

robotron

1

# SOFTWARE HANDBUCH

MIKRORECHNER  
BETRIEBSSYSTEM  
SCP

Das ideale Nachschlagewerk für den Programmierer

Betriebssystem  
SCPX

Das vorliegende Nachschlagewerk entspricht dem Stand Juni 1986.

Nachdruck, jegliche Vervielfältigung oder Auszüge daraus sind unzulässig.

Das Software-Handbuch wurde erarbeitet durch ein Autorenkollektiv der Kammer der Technik im VEB ROBOTRON, Büromaschinenwerk "Ernst Thälmann" Sömmerda.

Dieses Nachschlagewerk wurde mit dem Textprogramm TP erarbeitet und auf dem Mosaikdrucker robotron K6313 ausgedruckt.

Autorenkollektiv: Dr. Ing. Brode, Mathias  
Dipl.-Ing. Hubert, Martin  
Dipl.-Math. Klein. Udo  
Dipl.-Ing.-Oec. Fahr, Klaus

Herausgeber: VEB ROBOTRON Büromaschinenwerk  
"Ernst Thälmann" Sömmerda  
Weißenseer Straße 52  
Sömmerda  
DDR - 5230

Sömmerda 1986

# SOFTWARE - HANDBUCH

für das

## Mikrorechner-Betriebssystem SCP

Das ideale Nachschlagewerk für den Programmierer

Betriebssystem S C P X

VEB ROBOTRON  
Büromaschinenwerk Sömmerda

## Vorwort

Das vorliegende Software-Handbuch für Mikrorechner mit dem Betriebssystem SCP ist als ein ideales Nachschlagewerk für Programmierer in Beruf und Hobby geschrieben.

Dieses Handbuch besteht aus den Teilen

- Betriebssystem SCPX
- BASIC-Interpreter/ - Compiler
- Assemblerprogrammierung
- Tabellen und Arbeitsblätter

und wird ständig erweitert.

Die Nutzung dieses Nachschlagewerkes setzt Grundkenntnisse über das Betriebssystem SCP sowie der unter Steuerung dieses Systems laufenden Software voraus.

Zur Aneignung von Grundkenntnissen und zur Vertiefung des Wissens über SCP-Software wird vom VEB ROBOTRON, Büromaschinenwerk Sömmerda u.a. folgende Dokumentation bereitgestellt:

- Systemhandbuch SCP, Anleitung für den Bediener
- Systemhandbuch SCP, Anleitung für den Programmierer
- Systemhandbuch SCP, Assemblerprogrammierung
- Anwendungsbeschreibung Textprogramm
- Bedienungsanleitung und Sprachbeschreibung BASIC-Interpreter
- Bedienungsanleitung BASIC-Compiler
- Programmieranleitung BASIC
- Bedienungsanleitung und Sprachbeschreibung PASCAL-Compiler.

Autorenkollektiv

## Inhaltsverzeichnis

	<u>Seite</u>
<b>1. SCPX - Kommandos</b>	4
Wie muß ein SCPX-Kommando eingegeben werden?	4
SCPX-Dateispezifikationen	4
Jokerzeichen	5
Zur Syntax	5
Residente SCPX-Kommandos	6
Wahl des aktuellen Laufwerkes	d: 6
Diskettenverzeichnis anzeigen	DIR 6
Datei löschen	ERA 6
Datei umbenennen	REN 7
Textdatei anzeigen	TYPE 7
Arbeitsspeicher in Datei sichern	SAVE 7
Wahl des aktuellen Nutzerbereiches	USER 8
Transiente SCPX-Kommandos	8
beliebige Datei anzeigen	DUMP 8
Datenaustausch	PIP 9
Anzeige und Veränderung des Systemzustandes	STAT 13
Ausführung einer Befehlsfolge	SUBM 15
Erweiterung der SUBM-Funktion	XSUB 16
<b>2. Speicherkonzept</b>	
BIOS	17
BDOS	17
CCP	17
TPA	17
<b>3. Belegung des Verständigungsbereiches</b>	18
<b>4. BIOS - Beschreibung</b>	19
BIOS - Interface	19
Logische Gerätenamen für zeichenweisen E/A	19
Kurzbeschreibung der Funktionen	20
<b>5. Tabellen der Diskettenparameter</b>	22
Aufbau des DPH: (disk parameter header)	22
Aufbau des DPB: (disk parameter block)	23
<b>6. Beschreibung der BDOS-Funktionen</b>	24
Interface	24
Tabelle der BDOS-Funktionen	25
Erläuterung zu den Parametern	27
<b>7. Systemsteuerzeichen bei Pufferedition</b>	29
<b>8. Struktur des Dateisteuerblockes (FCB)</b>	30

## 1. SCPX-Kommandos

Wie muß ein SCPX-Kommando eingegeben werden?

Um ein SCPX-Kommando einzugeben, muß als Antwort auf die SCPX-Meldung eine vollständige Kommandozeile eingegeben werden. Eine Kommandozeile besteht aus dem Kommandowort, ggf. aus Zusätzen und der Abschlußtaste ( z.B. ET1, ET, ENTER, RETURN). Das Kommandowort bestimmt das Kommando, welches ausgeführt werden soll. Die Zusätze enthalten Dateinamen und sonstige Parameter. Buchstaben in einer SCPX-Kommandozeile können als Kleinbuchstaben oder Großbuchstaben eingegeben werden. Bevor eine Kommandozeile von SCPX interpretiert wird, werden alle Buchstaben in große umgewandelt. Eingaben können korrigiert werden, solange die Abschlußtaste noch nicht betätigt wurde. Dazu sind CTRL-Tasten zu verwenden, deren Funktion im Kapitel "Systemsteuerzeichen bei Puffer-Edition" beschrieben ist.

### SCPX-Dateispezifikationen

SCPX identifiziert jede Datei durch einen eindeutigen Namen. Eine Dateispezifikation ist zusammengesetzt aus drei Teilen:

- der Laufwerksspezifikation
- dem Dateinamen
- dem Dateityp.

Der Ausdruck dateispez ist eine Abkürzung für Dateispezifikation und meint eine zulässige Kombination von Laufwerksspezifikation, eigentlichem Dateinamen und Dateityp. In dieser Zusammenfassung werden die folgenden Symbole verwendet, um die Teile von dateispez zu kennzeichnen.

#### d:

steht für die Laufwerksspezifikation. Sie kann ersetzt werden durch "a:", "b:" usw. Wenn diese Angabe weggelassen wird, dann wird auf das aktuelle Laufwerk Bezug genommen.

#### dateiname

steht für den Dateinamen. Er kann aus 1 bis 8 Zeichen bestehen. Als Zeichen sind zugelassen:

A B ... Z

a b ... z

0 1 ... 9

! # \$ % & ' ( ) + - / \ ^ \_ ` @ { } | ~

"\*" und "?" sind Jokerzeichen (siehe dort).

#### .typ

steht für den Dateityp. Er kann aus einem Punkt und 1 bis 3 weiteren Zeichen bestehen. Als Zeichen sind die gleichen wie für dateiname zugelassen. Wird für .typ nichts eingegeben, so werden 3 Leerzeichen angenommen.

