robotron

Systemunterlagendokumentation

MOS K 1520

Anwenderdokumentation

Anleitung für den Programmierer Teil I

unter dem Betriebssystem SCPX 1526

Systemunterlagen- Anwenderdokumentation MOS

dokumentation K1520

Stand: 5/86

Anleitung fuer den Programmierer Teil I

unter dem Betriebssystem SCPX 1526

VEB Robotron Buchungsmaschinenwerk

Karl-Marx-Stadt

Die vorliegende 3. Auflage der Dokumentation entspricht dem Stand Mai 1986 und unterliegt nicht dem Aenderungsdienst.

Die Dokumentation wurde durch ein Kollektiv des VEB Robotron Buchungsmaschinenwerk Karl-Marx-Stadt ausgearbeitet.
Im Interesse einer staendigen Weiterentwicklung werden die Leser gebeten, dem Herausgeber ihre Vorschlaege bzw. Hinweise zur Verbesserung mitzuteilen.

Anmerkung:

Die Anwenderdokumentation zum SCP-System besteht 5/86 aus:

- Anleitung fuer den Bediener SCP 1520
- Anleitung fuer den Programmierer SCP 1520
  Teil I und Teil II (Sprachbeschreibung ASM)
- Anleitung fuer den Systemprogrammierer SCP 1520
- Hardwarebeschreibung
- Anwenderdokumentation BASIC-Interpreter
- Anwenderdokumentation BASIC-Compiler
- Anwenderdokumentation C-Compiler
- Anwenderdokumentation PASCAL/M2
- Anwenderdokumentation FORTRAN
- Anwenderdokumentation Textverarbeitungssystem TP
- Anwenderdokumentation Installierungs-Programm fuer TP
- Anwenderdokumentation KOMBO-Druck
- Schulungshandbuch fuer das Textverarbeitungssystem TP
- Anwenderdokumentation Datenbanksystem REDABAS
- Anwenderdokumentation Datenerfassungssystem DAT
- Anwenderdokumentation Kalkulationsprogramm KP
- Zusatzsoftware I
- Zusatzsoftware II
- Zusatzsoftware III

VEB Robotron-Buchungsmaschinenwerk
Karl-Marx-Stadt
Software-Zentrum

9010 Karl-Marx-Stadt
Postschliessfach 129

III-12-12 Kv 1824/86

Inhaltsverzeichnis

1. Einleitung

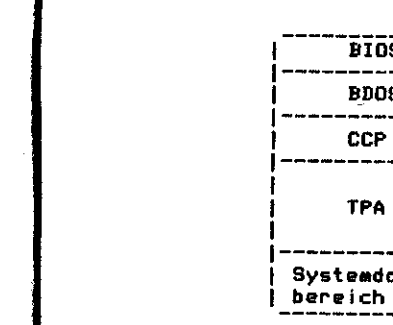
Das Erarbeiten von Anwenderprogrammen fuer ein CP/M -kompatibles Betriebssystem hat vor allem dann Bedeutung, wenn nicht auf das breite Spektrum von Standardsoftware zurueckgegriffen werden kann, die Standardsoftware zu modifizieren (installieren) ist oder Sonderprobleme (z.B. aufgrund nicht standardmaessiger externer Geraete) zu loesen sind.
Der zur Verfuegung stehende Assembler (und Binder) gestattet das komfortable Programmieren auf Maschinencode-Niveau. Dies ermoeglicht effektivste Programme bezueglich Laufzeit und Speicherbedarf. Es ist vor allem wichtig bei der Steuerung von E/A-Geraeten. Voraussetzung fuer den Programmierer sind entsprechende Hardware-Kenntnisse, und das Einarbeiten in die Assemblerprogrammierung erfordert einen relativ hohen Lernaufwand.
Wesentlich einfacher und schneller sind Programme in der hoeheren Sprache "BASIC" zu schreiben. Die Laufzeit und der Speicherbedarf sind schlechter bzw. groesser als bei Assemblerprogrammen, welches aber nicht ueberbewertet werden darf. Rechenprozesse und Standard-E/A-Aufgaben sind mit wesentlich geringerem Programmieraufwand loesbar.
Hoehere Sprachen sind vor allem geeignet, Anwendersoftware kompatibel fuer andere Rechner zu gestalten, sofern entsprechende Compiler bzw. Interpreter zur Verfuegung stehen.

Voraussetzung fuer die Programmierung ist die Kenntnis der "Anleitung fuer den Bediener". Die "Anleitung fuer den Programmierer" beschreibt nur die zum Grundpaket SCP 1520 gehoerenden Werkzeuge zur Assemblerprogrammierung. Die im Paket zur Verfuegung stehenden SCP-Funktionen und die Hardware des A 5120/30 sind fuer den Programmierer von Anwenderprogrammen interessant. Spezielle Probleme erfordern Kenntnisse aus der "Anleitung fuer den Systemprogrammierer".
Diese vorliegende Beschreibung ist Voraussetzung fuer die Arbeit mit der Dokumentation "Anleitung fuer den Systemprogrammierer".

2. Grundlagen fuer die Programmierung

2.1. Speicherkonzept

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



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

BDOS: Basic Disk Operating System
Dieser Teil des Systemes bietet dem Programmierer alle die Routinen an, die dem problemlosen Datenaustausch mit den logischen Geraeten (siehe Pkt. 2.2) gewaehrleisten. Zur Nutzung dieser Routinen, auch BDOS-Funktionen genannt, gibt es ein standardisiertes Verfahren zu deren Aufruf (siehe Pkt. 3.).
BDOS bedient sich der physischen BIOS-Routinen.

CCP: Consol Command Processor
Mit diesem Systemteil werden die residenten Kommandos DIR, REN, TYPE, ERA, SAVE und USER realisiert. Diese Kommandos sind in der "Anleitung fuer den Bediener" umfassend beschrieben.
Programme, die residente Kommandos nicht aufrufen, koennen den von CCP belegten Speicherbereich belegen. Am Ende dieser Programme ist aber unbedingt ein System-Neustart (siehe Pkt. 3.2.) zu programmieren, der CCP von der Systemdiskette nachlaedt.

TPA: Transient Program Area
Dies ist der Speicherbereich, der den transienten Kommandos und Programmen zur Verfuegung 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 fuer den Anwenderprogrammierer zur Verfuegung.
Die Groesse des TPA (einschliesslich des ueberlagerbaren CCP-Bereiches und des Systemdatenbereiches) wird in der Kaltstartmeldung auf dem Bildschirm angezeigt.

Systemdatenbereich:
In diesem Bereich sind Adressen, Kennbytes, Spruenge und Standardpuffer fuer die Arbeit des SCPX abgelegt:

Adresse	Bedeutung
0000H - 0002H	Sprung an den Sprungbefehl fuer Warmstart in der BIOS-Sprungtabelle. Dies ist der zweite Sprung in dieser Tabelle.
0004H	aktuelles logisches Laufwerk - LW A: Inhalt der Adresse 0004H = 0 . . -LW E: Inhalt der Adresse 0004H = 4
0005H-0007H	Sprung an BDOS. Die Adresse auf 0006H und 0007H - also die Anfangsadresse von BDOS - kann somit auch zur Berechnung der Groesse von TPA genutzt werden.
0038H-003AH	reserviert fuer das Programm DU (siehe Pkt. 8.).
003EH-003FH	Warmstartadresse
005CH-007CH	Standard-Dateisteuerblock (file control block - FCB, siehe auch Pkt. 2.3.)
0080H-00FFH	Standard Ein-/Ausgabe-Puffer (direct memory access - DMA, siehe auch Pkt. 2.3.)

2.2. Logische Geraete und ihr Aufruf

Vom SCPX 1520 werden folgende logische Geraete unterstuetzt:

- die Konsole,

- ein Drucker,

- 1...5 logische FD-Laufwerke mit der Bezeichnung A...E.

