusätzliche Belastbarkeit ist in den Systemen mit Einschubrahmen vorgesehen. In den anderen Ausführungen der PSI80-Serie sind nur die Spannungen +5V, +12V und -12V garantiert.

Alle Spannungen sind kurzschlußfest und gegen Überstrom und Überspannung geschützt.



KONTRON
MIKROCOMPUTER GMBH

8057 Eching b. München
Breslauer Straße 2
Tel. (0 89) 3 19 01-0
Telex 05 22 122
Telefax (0 89) 3 19 01-311

TECHNISCHE BÜROS:

8057 Eching
Obere Hauptstraße 5
Tel. (0 89) 3 19 01-318
Telex 05 213 671

8500 Nürnberg 21
Rennweg 62
Tel. (09 11) 53 33 06
Telex 06 26 391

7000 Stuttgart 30
Maybachstraße 39 a
Tel. (07 11) 81 46 21
Telex 07 23 061

6000 Frankfurt 70
Kennedy-Allee 34
Tel. (06 11) 63 60 61
Telex 04 14 881

4000 Düsseldorf 1
Ronsdorfer Str. 145
Tel. (02 11) 733 14 53
Telex 08 582 675

3000 Hannover 81
Hermann-Guthe-Str. 3
Tel. (05 11) 83 90 51-57
Telex 09 23 729

2000 Hamburg 70
Königsreihe 2
Tel. (0 40) 6 82 95-0
Telex 02 11 998

1000 Berlin 41
Albrechtstraße 34
Tel. (0 30) 792 30 31-3
Telex 01 85 464

ÖSTERREICH:
Industriestr. B 13, A-2345 Brunn a. Gebirge
Tel. (0 22 36) 8 66 31, Telex 7 9 337

SCHWEIZ:
Bernerstr. Süd 169, CH-8048 Zürich
Tel. 01 62 82 82, Telex 5 2 115

1066 Epalinges, 10, ch. des Croisettes
Téléphone 021 / 33 15 35