Zulässige Kombinationen aus den Elementen von dateispez sind:

- dateiname
- d:dateiname
- dateiname.typ
- d:dateiname.typ

## Jokerzeichen

dateispez die keine "Jokerzeichen" enthalten, sind eindeutige Dateispezifikationen. Enthalten dateispez "Jokerzeichen", so ist die Dateispezifikation mehrdeutig. Sie bezieht sich dann nicht auf eine einzelne Datei, sondern auf eine Dateigruppe.

Die zwei "Jokerzeichen" sind "?" und "\*". Das "?" steht für genau ein beliebiges Zeichen an genau dieser Position. Der "\*" steht für eine beliebige Anzahl von Zeichen ab genau dieser Position bis zum Ende dateiname oder zum Ende .typ.

### Beispiele:

abc?.doc Die Dateien abc1d.doc, abc2d.doc und abc3d.doc entsprechen dieser dateispez; die Dateien abc1d1.doc, abc12.doc und abc11d.doc entsprechen nicht dieser dateispez.

abc\*.doc Die Dateien abc.doc, abc1d1.doc und abc12.doc entsprechen dieser dateispez; die Datei abc12.doc entspricht nicht dieser dateispez.

\*.bak Alle Dateien mit .typ ".bak" entsprechen dieser dateispez.

nicht sinnvoll:

tp\*11.mac "\*" meint alle Zeichen bis zum Ende, d.h., "11" wird nicht benötigt.

### Zur Syntax

- Unterstrichene Begriffe müssen durch einen aktuellen Wert ersetzt werden, z.B. "d:" durch "a:" oder "n" durch "3".
- In unterstrichenen eckigen Klammern stehende Begriffe können ggf. weggelassen werden, z.B. kann für "DIR d:[dateispez]" nur "dir a:" geschrieben werden. Nicht unterstrichene eckige Klammern müssen mit eingegeben werden, z.B. bei PIP-Optionen.
- Begriffe oder Zeichen, die in der Syntax groß geschrieben sind, müssen direkt eingegeben werden. Bei der Eingabe sind Groß- und Kleinschreibung gleichwertig. Für "DIR" kann z.B. "DIR", "dir" oder "Dir" eingegeben werden.
- Das Zeichen "^", gefolgt von einem Buchstaben, bedeutet, daß die Kontroll-Taste und der Buchstabe gleichzeitig zu betätigen sind, z.B. bei der PIP-Option Stext^Z.
- Drei Punkte ("...)" bedeuten, daß die Liste beliebig fortgesetzt werden kann.

## Residente SCPX-Kommandos

### Wahl des aktuellen Laufwerkes

**d:**

Syntax: **d:**

Bedeutung:

Mit diesem Kommando wird das aktuelle Laufwerk ausgewählt. Alle nachfolgenden dateispez ohne Laufwerksangabe beziehen sich auf das mit diesem Kommando eingestellte aktuelle Laufwerk.

Beispiele:

```
A>c:          C als aktuelles Laufwerk anwählen
C>a:          A als aktuelles Laufwerk anwählen
```

### Diskettenverzeichnis anzeigen

**DIR**

Syntax: **DIR** d:[dateiname.typ]

Bedeutung:

DIR gibt die Namen und Typen aller angesprochenen Dateien aus. "Jokerzeichen" sind zugelassen. Dateien mit dem Status "SYS" werden nicht angezeigt.

Beispiele:

```
A>dir          zeigt alle Dateinamen von Laufwerk A an
A>dir b:       zeigt alle Dateinamen von Laufwerk B an
a>dir b:abcde.txt zeigt den Dateinamen "ABCDE.TXT" von Laufwerk
                B an (falls vorhanden)
a>dir a*.mac   zeigt alle Dateinamen an, die mit "A" begin-
                nen und vom .typ ".MAC" sind
A>dir prog???.prn zeigt alle Dateinamen an, die mit "PROG"
                beginnen und maximal drei weitere Zeichen
                sowie den .typ ".PRN" haben
A>dir tp*.*    zeigt alle Dateinamen an, die mit "TP"
                beginnen
```

### Datei löschen

**ERA**

Syntax: **ERA** d:dateiname.typ

Bedeutung:

ERA löscht alle Dateien, auf die sich dateispez bezieht. "Jokerzeichen" sind zugelassen.

Beispiele:

```
A>era b:abcde.txt löscht die Datei "ABCDE.TXT" von Laufwerk B
A>era a*.mac      löscht alle Dateien, die mit "A" beginnen und
                den .typ "MAC" haben
A>era tp*.*       löscht alle Dateien, die mit "TP" beginnen
A>era *.*         löscht alle Dateien
```



## Datei umbenennen

## REN

Syntax: **REN** d:dateiname.typ=dateiname.typ

neuer Name = alter Name

### Bedeutung:

REN gibt einer existierenden Datei einen neuen Namen. "Jokerzeichen" sind nicht zugelassen.

### Beispiele:

A>ren text.dok=text.bak Die Datei "TEXT.BAK" wird umbenannt in "TEXT.DOK"

A>ren b:abc=def Die Datei "DEF" auf Laufwerk B wird umbenannt in "ABC"

*falsch*

A>ren B:abc=c:def alter Name und neuer Name befinden sich auf verschiedenen Laufwerken

## Text-Datei anzeigen

## TYPE

Syntax: **TYPE** d:dateiname.typ

### Bedeutung:

TYPE gibt den Inhalt einer Textdatei zeichenweise auf die Konsole aus. "Jokerzeichen" sind nicht zugelassen.

### Beispiele:

A>type brief.txt

A>type b:beispiel.bs

A>type c:tpi.prn

*nicht sinnvoll*

A>type pip.com keine Textdatei

## Arbeitsspeicher in Datei sichern

## SAVE

Syntax: **SAVE** n d:dateiname.typ

### Bedeutung:

SAVE schreibt den Speicherinhalt in die Datei dateispez. Ab der Speicheradresse 100H (TPA-Anfang) werden n Speicherseiten zu je 100H (256 dezimal) Bytes in die Datei geschrieben. "n" wird dezimal angegeben. "Jokerzeichen" sind nicht zugelassen.

### ACHTUNG!

Existiert bereits eine Datei mit dem Namen dateispez, dann wird diese zuerst gelöscht und dann eine neue Datei angelegt.

### Beispiele:

A>save 3 beispiel.com Speicherinhalt 100H ... 3FFH wird in die Datei "BEISPIEL.COM" geschrieben.

A>save 10 b:a1.com Speicherinhalt 100H ... 0AFFH wird in die Datei "A1.COM" geschrieben.

## Wahl des aktuellen Nutzerbereiches

## USER

-----

**Syntax:** USER n

### Bedeutung:

USER stellt den aktuellen Nutzerbereich auf n ein. Für n sind dezimale Werte von 0 ... 15 zugelassen. Vom SCP wird standardmäßig der Nutzerbereich 0 als aktueller Nutzerbereich eingestellt. Mit STAT USER kann die Nummer des aktuellen Nutzerbereiches abgefragt werden.

### Beispiel:

```
A>user 2           stellt den Nutzerbereich 2 ein
A>user 15          stellt den Nutzerbereich 15 ein
```

## Transiente SCPX-Kommandos

### Beliebige Datei anzeigen

### DUMP

-----

**Syntax:** DUMP d:dateiname.typ

### Bedeutung:

DUMP gibt den Inhalt einer beliebigen Datei auf die Konsole aus. "Jokerzeichen" sind nicht zugelassen. Die Bytes der Datei werden sequentiell als HEX-Zeichen ausgegeben.