Die Konsole enthaelt fuer die Zeicheneingabe eine Tastatur, Zeichen (64x16 Zeichen in eingeschraenktem Umfang auch moeglich).

In der Dokumentation "Hardwarebeschreibung" ist dargestellt, welche physischen Geraete den oben genannten logischen Geraeten zugeordnet werden koennen.

Je nach Anlagenausstattung koennen 1...4 physische FD-Laufwerke angeschlossen werden. Das letzte 8"-Laufwerk wird jeweils zum "zusaetzlichen" logischen Laufwerk, d.h. in diesem Laufwerk kann sowohl das Diskettenformat mit 1024 Byte/Sektor als auch das 128 Byte/Sektor-Format mit dem Verschiebefaktor 6 (zwei logisch aufeinanderfolgende Sektoren sind auf der Diskette physisch durch 6 Sektoren getrennt) verarbeiten.

Sind in einer Anlage z.B. zwei 8"-Laufwerke und zwei 5,25"-Laufwerke angeschlossen, so wird das zweite 8"-Laufwerk als Laufwerk "B" das Diskettenformat 1024 Byte/Sektor und als "zusaetzliches" logisches LW "E" das Diskettenformat 128 Byte/Sektor verarbeiten. Enthaelt die Anlage nur 5,25"-Laufwerke, kann das letzte Laufwerk beide Formate bei entsprechender Anwahl verarbeiten.
Um mit den logischen Geraeten arbeiten zu koennen, sind die BDOS-Funktionen zu verwenden (siehe Pkt. 3.).

2.3. Kommando-Eingabe

Ein Kommando-/Programmaufruf erfolgt stets durch die Eingabe des residenten Kommandos oder des Dateinamens des Programmes. Der Dateityp COM darf nicht eingegeben werden. Danach ist noch die Eingabe von Parametern moeglich. Diese werden generell in Page 0 des Speichers auf den Adressen 0081H bis 00FBH abgelegt. Es sind somit insgesamt 123 Zeichen bei der Parametereingabe moeglich, die tatsaechliche Anzahl eingegebener Zeichen wird auf der Adresse 0080H abgelegt.
Der Programmierer hat aber darauf zu achten, dass die Adresse 0080H nach einem Kalt- und Warmstart die aktuelle DMA-Adresse fuer Floppy-Disk-Zugriffe ist. Werden die Parameter noch nach einem FD-Zugriff benoetigt, muss der Programmierer diese Parameter in einen anderen Speicherbereich verschieben oder eine andere DMA-Adresse definieren (siehe Pkt.3., Funktion 26).
Neben der Speicherung der Parameter ab 0080H werden ausserdem

die direkt dem Kommando bzw. Programmnamen moeglicherweise folgenden zwei Dateibezeichnungen (Laufwerkscode, Dateiname, Dateityp) auf 005CH und 006CH abgelegt. Sind der Name und Typ der Datei kuerzer als 11 Zeichen, werden die freien Bytes mit 20H belegt. Zwischen Dateiname und -typ ist bei der Eingabe ein Punkt einzugeben.
Die Adresse 005CH ist aber auch als Standardadresse fuer den FCB nach Kalt- und Warmstart definiert. Dies hat der Programmierer in gleicher Weise zu beachten wie die Nutzung des Speicherbereichs ab 0080H.
Bei der Speicherung der Parameter ab 005CH, 006CH und 0080H werden generell alle eingegebenen Buchstaben als Grossbuchstaben abgelegt!

Beispiel:

A)TEST 123.abc B:5.DE fgh(ENTER)

Es wird gespeichert:

- auf 005CH: 00H
  auf 005DH: 31H ("1")
  auf 005EH: 32H ("2")
  auf 005FH: 33H ("3")
  auf 0060H
  bis 0064H: 20H
  auf 0065H: 41H ("A")
  auf 0066H: 42H ("B")
  auf 0067H: 43H ("C")

- auf 006CH: 02H
  auf 006DH: 35H ("5")
  auf 006EH
  bis 0074H: 20H
  auf 0075H: 44H ("D")
  auf 0076H: 45H ("E")
  auf 0077H: 20H

- auf 0080H bis 0093H:
  13H, 20H, 31H, 32H, 33H, 2EH, 41H, 42H, 43H, 20H, 42H, 3AH, 35H, 2EH, 44H, 45H, 20H, 46H, 47H, 48H

Ein Kommando- oder Programmaufruf erfolgt stets vom CCP-Status aus, d.h. das Kommando bzw. Programm kann mit RET zu CCP zurueckkehren. Der Programmierer muss aber unbedingt beachten:

- Der von CCP bereitgestellte Stack ist nur 16 Byte gross, wobei bereits die adresshoechsten Bytes die Rueckkehradresse zu CCP enthalten. Deshalb sollte in jedem Programm ein eigener Stack definiert werden. Vor dem RET zur Rueckkehr in das CCP ist dann der alte Stackpointer wieder einzustellen.

- Groessere Programme koennen CCP ueberlagern. In diesem Fall ist das Programm unbedingt mit dem System-Neustart (siehe Pkt.3.2) zu beenden, damit CCP von der Systemdiskette im Laufwerk A neu geladen wird. Dadurch wird der CCP-Status

wieder erreicht, ohne dass der alte Stackpointerstand eingestellt werden muss.
Es ist zweckmaessig, das Anlegen eines programmeigenen Stacks und das Beenden eines Programms mit dem System-Neustart generell vorzusehen.

3. BDOS-Funktionen

3.1. Vorbemerkungen

Fuer den Zugriff auf die logischen Geraete Konsole, Drucker und FD-Laufwerk sind die BDOS-Funktionen zu verwenden.
Zum Aufruf dieser insgesamt 37 Funktionen gibt es ein standardisiertes Verfahren:

Die Funktionen sind von 0-36 durchnumeriert. Diese Nummer ist im Register C einzustellen. Der Aufruf der BDOS-Funktion erfolgt durch einen CALL an die Adresse 0005H. Dort ist vom SCPX ein Sprung an den Anfang von BDOS eingetragen. Sind fuer die ausgewaehlte Funktion weitere Zusatzinformationen erforderlich, sind diese vor dem CALL 5 im Registerpaar DE abzulegen.

Demnach sieht der Aufruf einer BDOS-Funktion n im allgemeinen wie folgt aus:

```
LD   C,n
[LD  DE,par]
CALL 5
```

Zahlreiche BDOS-Funktionen senden eine Rueckmeldung ueber die Ausfuehrung der aufgerufenen Funktion. Diese ist entweder im Register A oder im Registerpaar HL abgelegt.

Die BDOS-Funktionen kann man in zwei grosse Gruppen einteilen:

- Funktionen fuer die "einfachen" E/A-Geraete (Konsole, Drucker),

- Funktionen fuer die Diskettenarbeit.

Dazu kommt noch die Funktion 0, die sich nicht in diese beiden Gruppen einordnen laesst.

3.2. BDOS-Funktion 0: - System-Neustart

Mit dieser Funktion erfolgt die Rueckkehr in den CCP-Status durch einen Warmstart.
Die Wirkung ist gleich der eines Sprunges auf die Adresse 0000H, da dort der Sprung zum Warmstart des BIOS eingetragen ist.
Nach dem System-Neustart (LD C,0 und CALL 5) oder dem Sprung

an die Adresse 0000H (z.B. JP 0) meldet sich CCP mit dem aktuellen Laufwerk und dem Bereitschaftszeichen, und es erwartet eine neue Kommandoeingabe.

3.3. BDOS-Funktionen fuer Konsole und Drucker

Funktion 1: Konsolen-Eingabe

Das naechste Zeichen von der Tastatur wird in das Register A uebergeben. Die Routine wartet so lange, bis ein Zeichen eingegeben wurde. Das eingegebene Zeichen wird auf dem Bildschirm stets mit ausgegeben. Die Steuerzeichen 08H, 09H, 0AH und 0DH sind zugelassen.