### Beispiele:

```
A>dump brief.txt
A>dump b:beispiel.bas
A>dump c:stat.com
```

**Syntax:** PIP pipkommando  
 oder  
 PIP

Das PIP-Kommando kann auch erst nach Aufruf des Programms eingegeben werden.

pipkommando allgemein

ziel=quelle[option],quelle[option],...

PIP kann Quelldatenströme verketteten und zu einem Zieldatenstrom zusammenfassen. Als Quelle können Dateien oder Eingabegeräte, als Ziel eine Datei oder ein Ausgabegerät angegeben werden. Links vom Gleichheitszeichen steht das Ziel. Rechts können eine oder mehrere Quellen angegeben werden. Zu jeder Quelle können in eckigen Klammern beliebig viele Optionen angegeben werden. Die Reihenfolge der Optionen ist beliebig. Zwischen mehreren Optionen können Leerzeichen stehen. Für Ziel und Quellen kommen dateispez oder Gerätenamen in Frage.

### Dateien kopieren ohne Umbenennen

pipkommando

- (1) d:dateiname.typ=x:[option]
- (2) x:=d:dateiname.typ[option]
- (3) d:dateiname.typ=d:dateiname.typ[option]

x: ist eine Laufwerksspezifikation, die im Gegensatz zu d: immer angegeben werden muß.

### Bemerkungen:

Der Name der Zieldatei darf kein "Jokerzeichen" enthalten. Der Name der Quelldatei darf "Jokerzeichen" enthalten. Es wird dann sequentiell jede Datei der Quellgruppe einzeln in jeweils eine Zieldatei gleiches Namens geschrieben. Sollen Quelle und Ziel auf dem selben Laufwerk stehen, so ist nur (3) zugelassen.

Beispiele zu (1)

```
*a:stat.com=b:
*stat.com=a:          nur wenn aktuelles Laufwerk nicht A
*dok1.txt=b:[z]      nur wenn aktuelles Laufwerk nicht B
falsch:
*a:dok1.txt=a:[z]    bei gleichen Laufwerken nur (3) zugelassen
*a:dok1.txt=[z]      x: muß immer angegeben werden
*a:*.*=b:           im Ziel sind keine "Jokerzeichen" zugelassen
```

Beispiele zu (2)

```
*a:=b:stat.com
*a:=dok1.txt[uz]     nur wenn aktuelles Laufwerk nicht A
falsch:
*a:=a:*.txt[uz]     bei gleichen Laufwerken nur (3) zugelassen
*=b:*.*
```

Beispiele zu (3)

```
*a:dok1.txt=b:dok1.txt[z]
*dok1.txt=dok1.txt[u z]
```

---

**Eine Datei kopieren mit umbenennen**

Syntax für pipkommando:

d:dateiname.typ=d:dateiname.typ[option]

**Bemerkung:**

Quelle und Ziel dürfen keine "Jokerzeichen" enthalten.

**Beispiel:**

\*a:test2.mac=a:test1.mac            Auf Laufwerk A wird unter dem Namen "TEST2.MAC" eine Kopie der Datei "TEST1.MAC" angelegt.

**Verkettung mehrerer Dateien**

Syntax für pipkommando:

d:dateiname.typ=dateispez1[option],dateispez2[option],...

**Bemerkungen:**

Zieldatei und Quelldateien dürfen keine "Jokerzeichen" enthalten. PIP geht im Standardfall davon aus, daß Textdateien verkettet werden sollen. Als Ende einer Quelldatei wird ^Z (1AH) erkannt. Dieses Steuerzeichen wird nur von der letzten Quelle mit kopiert. Sonderbehandlung mit der Option "o".

**Beispiel:**

\*dneu.txt=d1.txt,d2.txt,d3.txt    Die Datei "DNEU.TXT" entsteht aus den Verkettungen der Dateien "D1.TXT", "D2.TXT" und "D3.TXT".

**Verwendung logischer Gerätenamen**

Syntax für pipkommando:

ausgabegeraet=dateispez

oder

ausgabegeraet=dateispez1,dateispez2,...

oder

ausgabegeraet=eingabegeraet

oder

ausgabegeraet=liste aus eingabegeraeten und dateispez

oder

dateispez=eingabegeraet

oder

dateispez= liste aus eingabegeraeten und dateispez

**Bemerkungen:**

In dateispez sind keine "Jokerzeichen" zugelassen. Sinnvollerweise kann bei der Verwendung von Geräten als Quelle oder Ziel nur mit Textdateien gearbeitet werden, da als Dateiende ^Z (1AH) erkannt wird. Außerdem ist zu beachten, daß Gerätetreiber u.U. nur mit 7 Bit arbeiten. Sollen Daten verkettet werden, so sind die Bemerkungen bei "Verketteten mehrerer Dateien" zu beachten.

**ausgabegeraete**

con: Ausgabe auf Consol-Gerät (i.a. Bildschirm)

lst: Ausgabe auf List-Gerät (i.a. Drucker)

pun: Ausgabe auf Puncher

prn: Ausgabe auf List-Gerät mit:

- Tabulator auf jeder 8. Stelle

- Seitenvorschub (FF als 0CH) alle 60 Zeilen

Zeilennumerierung

out: Ausgabe auf ein Nutzergerät; dazu muß PIP geändert werden. Im PIP ist Platz reserviert, wo sich Treiber nachrüsten lassen.

**eingabegeraete**

con: Eingabe von der Console (i.a. Tastatur)

rdr: Eingabe vom Reader

inp: Eingabe von einem Nutzergerät; dazu muß PIP geändert werden. Im PIP ist Platz reserviert, wo sich Treiber nachrüsten lassen.

eof: fiktives Eingabegerät; liefert genau ein Endekennzeichen "^Z" (1AH)

nul: fiktives Eingabegerät; liefert 40 mal NUL (00H)

**Beispiele:**

\*dateil.txt=rdr: Vom Reader werden alle Daten, bis einschließlich ^Z (1AH), in die Datei "DATEI1.DAT" geschrieben.

\*lst:=text1,text2,text3 Die Dateien "TEXT1", "TEXT2" und "TEXT3" werden nacheinander auf das Listgerät ausgegeben.

-----

\*lst:=con: Die über die Konsole (Tastatur) eingegebenen Zeichen werden bis ^Z (1AH) unmittelbar dem Listgerät (Drucker) angeboten.

### PIP Optionen

#### Bemerkungen:

Zu jeder Quelle (dateispez oder eingabegerät) können eine oder mehrere Optionen in "[ ]" angegeben werden (bezugnehmend auf die vorhergehende Datei). Die Reihenfolge der Optionen ist innerhalb der Klammern beliebig. Sie können durch Leerzeichen getrennt werden.

**B** liest Blöcke bis ^S (z.B. schnelleres Lesen vom Lochstreifen)

**Dn** kopiert nur die Spalten 1 bis n einer Textzeile

**E** kopierte Daten werden zusätzlich auf die Konsole ausgegeben (Echo)

**F** Steuerzeichen für Seitenvorschub (FF als 0CH) werden nicht mit kopiert

**Gn** kopiert aus Nutzerbereich n

**H** prüft, ob die zu kopierenden Daten dem HEX-Format entsprechen

**I** läßt im HEX-Format alle Aufzeichnungen unberücksichtigt, die mit "00" eingeleitet werden

**L** wandelt große Buchstaben in kleine Buchstaben um

**N** fügt Zeilennummern, beginnend mit 1, in die Kopie ein

**O** überträgt den gesamten Dateiinhalt ohne Prüfung des Endekennzeichens ^Z (1AH); ist nur beim Verketteten von Dateien nötig; bei ".COM"-Dateien wird ^Z immer ignoriert.