Funktion 2: Konsolen-Ausgabe

Das im Register E befindliche Zeichen wird an die Konsole (Bildschirm) ausgegeben. Die Steuerzeichen CTRL S (siehe Anlage D), CTRL P und CTRL I (siehe Funktion 10) werden nicht auf dem Bildschirm angezeigt (im weiteren Text CTRL = ^).

Beispiel:

```
LD    E,31H
LD    C,2
CALL  5          --->   Ausgabe einer "1" auf die
                        aktuelle Kursorposition
```

Funktion 3: nicht verwenden

Funktion 4: nicht verwenden

Funktion 5: Drucker-Ausgabe

Das Zeichen, das im Register E steht, wird an den Drucker ausgegeben. Es sind nur die Steuerzeichenfolgen auszugeben, die vom jeweilig angeschlossenen Drucker auch verarbeitet werden koennen.

Funktion 6: Direkte Konsolen-Ein-/Ausgabe

Enthaelt das Register E den Wert FFH, so wird das Zeichen von der Tastatur in das Register A uebernommen. Wurde keine Taste betaetigt, so enthaelt das Register A den Code 00H. Es wird also nicht auf eine Eingabe gewartet! Enthaelt das Register E einen Wert ≠ FFH, so wird dieses Zeichen an die Konsole (Bildschirm) ausgegeben.

Funktion 7: nicht verwenden

Funktion 8: nicht verwenden

Funktion 9: Ausgabe einer Zeichenkette

Mit dieser Funktion wird eine Zeichenkette auf den Bildschirm ausgegeben. Die Zeichenkette ist mit dem Registerpaar DE adressiert, ihr Ende wird durch das Zeichen "x" (24H) definiert. Steuerzeichen sind in der Zeichenkette zugelassen. Dabei gilt fuer ^S, ^P und ^I das unter Funktion 2 beschriebene.

Beispiel:

```
        LD   DE,KETTE
        LD   C,9
        CALL 5
        .
        .
KETTE:DB   "MATTHIASx"  --> auf dem Bildschirm wird ab
                            aktueller Kursorposition
                            MATTHIAS angezeigt.
```

Funktion 10: Eingabe in Konsol-Puffer

Diese Funktion dient der Eingabe einer ganzen Zeile in einen durch das Registerpaar DE adressierten Puffer, wobei gleichzeitig die Anzeige der eingegebenen Zeichen auf dem Bildschirm erfolgt.
Das 1. Byte des Puffers muss die maximal eingebbare Zeichenzahl enthalten (1-255; dem Wert 0 wird die Anzahl 1 zugeordnet).
Das 2. Byte des Puffers enthaelt die tatsaechliche Anzahl eingegebener Zeichen.
Ab der 3. Position des Puffers stehen die eingegebenen Zeichen. Der Puffer wird vor der Eingabe nicht geloescht, so dass nach dem zuletzt eingegebenen Zeichen beliebige Zeichen folgen.

Die Zeicheneingabe wird beendet durch

- Erreichen der Maximalanzahl (1.Byte des Puffers),
- Eingabe (ENTER),
- Eingabe ^J (siehe Anlage D),
- Eingabe ^M (siehe Anlage D).

Auf dem Bildschirm wird dabei stets der "Kursor an Anfang der neuen Zeile" ausgegeben.

Eine Reihe von Steuerzeichen ermoeglicht ein einfaches zeilenorientiertes Editieren:

DEL : Das letzte Zeichen im Puffer wird geloescht, auf den Bildschirm aber erneut ausgegeben.

^R : Die Zeile wird nochmals ausgegeben - so, wie sie im Puffer steht. Dies ist nach DEL sinnvoll.

^C : Wird diese Taste am Zeilenanfang betaetigt, so wird ein Warmstart ausgeloest. An anderen Positionen bewirkt dieses Steuerzeichen die Ablage des Codes (Ø3H) im Puffer und die Anzeige ^C auf dem Bildschirm.

^E : Dieses Steuerzeichen gelangt nicht in den Puffer, es bewirkt auf dem Bildschirm eine Zeilenschaltung und eine Positionierung des Kursors an den Zeilenanfang (kein Eingabeende!).

^H : Das zuletzt eingegebene Zeichen wird sowohl im Puffer als auch auf dem Bildschirm geloescht.

^U : Der Puffer wird geloescht. An den Bildschirm wird ein "#" ausgegeben, die folgende Eingabe wird auf der naechsten Zeile angezeigt.

^X : Der Puffer wird geloescht, ebenfalls alle eingegebenen Zeichen auf dem Bildschirm. Der Kursor nimmt die Position ein, die er vor dem Funktionsaufruf hatte.

^I : In den Puffer wird der Code (Ø9H) eingetragen. Auf dem Bildschirm werden so viele Leerzeichen ausgegeben, dass eine 8-Zeichen-Tabulation entsteht (Kolonnen 1,8,15,22...).

^P : Durch dieses Steuerzeichen wird bewirkt, dass jedes Zeichen nicht nur zum Bildschirm sondern auch zum Drucker ausgegeben wird. Diese Parallelausgabe bleibt ueber die Funktion 10 hinaus erhalten, so dass bei folgenden Funktionen 2, 9 und 1Ø eine Ausgabe an Bildschirm und Drucker erfolgt.
Die zusaetzliche Ausgabe an den Drucker kann durch eine erneute Eingabe von ^P innerhalb der Funktion 1Ø abgeschaltet werden.

Funktion 11: Abfrage Konsolen-Status

Mit dieser Funktion wird geprueft, ob ein Zeichen von der Tastatur eingegeben wurde. Der Code ØØH im Register A zeigt an, dass keine Taste betaetigt wurde. Im anderen Fall enthaelt Register A Ø1H. Mit der Funktion 1 oder 10 kann der Tastaturwert uebernommen werden. Bis dahin wird auch bei wiederholter Ausfuehrung der Funktion 11 stets der Rueckgabecode in A den Wert Ø1H annehmen, so dass kein Zeichen verloren geht.

Beispiel:

Programmschleife, die mit einer beliebigen Taste beendet werden soll:

```
SCHLEIFE:  ...
           ...
           ...
           LD   C,ØBH
           CALL 5
           OR   A
           JP   Z,SCHLEIFE
```

Funktion 12: Abfrage Versionsnummer

Im Register H wird der Code ØØH, im Register L der Code 22H abgelegt. Damit wird dem Anwender programmtechnisch mitgeteilt, dass das SCPX kompatibel zur Version 2.2 des Betriebssystems CP/M ist.

3.4. BDOS-Funktionen fuer den Diskettenzugriff

Bei fast allen dieser Funktionen wird ein Parameterblock benoetigt, der alle Informationen zur Diskettenarbeit enthaelt. Dieser Block wird Standard-Dateisteuerblock (FCB) genannt, seine Adresse ist vor dem BDOS-Aufruf im Registerpaar DE einzustellen.

Der FCB besteht bei sequentiellem Zugriff aus 33 Byte. Wird auf eine Datei direkt zugegriffen, so sind 36 Bytes erforderlich. Die Bytes haben folgende Bedeutung:

Byte	Kurzbezeichnung	Bedeutung
Ø	dr	Laufwerkscode Ø...aktuelles LW (siehe auch Funktion 14) 1...LW A 2...LW B 3...LW C 4...LW D 5...LW E
1...8	f1...f8	Dateiname Bit 7=Ø
9...11	t1...t3	Dateityp Die Bits 7 von t1 und t2 haben folgende Bedeutung: Bit 7 von t1 = 1 ---> die Datei ist eine Nur-Lese-Datei (R/O). Bit 7 von t2 = Ø ---> die

		Datei ist eine SYS-Datei. (siehe dazu auch "Anleitung fuer den Bediener" Pkt.3.7)
12	ex	aktuelle Bereichs(Extent-)-Nummer
13,14		fuer SCPX reserviert
15	rc	Anzahl Saetze (128 Byte-Satz) des aktuellen Extent (Wertebereich: Ø-128)
16...31		fuer SCPX reserviert
32	cr	aktuelle Satznummer fuer den naechsten sequentiellen Zugriff
33-35	r0,r1,r2	nur fuer Direktzugriff erforderlich! r0- L-Teil) r1- H-Teil) der Satznummer (Wertebereich:Ø...65535) r2- fuer Ueberlauf

Vor jeder Dateioperation ist ein so aufgebauter FCB im Programm bereitzustellen. Waehrend in die Bytes Ø-11 die o.g. Werte einzutragen sind, sind die anderen Bytes vor dem Eroeffnen einer Datei normalerweise auf ØØH zu setzen (vor allem "ex" und "cr").

Funktion 13: Ruecksetzen Diskettensystem

Um in einem Programm Disketten ohne Warmstart (der zum Programmabbruch fuehrt) wechseln zu koennen, ist diese Funktion erforderlich. Sonst fuehrt ein Diskettenwechsel ohne Warmstart oder ohne anschliessende Funktion 13 zu R/O-Fehler. Mit dieser Funktion wird das LW A das aktuelle Laufwerk, alle Laufwerke erhalten den R/W-Status (siehe Funktion 28) und die aktuelle DMA-Adresse wird auf ØØ8ØH eingestellt (siehe Funktion 26).

Funktion 14: Aktuelles logisches Laufwerk selektieren

Der Inhalt vom Register E gibt an, welches logische Laufwerk fuer die folgenden Dateioperationen das aktuelle LW ist. Bei LW A ist E=ØØ zu setzen, bei LW B ist E=Ø1 zu setzen usw..
Ist z.B. mit dieser Funktion das LW C selektiert, so bedeutet der Wert ØØ im ersten Byte des FCB, dass mit diesem FCB nun die Diskette im Laufwerk C angesprochen wird. Das ausgewaehlte Laufwerk bleibt jetzt aktiviert (im "on-line-Zustand") bis zum naechsten Ruecksetzen Diskettensystem oder Warmstart. Bei einem vorherigen Diskettenwechsel wuerde das Laufwerk in den R/O-Status gehen.

Funktion 15: Eroeffnen einer Datei (OPEN)

Im Registerpaar DE steht die Adresse des FCB. Mit dieser Funktion wird die im FCB benannte Datei (Bytes f1...f8, t1...t3), die auf dem durch das Byte Ø des FCB ausgewaehlten Laufwerks existieren muss, aktiviert. Dabei werden bestimmte Werte aus der Directory-Eintragung in den FCB uebernommen, so dass anschliessend die Funktionen fuer das Lesen und Schreiben ausgefuehrt werden koennen.
Wird die Datei gefunden, so wird im Register A der Wert ØØH, Ø1H, Ø2H oder Ø3H gesetzt. Im anderen Fall enthaelt das Register A den Wert FFH.
Im Namen der Datei im FCB koennen auch Fragezeichen (Codierung 3FH) enthalten sein. Fuer sie kann jedes andere beliebige Zeichen im Namen der Datei in der Directory-Eintragung an der gleichen Position stehen. Es wird die erste Datei in der directory aktiviert, die mit dem Zeichen in den anderen Positionen uebereinstimmt.
Soll nach dem Eroeffnen der Datei sequentiell auf den ersten Satz zugegriffen werden, so ist das Byte "cr" im FCB auf ØØH zu setzen.

Funktion 16: Schliessen einer Datei (CLOSE)

Im Registerpaar DE ist die Adresse des FCB anzugeben. Mit dieser Funktion werden die im FCB gespeicherten Informationen in die entsprechende Directory-Eintragung uebernommen. Die Datei muss aber vorher durch die Funktion 15 oder 22 aktiviert worden sein. Wurde von der Datei nur gelesen, ist kein Schliessen erforderlich, aber nach Schreiboperationen unbedingt.
Steht nach dem Funktionsaufruf im Register A ein Wert zwischen ØØH und Ø3H, so war die Funktion erfolgreich. Wurde der Name der Datei nicht gefunden, so enthaelt Register A den Wert FFH.

Funktion 17: Suchen nach erster Eintragung

Mit dieser Funktion werden dem Benutzer (vor allem dem Systemprogrammierer) die Directory-Eintragungen zugaenglich gemacht. Die directory wird von Anfang an nach einer Eintragung entsprechend der FCB-Eintragungen dr, f1...f8, t1...t3, ex durchsucht.
Das Registerpaar DE enthaelt wieder die FCB-Adresse. Fragezeichen im Namen lassen an diesen Positionen jeden Buchstaben zu. Ein Fragezeichen im Byte Ø des FCB (dr) ermoeglicht das Durchsuchen nach allen Eintragungen, auch aller USER-Bereiche, im aktuellen Laufwerk.
Ein erfolgreiches Suchen wird im Register A durch die Werte ØØH, Ø1H, Ø2H oder Ø3H angezeigt. Im anderen Fall enthaelt das Register den Wert FFH.
Die Eintragung, die entsprechend der FCB-Vorgaben ermittelt wurde und 32 Byte lang ist, wird in einem 128 Byte-Puffer eingetragen. Dieser beginnt entweder bei ØØ8ØH oder bei einer mit der Funktion 26 vorgegebenen Adresse.

Je nach dem Wert von A ist die Eintragung ab dem ersten Byte (A=ØØH), 33.Byte (A=Ø1H), 65. Byte (A=Ø2H) oder 97. Byte (A=Ø3H) innerhalb des Puffers abgelegt.

Funktion 18: Suchen nach der naechsten Eintragung

Diese Funktion wirkt aehnlich der Funktion 17, das Durchsuchen der directory erfolgt aber ab der aktuellen Position. Diese Funktion ist im Anschluss an die Funktion 17 zu programmieren, wenn nach weiteren Dateien (im Namen des FCB sind Fragezeichen enthalten) oder weiteren Eintragungen der gleichen Datei (grosse Dateien koennen mehrere Eintragungen belegen) gesucht wird.

Funktion 19: Loeschen einer Datei

Im Registerpaar DE ist die FCB-Adresse vorzugeben. Mit dieser Funktion wird die Datei auf der Diskette geloescht. Enthaelt der Dateiname oder Dateityp im FCB ein oder mehrere Fragezeichen, so werden alle Dateien geloescht, die in den anderen Positionen gleiche Zeichen enthalten.
Steht als Rueckmeldung im Register A ein FFH, so wurde keine Datei auf der Diskette gefunden, die der Dateibezeichnung im FCB entspricht.
Erfolgreiches Loeschen einer Datei wird mit den Werten ØØH, Ø1H, Ø2H oder Ø3H im Register A zurueckgemeldet.

Funktion 20: Sequentielles Lesen

Im Registerpaar DE ist die FCB-Adresse vorzugeben. Es werden mit dieser Funktion 128 Byte (ein Satz) von dieser Datei in den Speicher eingelesen. Die Speicheranfangsadresse (DMA-Adresse) ist entweder ØØ8ØH oder sie wird durch die Funktion 26 vorgegeben. Die Angabe "cr" im FCB gibt den naechsten zu lesenden Satz an (cr=ØØH bedeutet also, es wird der 1. Satz gelesen). Nach dem Lesen wird "cr" automatisch um 1 erhoeht. Wird die Extentgrenze erreicht, so wird auch automatisch auf den folgenden Extent umgeschaltet und "cr" auf ØØH gesetzt. Wird im Register A ein Wert ungleich ØØH gemeldet, so ist das Dateiende erreicht. Bei erfolgreichem Lesevorgang steht im Register A der Wert ØØH.