**Pn** fügt alle n Zeilen einen Seitenvorschub (FF als 0CH) ein

**Qtext^Z** kopiert nur bis einschließlich "text"

**R** erlaubt die Übertragung von Dateien mit dem Status "SYS"

**Stext^Z** kopiert ab "text"

**Tn** setzt Tabulatorweite auf n; Tabs werden in entsprechend viele Leerzeichen umgewandelt

**U** wandelt alle kleinen Buchstaben in große Buchstaben um

**V** Zieldatei wird nochmals mit den Quelldaten verglichen

**W** erlaubt Dateien vom Status "R/O" zu überschreiben

**Z** löscht das höchstwertige Bit in jedem kopierten Zeichen

Syntax:     **STAT**  
               **STAT** argumente

#### Bemerkungen:

Benötigte argumente müssen in der CCP-Kommandozeile eingegeben werden. "Jokerzeichen" sind zugelassen.

Wenn STAT aufgerufen wird, sollte das entsprechende Laufwerk nicht den Status R/O haben (ggf. ^C eingeben). Die Anzeigen können sonst unkorrekt sein.

**stat** liefert Angaben über den Schreibschutz der bisher angesprochenen Laufwerke und den freien Speicherplatz auf den zugehörigen Disketten.

#### **stat** d:

liefert Angaben über den Schreibschutz des angegebenen Laufwerkes und den freien Speicherplatz auf der Diskette.

#### **stat** d:dateiname.typ

liefert einen Überblick über die angegebenen Dateien, den von ihnen belegten Speicherplatz und den auf der Diskette noch verfügbaren Speicherplatz. "Jokerzeichen" sind zugelassen.

Beispiele:

A>stat b:\*. \*     STAT wird vom aktuellen Laufwerk gelesen und gibt Informationen über alle Dateien von Laufwerk B.

A>b:stat \*.com   STAT wird vom Laufwerk B gelesen und gibt Informationen über alle ".COM"-Dateien vom aktuellen Laufwerk.

**stat** d:dateiname.typ \$R/O

**stat** d:dateiname.typ \$R/W

**stat** d:dateiname.typ \$SYS

**stat** d:dateiname.typ \$DIR

setzt alle angesprochenen Dateien in den angegebenen Zustand.

R/O - setzt Schreibschutz

R/W - löscht Schreibschutz

SYS - Systemdatei (Datei wird bei DIR nicht angezeigt)

DIR - Nutzerdatei (Datei wird bei DIR angezeigt)

#### **stat** d:dateiname.typ \$\$

zeigt zusätzlich noch die Nummer des letzten Satzes der Datei an (size). Bei sequentiellen Dateien stimmt die Größe mit der Anzahl der belegten 128 Byte Sätze (recs) überein. Bei direkten Dateien kann RECS wegen eventueller Lücken kleiner sein als SIZE.

#### **stat** d:DSK:

liefert eine tabellarische Übersicht über die Eigenschaften der Disketten.

**stat USR:**

liefert eine Übersicht über die auf der Diskette vorliegenden Nutzerbereiche und gibt die Nummer des aktuellen Nutzerbereiches an.

**stat d:=R/O**

setzt für das spezifizierte Laufwerk bis zum nächsten Warm- oder Kaltstart den Schreibschutz.

**stat VAL:**

liefert eine Tabelle der STAT-Befehle und -Befehlsargumente.

**stat DEV:**

liefert eine Tabelle, in der der jedem logischen Kanal zugewiesene Sub-Kanal aufgeführt ist. Den einzelnen logischen Kanälen können folgende logische Sub-Kanäle zugewiesen sein:

Kanal	Sub-Kanäle
CON:	TTY: CRT: BAT: UC1:
LST:	TTY: CRT: LP1: UL1:
PUN:	TTY: PTP: UP1: UP2:
RDR:	TTY: PTR: UR1: UR2:

**stat kanal=subkanal[,kanal=subkanal,...]**

weist den Kanälen die entsprechenden Sub-Kanäle zu.

**Beispiel:**

A>stat con:=crt:,lst:=lpt:



**Syntax:** SUBM d:dateiname par1 par2 ...

**Bedeutung:**

Die in der Kommandodatei dateiname.SUB stehenden CCP-Kommandozeilen werden nacheinander aufgerufen. Die Kommandozeilen können formale Parameter (\$1...\$9) enthalten. Diese formalen Parameter werden mit einer einfachen Zeichenkettensubstitution durch die aktuellen Parameter par1 ... par9 ersetzt. Beim Abarbeiten der Kommandodatei wird jede Kommandozeile auf der Konsole angezeigt. In dieser Meldung sind die formalen Parameter schon substituiert (ersetzt).

**Bemerkungen:**

- Die Kommandodatei muß den .typ ".SUB" haben.
- In der Kommandodatei muß jede Kommandozeile mit Zeilenende abgeschlossen sein.
- Nach der letzten Kommandozeile darf kein Zeichen, auch kein Leerzeichen, in der Datei stehen.
- Beim Aufruf von SUBM muß das aktuelle Laufwerk A sein! Auf diesem Laufwerk wird die Zwischendatei "\$\$\$.SUB" angelegt.
- Jede beliebige Tastenbetätigung bricht die Kommandofolge beim Aufruf des nächsten Kommandos ab.
- Kommandozeilen, die mit ";" beginnen, werden nur ausgegeben (Kommentarzeilen).
- Kommandodateien dürfen nur aus Textzeichen und Zeilenende bestehen. Deshalb sollte eine Kommandodatei im Textprogramm immer wie eine Programmdatei und nicht wie eine Textdatei behandelt werden.
- Leerzeichen in der Kommandodatei sind nicht zulässig.

**Beispiel:**

```
Die Kommandodatei "TEST.SUB" hat folgenden Aufbau:
;Verzeichnisinhalt von Laufwerk A anzeigen
;Datei $1 von Laufwerk A nach Laufwerk B kopieren
;Datei $1 auf Laufwerk A löschen
;Verzeichnisinhalt von Laufwerk B anzeigen
dir a:
pip b:=a:$1
era a:$1
dir b:
;Ende der Kommandodatei
```

Starten der in der Kommandodatei abgelegten Befehlsfolge

```
A>subm b:test beispiel.txt
```

Voraussetzungen:

- auf Laufwerk A befinden sich die Programme SUBM, PIP, STAT, BEISPIEL.TXT;
- auf Laufwerk B befindet sich das Programm TEST.SUB

**falsch**

```
B>a:subm test beispiel.txt
```

--> A muß aktuelles Laufwerk sein

Syntax: XSUB

**Bedeutung:**

Mit SUBM können lediglich CCP-Kommandos aus einer Kommandodatei abgearbeitet werden. Wenn XSUB in einer Kommandodatei verwendet wird, akzeptieren auch Programme alle Tastatureingaben aus der Kommandodatei. Der Eingabedatenstrom ist hiermit vollständig von der Tastatur auf die Kommandodatei umgelenkt. Es existieren jedoch auch Programme, die unter XSUB nicht arbeiten. Wird XSUB benötigt, so muß es das erste Kommando in einer Kommandodatei sein. Um nach Abarbeiten der Kommandodatei sicher den ursprünglichen Systemzustand wieder herzustellen, sollte ein Warmstart ausgelöst werden (ggf. ^C eingeben).

**Beispiel:**

Aufbau der Kommandodatei "TEST.SUB"

```
xsub
pip
b:=a:$1
b:=a:$2
b:=a:$3
dir b:
;Ende der Kommandodatei
```

Der Aufruf der Kommandodatei kann z.B. so erfolgen:

```
A>subm a1.dat a2.dat a3.dat
```

## 2. Speicher - Konzept

Nach dem Laden des Programmes SCPX von der Systemdiskette (Kaltstart) ist der zur Verfügung stehende 64 KByte-Speicher in 5 Bereiche aufgeteilt.

----- Verständigungsbereich -----	Adresse: 0000H
TPA -----	Adresse: 0100H
CCP -----	( C300H bei SCP )
BDOS -----	( D900H )
BIOS -----	Adresse: FFFFH (64K - 1)

**BIOS:** Basic-I/O-System  
Dieser Systembereich enthält die hardwarespezifischen Programmteile. Am Beginn steht eine Sprungtabelle, die zu den einzelnen Routinen zur Bedienung der im System eingebundenen physischen E/A-Geräte führt.

**BDOS:** Basic Disk Operating System  
Dieser Teil des Systems bietet dem Programmierer alle die Routinen an, die den Datenaustausch mit den logischen Geräten gewährleisten. Zur Nutzung dieser Routinen (BDOS-Rufe) gibt es ein standardisiertes Verfahren zu deren Aufruf.  
BDOS bedient sich der physischen BIOS-Routinen.

**CCP:** Consol Command Prozessor  
Nach dem Kaltstart bzw. Systemneustart übernimmt CCP die Steuerung des Betriebssystems. In diesem Systemteil sind die residenten Kommandos DIR, REN, TYPE, ERA, SAVE und USER realisiert, die den TPA nicht verändern. Außerdem können mit CCP auch die transienten Funktionen aufgerufen werden. Diese Programme werden in den TPA geladen. Dann übergibt CCP die Steuerung an die transienten Programme. Nach Abarbeitung wird die Steuerung an das CCP-Programm zurückgegeben. Das erfolgt durch Systemneustart oder über RET aus dem transienten Programm.

**TPA:** Transient Program Area  
Dies ist der Speicherbereich, der den transienten Kommandos und Programmen zur Verfügung steht und bei der Adresse 0100H beginnt. Ab dieser Adresse werden die transienten Kommandos oder Programme geladen, die dann auch bei 0100H gestartet werden.  
Dieser Bereich steht ebenfalls für den Anwenderprogrammierer zur Verfügung.

### 3. Belegung des Verständigungsbereiches

Die erste Seite des Hauptspeicherbereiches (Adressenbereich 0000 - 00FFH) ist als Verständigungsbereich eingerichtet.

Dieser Bereich dient der Verständigung zwischen dem Betriebssystem und den transienten Programmen, die in den TPA geladen und abgearbeitet werden. Zwischen beiden werden Parameter ausgetauscht und Zustandsvariablen eingetragen. Außerdem sind hier Speicherbereiche reserviert, die aufgrund der Hardwarebedingungen der Software nicht zur Verfügung stehen.

Adressen	Inhalt
00H - 02H	Sprungbefehl zum Warmstarteintrittspunkt (WBOOT). Damit ist eine einfache Reinitialisierung des Programmes oder des Betriebssystems möglich.
03H	E/A-Byte zur Zuordnung der Subkanäle für die zeichenweisen E/A-Geräte.
04H	enthält das aktuelle Standardlaufwerk und den Nutzerbereich.
05H - 07H	enthält einen Sprung nach BDOS. Damit können die BDOS-Funktionen aufgerufen werden. Außerdem steht in 0005H/0007H die aktuelle Adresse Ende-TPA + 1.
08H - 2FH	reserviert für die Verzweigungen von RESTART 1 bis 5. Bereich wird vom System nicht genutzt.
30H - 37H	Ansprungstelle für RESTART 6 (z.Zt. ungenutzt).
38H - 3AH	Ansprungstelle für RESTART 7. Wird vom Debugger genutzt zum Umschalten vom Echtzeitmode in den Debuggermode an gesetzten Unterbrechungspunkten.
3BH - 3FH	nicht benutzt, aber reserviert.
40H - 4FH	von CCP und BDOS nicht benutzt; kann für BIOS-Eintragungen genutzt werden.
50H - 5BH	nicht benutzt, aber reserviert.
5CH - 6BH	Kopfzeile des FCB1. Eintragung wird vom CCP vorgenommen, dient der Parameterübergabe an transientes Programm.
6CH - 7BH	Kopfzeile des FCB2. Eintragung von CCP als Parameterübergabe des 2. Parameters.
7CH - 7FH	Bereich des vollständigen Standard-FCB. Bereich steht transientem Programm zur Verfügung.
80H - FFH	Standard-Diskettenpuffer für einen SCP-Satz Bereich kann vom transienten Programm genutzt werden. Beim Aufruf des transienten Programmes wird vom CCP die Zeichenkette der Parameter der CCP-Kommandozeile eingetragen.

## 4. BIOS-Beschreibung

BIOS beinhaltet den geräteabhängigen Programmteil des Betriebssystems mit den E/A-Treibern und den Initialroutinen. Der Eintritt ins BIOS erfolgt über einen Sprungvektor, der am Anfang des Programmes steht. Über diesen Sprungvektor gelangt man an die auch dem Anwender zugänglichen Unterprogramme.

Sprungvektor:

<u>Offset</u>	<u>Fkt</u>	<u>Name</u>	<u>Funktion</u>
+0000H	00H	boot	Initialisierung des Systems
+0003H	01H	wboot	Reinitialisierung (Warmstart)
+0006H	02H	const	Statusabfrage für Konsoleneingabe
+0009H	03H	conin	Eingabe 1 Zeichen von Konsole
+000CH	04H	conout	Ausgabe 1 Zeichen auf Konsole
+000FH	05H	list	Ausgabe 1 Zeichen auf Listgerät
+0012H	06H	punch	Ausgabe 1 Zeichen auf LB-Stanzer
+0015H	07H	reader	Eingabe 1 Zeichen von LB-Leser
+0018H	08H	home	Initialisierung FD-Laufwerk
+001BH	09H	seldsk	FD-LW selektieren
+001EH	0AH	settrk	Spur adressieren
+0021H	0BH	setsec	SCP-Satz adressieren
+0024H	0CH	setdma	Satzpuffer im Speicher adressieren
+0027H	0DH	read	adressierten Satz in Puffer lesen
+002AH	0EH	write	adressierten Satz aus Puffer schreiben
+002DH	0FH	listst	Statusabfrage des Listgerätes
0030H	10H	sectran	Transformation der Satznummer
+0033H		ENDE DES BEREICHES	

BIOS-Interface:

Eintritt: - CPU-Register C: 1. Parameter 8 Bit  
- CPU-Register BC: 1. Parameter 16 Bit  
- CPU-Register DE: 2. Parameter 16 Bit  
- BIOS-Funktion mit UP-Sprung aufrufen  
- Sprungvektor erreichbar über Verständigungsbereich:  
auf Adresse 0001/2H steht Adresse WBOOT des Vektors

Austritt: - Verlassen des aufgerufenen UP  
- CPU-Register A: Parameter 8 Bit  
- CPU-Register HL: Parameter 16 Bit  
- die anderen CPU-Register sind i.a. verändert

3 Gruppen von Unterprogrammen:

- Initialisierung des Systems
- zeichenweise Ein-/Ausgabe
- Diskettenoperationen

Logische Geräte für zeichenweise E/A:

- CON : Dialog mit Bediener
- LIST : Listgerät (hard-copy)
- PUNCH : sequentielle Zeichenausgabe
- READER : sequentielle Zeicheneingabe

Zuordnung der Subkanäle über E/A-Byte (0003H):

bit	Kanal	Subkanäle			
7/6	LIST	TTY:	CRT:	LPT:	UL1:
5/4	PUNCH	TTY:	PUN:	UP1:	UP2:
3/2	READER	TTY:	RDR:	UR1:	UR2:
1/0	CON	TTY:	CRT:	BAT:	UC1:
Bitbelegung		0/0	0/1	1/0	1/1

Die Zuordnung der Treiber zu den Subkanälen ist abhängig von der BIOS-Installation.