Funktion 21: Sequentielles Schreiben

Im Registerpaar DE ist die FCB-Adresse vorzugeben. Von der aktuellen DMA-Adresse wird ein Satz in die durch FCB vorgegebene Datei uebertragen, wobei "cr" die Position des Satzes in der Datei festlegt. Nach dem Schreiben wird "cr" automatisch um 1 erhoeht. Das Umschalten auf einen naechsten Extent erfolgt ebenfalls automatisch, "cr" wird dabei auf ØØH gesetzt.
Ein erfolgreicher Schreibversuch wird mit ØØH im Register A

zurueckgemeldet. Ist die Diskette gefuellt und der Satz kann nicht mehr aufgezeichnet werden, so steht im Register A ein Wert ungleich ØØH.

Beachte:
Nach dem Schreiben muss die CLOSE-Funktion folgen, damit die Informationen der directory auf der Diskette aktualisiert werden.

Lese - und Schreibfehler fuehren zu der Bildschirmausschrift:

SCPX ERR ON d: BAD SECTOR

(d gibt das logische Laufwerk an: A, B, C, D oder E).

Mit der Taste <ENTER> wird die Fehlermeldung ignoriert und das Programm fortgesetzt. Dabei steht im Register A ein Wert ungleich ØØH!
Es besteht ausserdem die Moeglichkeit, mit ^C das Programm durch Sprung zum Warmstart zu beenden.

Funktion 22: Anlegen Datei

Ist eine neue Datei anzulegen, so ist diese Funktion zu verwenden. Die Funktion 15 (OPEN) ist nur fuer bereits existierende Dateien wirksam! Im Registerpaar DE ist die FCB-Adresse vorzugeben.
Da bei dieser Funktion keine Kontrolle auf eine bereits unter gleicher Bezeichnung bestehende Datei erfolgt, sollte sicherheitshalber vorher die Funktion 19 (Loeschen einer Datei) ausgefuehrt werden.
Die Datei wurde erfolgreich in der directory angelegt, wenn im Register A als Rueckmeldung der Wert ØØH bis Ø3H steht. Ein FFH im Register A sagt aus, dass in der directory keine Eintragung mehr frei ist.
Nach der Funktion 22 ist die Funktion 15 (OPEN) nicht erforderlich!

Funktion 23: Umbenennen Datei

Mit dieser Funktion kann auf der Diskette eine Datei umbenannt werden.
Das Registerpaar DE adressiert einen FCB, der etwas anders aufgebaut ist als bei den anderen Funktionen:

Die Bytes Ø-15 enthalten im ueblichen Format die "alte" Dateibezeichnung einschliesslich Laufwerksangabe, die Bytes 16-32 die "neue" Dateibezeichnung. Hier ist die Laufwerksangabe bedeutungslos. Im Register A steht nach erfolgreicher Ausfuehrung ein Wert zwischen ØØH und Ø3H. Wenn die umzubennende Datei nicht gefunden wurde, so steht im Register A der Wert FFH.

Beispiel:

```
        .
        .
        LD DE,FCBU      ;Die Datei DATEIALT.123 wird
        LD C,17H        ;in Datei DATEINEU.456 um-
        CALL 5          ;benannt
        .
        .
FCBU:db 0,'DATEIALT123',0,0,0,0
     db 0,'DATEINEU456',0,0,0,0
```

Funktion 24: Abfrage nach Laufwerken im "on-line-Zustand"

Mit dieser Funktion laesst sich feststellen, mit welchen Laufwerken bereits gearbeitet wurde. Im Register A und L entsprechen nach der Funktion die Bits 0-4 den logischen Laufwerken A, B, C, D, E. Ein gesetztes Bit ("1") bedeutet, dass auf die Diskette in diesem Laufwerk bereits zugegriffen wurde.

Nach der Funktion 13 (RESET) ist z.B. nur das Bit 0 vom Register A und Register L gesetzt.

Die Bits 5-7 der Register A und L und alle Bits des Registers H sind stets nicht gesetzt ("0").

Funktion 25: Abfrage nach aktuellem Laufwerk

Diese Funktion meldet in den Bits 0-4 des Registers A, welches logische Laufwerk das aktuelle ist. Ist z.B. das Bit 3 gesetzt, so ist das logische Laufwerk D das aktuelle Laufwerk.

Funktion 26: Einstellen DMA-Adresse

Fuer den Datentransport von und zur Diskette ist ein 128 Byte grosser Puffer erforderlich. Nach Kalt-, Warmstart und Reset Diskettensystem (Funktion 13) hat dieser Puffer die Adresse 0080H.
Durch diese Funktion kann eine andere Adresse fuer diesen Puffer eingestellt werden. Sie ist im Registerpaar DE vorzugeben. Diese DMA-Adresse bleibt wirksam bis zur naechsten Funktion 26, Funktion 13 oder bis zum naechsten Kalt- oder Warmstart.

Funktion 27: Bereitstellen Adresse des Allocation-Vektors

Fuer jedes logische Laufwerk, auf das einmal zugegriffen wurde, gibt es im Speicher eine Tabelle, die die Belegung der Diskette (belegte Bloecke - freie Bloecke) widerspiegelt.
Mit dieser Funktion wird im Registerpaar HL die Adresse dieser Tabelle (Allocation-Vektor) fuer das aktuelle Laufwerk

bereitgestellt.
Diese Funktion ist normalerweise nur fuer den Systemprogrammierer von Bedeutung.

Funktion 28: Diskette mit Schreibschutz versehen

Nach dieser Funktion kann von der Diskette im aktuellen Laufwerk nur noch gelesen werden. Ein Schreiben wird verhindert und es wird in diesem Fall folgende Ausschrift auf dem Bildschirm erscheinen:

SCPX ERR ON d:R/O

(d steht fuer das angesprochene Laufwerk: A, B, C, D oder E).

Funktion 29: Abfrage nach Laufwerk mit Schreibschutz

In den Registern A und L wird wie bei Funktion 24 ein Code abgelegt, aus dem zu entnehmen ist, welches logische Laufwerk den R/O(read only)-Status hat. Das diesen Laufwerken zugeordnete Bit ist auf "1" gesetzt.
Der R/O-Status kann durch die Funktion 28 und durch einen Diskettenwechsel (ohne folgende Funktion 13) gesetzt sein.

Funktion 30: Setzen Datei-Attribute

Mit dieser Funktion koennen Dateien mit den Attributen R/O (Nur-Lese-Datei) oder R/W (Lese-Schreib-Datei) und DIR (Datei wird beim residenten Kommando DIR angezeigt) oder SYS (keine Anzeige bei DIR) versehen werden (siehe auch "Anleitung fuer den Bediener", Pkt. 3.7.).
Das Registerpaar DE adressiert einen FCB. Ist das Bit 7 von t1 gesetzt, so wird die Datei mit dieser Funktion zu einer R/O-Datei. Ist das Bit nicht gesetzt, wird die Datei zu einer R/W-Datei.
Ein gesetzes Bit 7 von t2 versetzt die Datei in den SYS-Datei-Status. Im anderen Fall erhaelt die Datei das DIR-Attribut.

Funktion 31: Bereitstellen Adresse der Laufwerksparameter

Nach dem Funktionsaufruf steht im Registerpaar HL die Adresse der Parametertabelle des aktuellen Laufwerkes. Damit kann der Systemprogrammierer spezielle Informationen auswerten oder Manipulationen der Parameter durchfuehren.

Funktion 32: Einstellen/Abfragen Benutzercode

Wird im Register E ein FFH eingetragen, so wird nach dem Funktionsaufruf im Register A der aktuelle Benutzercode zurueckgemeldet.
Ist der Wert im Register ungleich FFH, so werden durch die

Funktion 32 die unteren 5 Bit dieses Wertes als neuer Benutzercode eingestellt (ØØH...1FH).