Kurzbeschreibung der Funktionen:

- BOOT Initialisierung des Systems
- Kaltstart der Treiber
  - Voreinstellung Stapel und Verständigungsbereich
  - Übergang zum Warmstart
  - INP : ohne
  - OUT : ohne
- WBOOT Re-Initialisierung des Systems
- Normalisierung Stapel
  - Einstellung Standard-LW und Nutzer-Nr. aus VB
  - Einstellung der Adresse Ende-TPA und WBOOT-Adresse im VB
  - INP C: Standard-LW aus 0003H
  - OUT : ohne
- CONST - Statusabfrage Konsoleneingabe
- INP : ohne
  - OUT A: FFH - Zeichen liegt an  
00H - kein Zeichen
- CONIN Eingabe 1 Zeichen von Konsole
- INP : ohne
  - OUT A: Zeichen (8-Bit)
  - wartet auf Zeicheneingabe
- CONOUT - Ausgabe 1 Zeichen auf Konsole
- INP C: Zeichen (8-Bit)
  - OUT : ohne
- LIST Ausgabe 1 Zeichen auf Konsole
- INP C: Zeichen (8-Bit)
  - OUT : ohne
- PUNCH sequentielle Ausgabe 1 Zeichen
- INP C: Zeichen (8-Bit)
  - OUT : ohne
- READER Sequentielle Eingabe 1 Zeichen
- INP : ohne
  - OUT A: ASCII-Zeichen (8-Bit)

HOME Initialisierung des selektierten LW  
- INP: ohne  
- OUT: ohne

SELDSK Selektion des LW (Voreinstellung)  
- INP C : LW# (00...0FH für LW A...P)  
- OUT HL: Adresse des entspr. DPH  
- HL:=0000 : log. LW existiert nicht  
- Anzahl der log. LW ist BIOS-abhängig

SETTRK Selektion der Spur (Voreinstellung)  
- INP BC: Spurnummer  
- OUT : ohne

SETSEC Selektion des SCP-Satzes (Voreinstellung)  
- INP BC: Satz-Nummer  
- OUT : ohne  
- für die Ordnung der Satz-Nummern existieren keine Vorschriften

SETDMA Satzpuffer adressieren  
- INP BC: Adresse Satzpuffer  
- OUT : ohne  
- korrespondierender Bereich von 80H Bytes im Arbeitsspeicher für read-/write-Operationen  
- Initialbereich 80H - FFH

READ Lesen 1 Satz (80H Bytes) von Diskette  
- INP : ohne  
- OUT A: 00H - Operation erfolgreich  
01H - Fehler  
- Quelle ist der mit SELDSK, SETTRK, SETSEC adressierte Satz auf der Diskette  
- Ziel ist der mit SETDMA adressierte Puffer  
- Wiederholungen bei Fehler sind im BIOS zu realisieren

WRITE Schreiben 1 Satz auf Diskette  
- INP : ohne  
- OUT A: 00H - Operation erfolgreich  
01H - Fehler  
- Quelle ist der mit SETDMA adressierte Puffer  
- Ziel ist der mit SELDSK, SETTRK, SETSEC adressierte Satz auf der Diskette  
- Wiederholungen bei Fehler sind im BIOS zu realisieren

LISTST Statusabfrage des Listgerätes  
- INP : ohne  
- OUT A: 00H - Gerät noch besetzt  
FFH - Gerät frei

SECTRAN Transformation der Satzadresse in die SCP-Satznummer für die Funktion SETSEC  
- INP BC: alte Satzadresse (0...SPT-1)  
DE: XLT-Adresse  
- OUT HL: transformierte Satzadresse  
- Funktion wird vom BDOS automatisch aufgerufen  
- XLT befindet sich in DBP

## 5. Tabellen der Diskettenparameter

Diese Tabellen sind Bestandteil des BIOS und beschreiben die Organisation der Disketten zur Benutzung im SCP.

Es existieren bis zu 16 logische Laufwerke, die durch die Kennbuchstaben A...P adressiert werden. Im System kann davon eine beliebige Anzahl installiert und den physischen Laufwerken wahlfrei zugeordnet werden. Die Eigenschaften jedes dieser logischen Laufwerke und deren BDOS-typischen Parameter werden in einer laufwerksspezifischen DPH-Tabelle (disk parameter header) eingetragen. Jeder DPH besteht standardmäßig aus 16 Bytes, auf die vom BDOS zugegriffen wird. Jeder BIOS-Nutzer (auch BDOS) erhält bei Ausführung des BIOS-Rufes SELDSK als Rückmeldung die Adresse des dem selektierten Laufwerk zugeordneten DPH. Wird als Adresse 0000H übergeben, ist das angesprochene Laufwerk nicht installiert.

Innerhalb des DPH wird eine Adresse dpb eingetragen. Diese Adresse verweist auf eine Tabelle DPB (disk parameter block), durch welche die logische und physische Charakteristik des Laufwerkes beschrieben wird.

Aufbau des DPH: (disk parameter header)

<u>Offset</u>	<u>Name</u>	<u>Bedeutung</u>
+00H	XLT	Adresse des Transformationsvektors zur Umwandlung der logischen in die Physikalische Satznummer. Wenn XLT:=0000, dann wird keine Transformation durchgeführt.
+02H	0000	vom System reserviert
+04H	0000	vom System reserviert
+06H	0000	vom System reserviert
+08H	DIRBUF	Adresse für einen internen Satzpuffer (Länge 128 Bytes), den BDOS-Rufe zur Arbeit mit dem Verzeichnis benötigen.
+0AH	DPB	Adresse des Disketten-Parameterblockes
+0CH	CSV	Adresse des Prüfsummenvektors der Verzeichnis-Sätze
+0EH	ALV	Adresse des Blockbelegungsplanes der Diskette
+10H	ENDE DES DPH	



Aufbau des DPB: (disk parameter block)