Beachte:
Mit dem residenten Kommando USER lassen sich nur die Benutzerbereiche Ø-15 (ØØH...ØFH) einstellen, so dass die anderen Kommandos nur in diesen Benutzerbereichen wirksam werden koennen. Es sollte deshalb generell nur mit dem Benutzercode ØØH...ØFH gearbeitet werden.

Funktion 33: Lesen mit direktem Zugriff

Fuer den direkten Zugriff sind die Bytes rØ, r1 und r2 im FCB erforderlich, dessen Adresse im Registerpaar DE stehen muss. In rØ ist der L-Teil und in r1 der H-Teil der Adresse des Satzes einzutragen, der gelesen werden soll. Das Byte r2 ist auf ØØH zu setzen.
Bei den vorher erforderlichen Funktionen 15 bzw. 22 (OPEN, Anlegen Datei) ist zu garantieren, dass im adressierten FCB das Byte 12 (ex) auf ØØH gesetzt ist.
Der Satz wird im durch die DMA-Adresse vorgegebenen Puffer eingetragen.

Im Gegensatz zum sequentiellen Lesen wird die Satzadresse nach dem Zugriff nicht um 1 erhoeht, so dass eine sofort folgende Funktion 33 den gleichen Satz lesen wuerde.
Nach dem Lesen werden durch diese Funktion ausserdem die aktuelle Satznummer "cr" und die Extentnummer "ex" gesetzt. Damit ist ein anschliessendes sequentielles Arbeiten moeglich.

Es ist zu beachten, dass der zuletzt gelesene Satz eingestellt ist, d.h. er wird dann nochmals gelesen bzw. ueberschrieben.

Bei erfolgreichem Lesen wird im Register A eine ØØH gemeldet. Da nicht alle Satzadressen beim direkten Zugriff zwischen niedrigstem und hoechstem Wert existieren muessen (siehe Funktion 34), kann es zu folgenden Fehlermeldungen im Register A kommen:

ØlH Es sollte ein Satz mit unbeschriebenen Daten gelesen werden.

Ø3H Vor diesem Funktionsaufruf wurde die Funktion Schreiben mit direktem Zugriff ausgefuehrt. Das jetzt eventuell erforderliche Aktualisieren der Directory-Eintragung, durch das vorherige Schreiben hervorgerufen, konnte nicht ausgefuehrt werden.

Ø4H Es sollte ein Satz von einem unbeschriebenen Extent gelesen werden.

Ø6H Mit der eingestellten Satzadresse wuerde das Diskettenende ueberschritten werden.

Funktion 34: Schreiben mit direktem Zugriff

Wie beim Lesen mit direktem Zugriff geben das Registerpaar DE die FCB-Adresse und dort die Bytes rØ und r1 die Adresse des Satzes an. An diese Adresse wird der Satz aus dem durch die DMA-Adresse vorgegebenen Puffer geschrieben und falls ein neuer Extent dafuer erforderlich ist, dieser Extent in die directory eingetragen.
Die Satzadresse wird wie beim Lesen nicht erhoeht.
Es wird aber die aktuelle Satznummer "cr" und die Extentnummer "ex" im FCB aktualisiert, so dass ein anschliessendes sequentielles Arbeiten (wie beim Lesen mit direktem Zugriff beschrieben) moeglich ist.

Beachte:
Sobald ein Satz in einen Block (2K Byte) der Datei geschrieben wurde, gelten alle anderen 15 Saetze in diesem Block als geschriebene Daten. Das Lesen eines dieser 15 Saetze bringt also keinen Fehlercode ØlH.

Die Fehlercodes sind die gleichen wie beim Lesen mit direktem Zugriff (ausser ØlH und Ø4H), dazu kommt der Fehlercode Ø5H. Diese Fehlermeldung signalisiert, dass in der directory fuer die erforderliche neue Eintragung kein Platz mehr frei ist.

Funktion 35: Bereitstellen naechste freie Satzposition

Mit dem Registerpaar DE ist ein FCB zu adressieren.
Nach Ausfuehrung der Funktion enthalten die Bytes rØ und r1 die Adresse nach dem letzten Satz der Datei.
Damit ist es einfach moeglich, eine Datei mit Direktzugriff weiterzuschreiben.

Beachte:
Wurde die Datei sequentiell geschrieben, so geben rØ und r1 die Anzahl tatsaechlich geschriebener Saetze an.
Bei einer Datei, die mit Direktzugriff geschrieben wurde, ist das durch moegliche "Luecken" nicht garantiert.

Funktion 36: Bereitstellen Aktuelle Satzposition

Das Registerpaar DE adressiert den FCB, in dessen Bytes rØ und r1 durch diese Funktion die Satzadressen des letzten sequentiellen Zugriffes abgelegt werden. Damit kann z.B. auf einfache Weise vom sequentiellen zum direkten Zugriff uebergegangen werden.

4. Beispiel fuer die Anwendung der BDOS-Funktionen

Mit dem folgenden einfachen Programmbeispiel soll die prinzipielle Anwendung der BDOS-Funktionen demonstriert werden.

Aufgabe dieses Erfassungsprogramms ist:

- Bediener gibt Dateiname vor (Dateityp ist TEX).
  Eine eventuell bestehende Datei gleicher Bezeichnung wird geloescht!

- Dann kann er Saetze bis zu max. 126 Zeichen eingeben, die auf dem Bildschirm dargestellt werden. Gibt er weniger als 126 Zeichen ein, so hat er mit (ENTER) die Eingabe zu beenden. Korrekturen vor Abschluss eines Satzes sind moeglich (entsprechend BDOS- Funktion 10).

- Jeder Satz, am Ende mit Space aufgefuellt und mit den Steuerzeichen Zeilenschaltung und Kursor an Zeilenanfang versehen, wird auf der Diskette im aktuellen Laufwerk als ein Satz mit 128 Byte aufgezeichnet.

- Gibt der Bediener nur "END" bzw. "end" ein, so wird dies als letzter Satz gedeutet. Das Programm wird nach dem Aufzeichnen dieses Satzes und dem Schliessen der Datei beendet.

- Tritt ein Fehler auf, so wird die Datei nicht aufgezeichnet. Der Fehler wird angezeigt.


```
S                                       01B8  DMA
                                        01A5  FEHLER
                                        000A  LF
                                        0283  NAME
                                        0100  START

beispiel1       ASM 1520 (SCPX) V 1.0

Macros:

Symbols:
0005  BDOS                      000D  CR
01AF  ENDE                      0238  FCB
0291  FETEXT                    000C  HOME
0171  LOESCH                    0149  LOOP
01B6  PUFFER                    0259  STACK
0269  TEXT                      0175  WRITE

No Fatal error(s)
```

```
beispiel1       ASM 1520 (SCPX) V 1.0           1

                                                        title   beispiel1
0000'                                   aseg
                                        org     100h
                                        .z80
                                        page        36
```

26

```
beispiel1       ASM 1520 (SCPX) V 1.0           1-1

0005                    bdos    equ     5               ;Adresse vom Sprung an BDOS
000D                    cr      equ     0dh             ;Steuerzeichen: Kursor an Zeilenanf.
000A                    lf      equ     0ah             ;Steuerzeichen: Zeilenschaltung
000C                    home    equ     0ch             ;Steuerzeichen: Kursor an BS-Anfang

0100    31 0269         start:  ld      sp,stack+16     ;eigener Stack
0103    11 0269                 ld      de,text         ;Programmanzeige auf Bildschirm
0106    0E 09                   ld      c,9
0108    CD 0005                 call    bdos

010B    11 0283                 ld      de,name         ;Dateiname: Eingabeaufforderung
010E    0E 09                   ld      c,9
0110    CD 0005                 call    bdos
0113    11 01B6                 ld      de,puffer       ;Eingabe Dateiname
0116    0E 0A                   ld      c,10
0118    CD 0005                 call    bdos
011B    3A 01B7                 ld      a,(puffer+1)    ;Anzahl Zeichen Dateiname
011E    4F                      ld      c,a
011F    06 00                   ld      b,0
0121    21 01B8                 ld      hl,puffer+2
0124    11 0239                 ld      de,fcb+1
0127    ED B0                   ldir                    ;Dateiname---->FCB

0129    3E 7E                   ld      a,126           ;max. Eingabelaenge f.Satzeingabe
012B    32 01B6                 ld      (puffer),a
012E    11 0238                 ld      de,fcb          ;Loeschen einer evt. bestehenden Datei
0131    D5                      push    de              ;gleichen Namens
0132    0E 13                   ld      c,19
0134    CD 0005                 call    bdos
```