<u>Offset</u>	<u>Name</u>	<u>Bedeutung</u>
+00H	SPT	Anzahl SCP-Sätze pro Spur
+02H	BSH	Blockverschiebefaktor Funktion der Blockgröße BG: wenn $BG := 128 * 2^n$ Bytes, dann $BSH := n$
+03H	BLM	Blockmaske abhängig von BG und DSM: wenn $BG := 128 * 2^n$ Bytes, dann $BLM := 2^n - 1$
+04H	EXM	Extentmaske abhängig von BG und DSM: wenn $BG := 1024 * 2^m$ , dann für $DSM < 256$ ist $EXM := 2^m - 1$ für $DSM > 255$ ist $EXM := 2^{(m-1)} - 1$
+05H	DSM	Anzahl der Blöcke, die die Diskette enthält, ist $DSM+1$
+07H	DRM	Anzahl der möglichen Verzeichnis-Eintragungen in den durch AL0/AL1 reservierten Blöcken ist $DRM+1$
+09H	AL0	Initialwert des 1. Bytes des Blockbelegungsplanes zur Reservierung der Verzeichnisblöcke (Bit 7 gesetzt reserviert Block #0 usw.)
+0AH	AL1	Initialwert des 2. Bytes des Blockbelegungsplanes
+0BH	CKS	Länge des Prüfsummenvektors $CKS := (DRM+1)/4$
+0DH	OFF	Anzahl der reservierten Systemspuren der Diskette
+0FH	ENDE DES DPB	

## 6. Beschreibung der BDOS-Funktionen

Das Diskettenbasisbetriebssystem (BDOS) stellt den Kern des Betriebssystems SCPX dar. Es unterstützt die Ein- und Ausgabe auf den logischen Geräten (Konsole, Listgerät, sequentielle Datenkanäle) sowie die Dateiarbeit auf den Disketten.

Über BDOS-Ruf können folgende Aufträge von BDOS angefordert werden:

- Zeichenorientierte Ein- und Ausgabe
  - . Zuordnung der logischen zu den physischen Ein- und Ausgabekanälen
  - . zeichenweise Ein- und Ausgabe über die Konsole
  - . zeichenweise Ein- und Ausgabe über die sequentiellen Datenkanäle
  - . zeichenweise Ausgabe über den Listkanal
  
- Arbeit mit Diskettendateien
  - . allgemeine Dateimanipulationen und -verwaltung
  - . sequentieller Dateizugriff
  - . direkter Dateizugriff

Systemverwaltung

- . Initialisierung
- . Ermittlung des Systemzustandes
- . weitere Hilfsfunktionen

### Interface

Der BDOS-Ruf erfolgt über einen UP-Aufruf der absoluten Speicheradresse 5 im Verständigungsbereich. An dieser Stelle befindet sich ein Sprungbefehl, der bei der Systeminitialisierung eingetragen wird. Dieser Sprungbefehl setzt im BDOS-Körper auf. Die Rückkehr aus dem BDOS wird dort selbst durch einen RET-Befehl realisiert. Der BDOS-Ruf muß noch durch Belegung von Registern spezifiziert werden.

Die BDOS-Funktionen sind mit laufenden Nummern versehen. Im CPU-Register C muß vor Ausführung des Rufes diese Kodenummer eingestellt werden, die der gewünschten BDOS-Funktion zugeordnet ist.

Die zusätzlichen Eingabeparameter werden

- bei 8-Bit-Werten in das CPU-Register E,
- bei 16-Bit-Werten in das CPU-Doppelregister DE

eingetragen. Nach Ausführung der BDOS-Funktion wird das rufende Programm hinter der Aufrufstelle fortgesetzt. Ausgabeparameter nach Abarbeitung der BDOS-Funktion werden als

- 8-Bit-Werte im CPU-Register A,
- 16-Bit-Werte im CPU-Register HL,
- Feldeintragungen in einem Feld, das durch CPU-Doppelregister DE vor Aufruf adressiert wurde,

bereitgestellt.

Das Betriebssystem benutzt einen eigenen Stapel, so daß das aufrufende Programm nur mit einer Stapelebene durch den Aufruf belastet wird. Alle CPU-Register werden durch Bearbeitung einer BDOS-Funktion verändert.

**Tabelle der BDOS-Funktionen:**

Nr.	Bezeichnung	Eingang	Ausgang
0	Warmstart	- - -	- - -
1	Konsoleneingabe mit Konsolenecho	- - -	- - -
2	Konsolenausgabe	E: Zeichen (mit ^S, ^P)	
3	READER - Eingabe	- - -	A: Zeichen
4	PUNCH - Ausgabe	E: Zeichen	- - -
5	LIST - Ausgabe	E: Zeichen	- - -
6	direkte Konsolen E/A	E = FF; Status	A: Konsolenstatus (*2)
7	IOBYTE abfragen	- - -	A: IOBYTE
8	IOBYTE belegen	E: IOBYTE	- - -
9	Zeichenkette ausgeben auf Konsole	DE: -> Kette (EKZ = \$, mit ^S, ^P)	- - -
10	Eingabe Konsolpuffer	DE: -> Puffer (*3)	Pufferinhalte (*4)
11	Konsolenstatus	- - -	A: Konsolenstatus (*2)
12	Version ermitteln	- - -	HL: Version
13	Diskettensystem zurücksetzen	- - -	A: Batchmode-Flag (*9)
14	Auswahl Bezugs-LW	E: LW# (0...F) für A...P	- - -
15	Datei eröffnen	DE: -> FCB	A: DC (*7), = FF: Datei nicht vorhanden
16	Datei schließen	DE: -> FCB	A: DC (*7), = FF: Datei nicht vorhanden
17	erste Eintragung suchen	DE: -> FCB	A: DC (*7), = FF: Datei nicht vorhanden
18	folgende Eintragung suchen	- - -	A: DC (*7), = FF: Datei nicht vorhanden

Nr.	Bezeichnung	Eingang	Ausgang
19	Dateien löschen	DE: -> FCB	A: DC (*7), = FF: Datei nicht vorhanden
20	nächsten Satz lesen	DE: -> FCB	A: EC (*5), ≠ 00: EOF
21	nächsten Satz schreiben	DE: -> FCB	A: EC (*5), ≠ 00: Diskette voll
22	Datei erzeugen	DE: -> FCB	A: DC (*7), = FF: Verzeichnis voll
23	Datei umbenennen	DE: -> FCB	A: DC (*7), = FF: Datei nicht vorhanden
24	Abfrage angeschlossener LW	- - -	HL: LW-Vektor
25	Abfrage Bezugs-LW	- - -	A: LW# (0 - 15)
26	Datenpuffer adressieren	DE: -> Datenpuffer	- - -
27	Adresse von ALLOC (ALF) ermitteln	- - -	HL: -> ALLOC
28	Schutz des Bezugs-LW	- - -	- - -
29	Abfrage geschützte LW	- - -	HL: LW-Vektor
30	Dateiattribute setzen	DE: -> FCB	A: DC (*7), = FF: Datei nicht vorhanden
31	Adresse des DPB ermitteln	- - -	HL: -> DPB des selektierten LWs
32	Nutzernummer abfragen/setzen	E = FF: Abfrage E ≠ FF: Setzen mit Wert von E	A: Nutzer# (0 - 15)
33	direkt adressierten Satz lesen	DE: -> FCB	A: RC (*6)
34	direkt adressierten Satz schreiben	DE: -> FCB	A: RC (*6)
35	Dateigröße berechnen	DE: -> FCB	Eintrag FCB
36	Berechnung der aktuellen Satzadresse	DE: -> FCB	Eintrag in FCB + 33, 34, 35: r0-2 = fkt(ex,cr)

Nr.	Bezeichnung	Eingang	Ausgang
37	ausgewählte Laufwerke zurücksetzen	DE: LW-Vektor	A: 00
40	direkt adressierten Satz schreiben und Block initialisieren	DE: -> FCB	A: RC (*6)

### Erläuterungen zu den Parametern

- Die Zeichen "->" sind zu interpretieren als "Zeiger auf".
- Das Zeichen "≠" ist zu interpretieren als "ungleich".
- \*: Verweis auf nachfolgende Bemerkungen.