27

```
0137  D1              pop   de          ;Einrichten Datei
0138  D5              push  de
0139  0E 16           ld    c,22
013B  CD 0005         call  bdos
013E  3C              inc   a
013F  28 64           jr    z,fehler    ;Sprung bei Fehler

0141  11 01B8         ld    de,dma      ;Adresse fuer FD- Uebertragung
0144  0E 1A           ld    c,26
0146  CD 0005         call  bdos
0149  1E 0D     loop: ld    e,cr        ;Kursoreinstellung
014B  0E 02           ld    c,2
014D  CD 0005         call  bdos

0150  1E 0A           ld    e,1f
0152  0E 02           ld    c,2
0154  CD 0005         call  5

0157  11 01B6         ld    de,puffer   ;Texteingabe
015A  0E 0A           ld    c,10
015C  CD 0005         call  bdos

015F  21 01B8         ld    hl,dma      ;Ausgabepuffer FD
0162  3A 01B7         ld    a,(puffer+1) ;Anzahl Zeichen
0165  4F              ld    c,a
0166  06 00           ld    b,0
0168  09              add   hl,bc       ;erstes zu loeschendes Byte
0169  3E 7E           ld    a,126       ;max. Anzahl Zeichen
016B  91              sub   c
016C  28 07           jr    z,write     ;bei 126 Zeichen
```



```
016E  47              ld    b,a
016F  3E 20           ld    a,20h       ;Fuellzeichen
0171  77        loesch: ld    (hl),a
0172  23              inc   hl
0173  10 FC           djnz  loesch

0175  D1        write:  pop   de          ;Write sequentiell
0176  D5              push  de
0177  0E 15           ld    c,21
0179  CD 0005         call  bdos
017C  B7              or    a
017D  20 26           jr    nz,fehler   ;Sprung an Fehlerbehandlung

017F  3A 01B7         ld    a,(puffer+1) ;Abfrage auf "end" bzw "END"
0182  FE 03           cp    3
0184  20 C3           jr    nz,loop     ;keine 3 Zeichen eingegeben
0186  3A 01B8         ld    a,(puffer+2)
0189  CB AF           res   5,a
018B  FE 45           cp    'E'         ;Grossbuchstaben erzeugen
018D  20 BA           jr    nz,loop     ;Fortsetzen der Erfassung
018F  3A 01B9         ld    a,(puffer+3)
0192  CB AF           res   5,a
0194  FE 4E           cp    'N'
0196  20 B1           jr    nz,loop     ;Fortsetzen der Erfassung
0198  3A 01BA         ld    a,(puffer+4)
019B  CB AF           res   5,a
019D  FE 44           cp    'D'
019F  20 A8           jr    nz,loop     ;Fortsetzen der Erfassung

01A1  0E 10           ld    c,16        ;Schliessen der Datei
01A3  18 0A           jr    ende
```

```
01A5    11 0291      fehler: ld      de,fetext       ;Fehleranzeige
01A8    0E 09                ld      c,9
01AA    CD 0005              call    bdos

01AD    0E 13                ld      c,19            ;Loeschen Datei im Fehlerfall
01AF    D1           ende:   pop     de              ;FCB- Adresse
01B0    CD 0005              call    bdos
01B3    C3 0000              jp      0               ;Ende mit Warmstart

01B6    08 00        puffer: db      8,0             ;Eingabepuffer (max.Laenge,Istlaenge)
01B8                 dma:    ds      126             ;Puffer f. Eingabe und FD-Ausgabe
0236    0D 0A                db      cr,lf           ;letzte konstante Zeichen f.
0238    00           fcb:    db      0               ;FCB:aktuelles Laufwerk
0239                         ds      8,20h           ;    fuer Dateiname
0241    54 45 58             db      'TEX'           ;    Dateityp
0244                         ds      21,0            ;    Restbyte(auf 0 eingestellt)
0259                 stack:  ds      16,0            ;Anwenderstack

0269    0C 20 20 20  text:   db      home,'    ERFASSUNGSPROGRAMM',cr,lf,24h
026D    20 45 52 46
0271    41 53 53 55
0275    4E 47 53 50
0279    52 4F 47 52
027D    41 4D 4D 0D
0281    0A 24
0283    0A 44 61 74  name:   db      lf,'Dateiname : ',24h
0287    65 69 6E 61
028B    6D 65 20 3A
028F    20 24
0291    0D 0A 0A 20  fetext: db      cr,lf,lf,' Fehler (Directory oder Diskette voll) !'
0295    46 65 68 6C
0299    65 72 20 28
```



```
029D    44 69 72 65
02A1    63 74 6F 72
02A5    79 20 6F 64
02A9    65 72 20 44
02AD    69 73 6B 65
02B1    74 74 65 20
02B5    76 6F 6C 6C
02B9    29 20 21
02BC    0D 0A 24             db      cr,lf,24h

                             end
```

5. EDIT 1520 (SCPX)

5.1. Bedienung von EDIT

5.1.1. Einleitung

EDIT ist ein zeilenorientierter und zeichenorientierter Textaufbereiter. Die EDIT-Kommandos sind einfach und unkompliziert; sie sind jedoch leistungsfaehig genug, um auch den Anforderungen anspruchvoller Anwender zu genuegen.

Die ersten vier Kapitel dieser Unterlage enthalten alle Informationen, die erforderlich sind, eine umfassende Editieraufgabe zu realisieren. Die verbleibenden Kapitel beschreiben die Erweiterungen zu EDIT, die den Anwender mit komplizierteren Methoden ausruesten.

5.1.2. Abarbeiten von EDIT

Damit EDIT abgearbeitet werden kann, ist ueber Tastatur einzugeben:

EDIT (Kleinbuchstaben auch moeglich)

EDIT meldet sich mit der Ausschrift:

File: _

Anschliessend muss der Name der Datei eingegeben werden, unter Verwendung des Formats fuer Dateiname und Dateityp.
Wird bei Eingabe des Dateinamens kein Dateityp spezifiziert, haengt der Editor den Dateityp ".MAC" an.

Ein Dateiname kann ebenfalls ein anderes Laufwerk ansprechen. Das Format lautet dann:

d: Dateiname

d ist das Laufwerk, das zugewiesen werden soll. Die Zuweisung hat in Grossbuchstaben zu erfolgen!

Wenn der Dateiname auf eine Datei Bezug nimmt, die bereits existiert, ist der Dateiname einzugeben.
Besitzt die Datei keine Zeilennummern, fuegt EDIT sie ein, beginnend mit der Zeilennummer 100 und mit einer Schrittweite von 100.

Beispiel:

```
00100     xor a
00200     ld  b,a
00300     ld  c,a
```


Anschliessend zeigt EDIT an:

Editing (dateiname)

```
EDIT 1520 (SCP) V X.Y
Created: 30-Dec-83
xxxx Bytes free
*_
```

Durch das Sternzeichen (*) wird die Kommandoebene angezeigt. Saemtliche Kommandos von EDIT werden eingegeben, nachdem das Zeichen (*) angezeigt wurde.