### BDOS-Interface:

Eingang: CPU-Register C : BDOS-Funktionsnummer  
CPU-Register DE: Feldadressen, Vektoren  
E : Eingabezeichen  
UP-Ansprung : 0005H

Ausgang: CPU-Register A : Status, Zeichen  
CPU-Register HL: Vektoren, Adressen

### Fußnoten:

- \*1: Sonderzeichen  
^H: Rückschritt  
CR: Wagenrücklauf  
LF: Zeilenschaltung  
^I: Tabulation auf nächste 8. Stelle  
^S: start/stop Konsolenausgabe  
^P: List-Echo
- \*2: Status = 00 kein Zeichen im Konsoleneingabepuffer  
≠ Zeichen im Konsoleneingabepuffer
- \*3: Steuerzeichen  
siehe Systemsteuerzeichen bei Pufferedition
- \*4: Struktur des Puffers  
DE: +0
- |     |    |    |    |    |     |      |
|-----|----|----|----|----|-----|------|
| DE: | +0 | +1 | +2 | +3 | ... | +n+1 |
|     | mx | nx | Z1 | Z2 | ... | Zn   |
- mx - Kapazität des Puffers (ist voreinzustellen)  
nx - Füllstand des Puffers  
Z - Zeichen im Puffer
- \*5: Fehler-Code  
A = 00 : Operation in Ordnung  
≠ : Operation nicht ausführbar

- \*6: Direktzugriffs-Code
  - A = 01 : Lesen ungeschriebener Daten
  - = 03 : Extent-Wechsel nicht möglich
  - = 06 : Zugriff auf EOD-Satz der Datei
  
- \*7: Verzeichnis-Code
  - A = FF : Fehler
  - = 0,1,2,3 : Nummer des Eintrages im aktuellen Verzeichnissatz
  
- \*8: Aufbau des Datenbeschreibers für Umbenennung
  - FCB +0 - +15 : Datei alt (LW = 0,1, ... 16)
  - +16 - +31 : Datei neu (LW = 0 oder LW-alt)
  
- \*9: Batchmode
  - A = 00 : keine Batchmodedatei
  - A = FF : Batchmodedatei vorhanden

#### Aufbau der LW-Vektoren

Vektoren im CPU-Doppelregister

```

bit 0 : LA A
      1 : LW B
      : : .. .
      15 : LW P
  
```

## 7. System-Steuerzeichen bei Pufferedition

Die BDOS-Funktion #10 dient zur Eingabe einer Kommandozeile über die Konsole. Neben den transienten Programmen benutzt auch CCP diesen Funktionsaufruf, um Kommando- und Parametereingabe über die Konsole zu ermöglichen. Die eingegebenen Zeichen werden auf der Konsole als Echo ausgegeben. Damit ist eine Eingabe im Dialog möglich. Zur besseren Handhabung werden folgende Steuerzeichen berücksichtigt:

<u>Zeichen</u>	<u>Reaktion</u>
<DEL>	löscht das zuletzt eingegebene Zeichen und zeigt das durch Echo des gelöschten Zeichens auf der Konsole an.
^R	beendet die Anzeige der gerade eingegebenen Zeile mit "#" und protokolliert den tatsächlichen Pufferinhalt auf der nächsten Kommandozeile.
<BS>	(auch ^H) löscht das zuletzt eingegebene Zeichen sowohl auf der Konsole als auch im Puffer.
^U	löscht die gesamte Zeile im Puffer und kennzeichnet das durch Ausgabe von "#" auf der Konsole. Die Eingabe kann neu begonnen werden; das Echo erscheint in der folgenden Kommandozeile.
^X	löscht die eingegebene Zeile sowohl im Puffer als auch auf der Konsole; Eingabe kann neu begonnen werden.
^E	beginnt neue Zeile auf der Konsole. Zeichen wird nicht in den Puffer übernommen, sondern steuert nur das Konsolenbild.
<CR>	(auch ^M) schließt die Puffereingabe ab.
<LF>	(auch ^J) beendet ebenfalls den BDOS-Ruf.
<HT>	(auch ^J) gibt Zahl von Leerschritten auf der Konsole aus, die zum Erreichen der nächsten Spalte im 8er-Raster (1,9,17,25,...) notwendig ist.
^S	laufende Ausgabe, bis nächste Konsoleneingabe unterbrochen.
^P	schaltet den LIST-Kanal parallel zum CON-Kanal. Bedingung bleibt gültig bis zum nächsten ^P.
^C	bricht das gesamte Programm ab und verzweigt zur Reinitialisierung des Betriebssystems durch Ansprung des Warmstartprogrammes (WBOOT). Diese Funktion wird nur aktiv, wenn dieses Zeichen als 1. Zeichen in den Puffer eingegeben wird.

## 8. Struktur des Dateisteuerblockes (FCB)

Bei Zugriff auf Dateien ist es notwendig, im Arbeitsspeicher einen Dateisteuerblock anzulegen. Dieser Steuerblock wird bei BDOS-Rufen vom SCP herangezogen und verwaltet. Der Block besteht aus einem Feld von 33 Bytes (bei sequentielltem Zugriff) bzw. von 36 Bytes bei wahlfreiem Zugriff. Bei den BDOS-Rufen ist die Basisadresse des Blockes mit zu übergeben.

<u>OFFs</u>	<u>Symbol</u>	<u>Inhalt</u>
+00H	dr	Laufwerk-Codierung dr= 0: Bezugslaufwerk 1: LW A ..... 16: LW P
+01H	n1-8	Dateiname in ASCII Dateiattribute in den 8 Bits n1'-n8' n1'-n4': für den Nutzer reserviert n5'-n8': nicht genutzt, aber reserviert
+09H	t1-3	Dateityp in ASCII Dateiattribute in den 8 Bits t1'-t3' t1' = 1/0: read only / read-write t2' = 1/0: SYS-Datei / DIR-Datei t3' = : nicht genutzt, aber reserviert
+0CH	ex	laufende Nummer des Dateiteils (Extent) ex := 00...1FH
+0DH	s1-2	reserviert für internen Systemgebrauch (ggf. s2 = 0000 vor BDOS-fkt.#15,17,18,22 setzen)
+0FH	rc	Satzzähler im Extent rc := 00...80H
+10H	d00-15	Verzeichnis der für den zugehörigen Extent reservierten SCP-Blöcke
+20H	cr	Nummer des Satzes im Extent, der beim nächsten sequentiellen Zugriff im Eingriff ist
+21H	r0-2	(wird nur bei wahlfreiem Zugriff benutzt) Satznummer bezüglich der gesamten Datei
+24H	ENDE DES BEREICHES	



# robotron

**VEB Robotron Büromaschinenwerk**  
**»Ernst Thälmann« Sömmerda**  
Weißenseer Straße 52  
Sömmerda  
DDR – 5230

Exporteur:  
**Robotron Export-Import**  
Volkseigener  
Außenhandelsbetrieb  
der Deutschen  
Demokratischen Republik  
Allee der Kosmonauten 24  
Berlin  
DDR – 1140