Wenn sich der Dateiname auf eine neue Datei bezieht, die erst erzeugt werden soll, ist der Dateiname gefolgt von einem (ESC) einzugeben.
EDIT bringt dann folgende Anzeige:

Creating (dateiname)

```
EDIT 1520 (SCP) V X.Y
Created:30-Dec-83
xxxx Bytes free
*_
```

Im naechsten Schritt ist das Kommando I (i) einzugeben (siehe dazu Pkt. 5.2.1.).

EDIT zeigt die erste Zeilennummer an, 00100, gefolgt von einem Tabulator.

```
*I
00100    _
```

Nun kann die erste Zeile der Datei eingegeben werden. Die Eingabe von Kleinbuchstaben ist bei den EDIT Kommandos, ausser bei der Laufwerkszuweisung, auch moeglich. Eine Zeile besteht aus bis zu 255 Zeichen und wird durch (ENTER) abgeschlossen. Nach jeder Zeileneingabe zeigt EDIT die naechste Zeilennummer, die um 100 erhoeht wurde, an. Dies ist die "Standardschrittweite" (es gibt verschiedene Kommandos, die die Standardschrittweite abaendern koennen - siehe dazu Pkt. 5.2.). Es sind die Zeilennummern 00000 bis 99999 fuer die Nutzung in einer EDIT-Datei verfuegbar.

Verschiedene Programme von Robotron, wie z.B. ASM 1520 (SCP) und BASI 1520 (SCP) unterstuetzen solche Eingabedateien, die die EDIT-Zeilennummern einschliessen.
Tritt waehrend der Eingabe bzw. Aufbereitung einer Zeile ein Fehler auf, sollte "Loeschen" bedient werden, um die nicht korrekten Zeichen zu loeschen. Soll die gesamte Zeile geloescht werden ,ist "^U" zu betaetigen. Wenn die Eingabe der Zeilen gestoppt werden soll, ist per Taste ein (ESC) nach der naechsten verfuegbaren Zeilennummer einzugeben.

5.1.3. Beenden des Editorlaufes

***E**

Das Exit-Kommando schreibt die editierte Datei auf die Diskette und kehrt zum Systemgrundzustand zurueck. Die editierte Datei wird unter ihrer <dateibezeichnung> und die alte Version der Datei unter ihrem Dateinamen und dem Dateityp ".BAK" gerettet.

Bei Eingabe des "Quit"-Kommandos wird EDIT verlassen, ohne die editierte Datei auf Diskette auszugeben:

***Q**

Das Quit-Kommando speichert die alte Kopie der Datei unter deren urspruenglichen <dateibezeichnung> wieder ab, und kehrt dann zum Systemgrundzustand zurueck. Nachdem ein Quit-Kommando ausgefuehrt wurde, gehen alle Aenderungen verloren, die waehrend des Editorlaufes eingegeben wurden.

5.1.4. Zeilennummern und -bereiche

Die meisten Kommandos fuer EDIT erfordern eine Bezugnahme auf eine Zeilennummer oder einen Bereich von Zeilennummern. Eine Zeilennummer wird spezifiziert, indem die Nummer selbst verwendet wird (es ist nicht notwendig , fuehrende Nullen mit einzugeben), oder es wird eines der drei speziellen Zeichen verwendet, die EDIT als Zeilennummern anerkennt.

Diese speziellen Zeichen sind:

. (Punkt)	bezieht sich auf die aktuelle Zeile.
^ (Dach)	bezieht sich auf die erste Zeile der Datei.
* (Stern)	bezieht sich auf die letzte Zeile der Datei.

Bereiche koennen in einer der beiden dargestellten Moeglichkeiten spezifiziert werden:

1. Die Kennzeichnung mit einem Doppelpunkt

   300:1100

   bedeutet alle Zeilen von der Zeilennummer 300 bis zur Zeilennummer 1100 einschliesslich. Wenn die Zeilen 300 und 1100 nicht existieren, beginnt der Bereich mit der ersten Zeilennummer groesser 300 und endet mit der letzten Zeilennummer kleiner 1100.

2. Die Kennzeichnung mit einem Ausrufezeichen

   300!3

   bedeutet einen Bereich von 3 Zeilen, der ab Zeilennummer 300 beginnt. Wenn die Zeilennummer 300 nicht existiert, dann beginnt der Bereich der 3 Zeilen ab der ersten existierenden Zeilennummer groesser 300.

Beispiele fuer Zeilen- und Bereichsspezifikationen (hier zusammen mit dem Anzeigekommando "P", siehe Pkt. 5.2.4., dargestellt):

P.:2000	Anzeigen des Bereiches, von der aktuellen Zeile bis zur Zeile 2000.
P500	Zeigt die Zeile 500 an.
P.	Zeigt die aktuelle Zeile an.
P.!15	Zeigt den Bereich von 15 Zeilen, beginnend mit der aktuellen Zeile an.
P^:1500	Zeigt den Bereich an, der mit der ersten Zeile beginnt und mit der Zeile 1500 endet.
P^:*	Zeigt die gesamte Datei an.

5.1.5. Schreibweise des Formates

In diesem Dokument sind verallgemeinerte Formate der EDIT-Kommandos angegeben, die dem Anwender als Leitfaden dienen sollen. Diese Formate verwenden folgende Konventionen:

- Die Positionen, die in rechteckigen Klammern angefuehrt sind, sind wahlweise zu verwenden.
- Positionen, die in Grossbuchstaben angegeben sind, muessen auch so eingegeben werden.
- Die Positionen, die in Kleinbuchstaben ausgefuehrt und in spitze Klammern eingeschlossen sind, muessen vom Anwender zur Verfuegung gestellt werden:

<position>	Es ist eine Zeilennummer bereitzustellen (bis zu 5 Stellen) oder ".", "^" oder "*".
<range>	Es ist eine <position> (Zeilennummer) oder <range> (Bereich) bereitzustellen. <range>=<position>:<position> oder <position>!<number>

⟨number⟩ Anzahl der Zeilen, die bereitgestellt werden sollen.

⟨inc⟩ Bereitstellen einer ganzen Zahl ungleich Null, die als Schrittweite zwischen den Zeilennummern verwendet wird.

⟨dateiname⟩ Bereitstellen eines zulaessigen Dateinamens, wie im Pkt. 5.1.2. beschrieben.

- Die Positionen, die mit Hilfe eines vertikalen Striches voneinander getrennt sind, schliessen sich gegenseitig aus. Es ist eine auszuwaehlen.
- ⟨ESC⟩ wird durch EDIT als das *-Zeichen wiedergegeben.
- In einem beliebigen Kommandoformat sind Leerzeichen und Tabulatoren nicht von Bedeutung, mit Ausnahme, dass sie innerhalb einer Zeilennummer oder eines Dateinamens auftreten.
- Unterstrichene Positionen werden durch EDIT angezeigt oder gedruckt.

5.2. Beginn der Zwischenzeilenaufbereitung

Das Editieren einer Datei durch Drucken, Einfuegen, Loeschen und Ersetzen ganzer Zeilen oder Gruppen von Zeilen wird mit dem Begriff "interline editing" (Aufbereitung zwischen den Zeilen) bezeichnet. Dieser Abschnitt beschreibt die Kommandos, die verwendet werden, um diese Funktionen zu realisieren.

5.2.1. Einfuegen (Insert)

Das Insert-Kommando wird zum Einfuegen von Textzeilen in eine Datei benutzt. EDIT zeigt jede Zeilennummer waehrend der Insert-Betriebsart an. Das Format des Kommandos ist:

I[⟨position⟩[,⟨inc⟩][;⟨inc⟩]]

Das Einfuegen der Zeilen beginnt ab ⟨position⟩ und wird solange fortgesetzt, bis ⟨ESC⟩ betaetigt wurde oder bis der an dieser Stelle in der Datei zur Verfuegung stehende Speicherplatz zu Ende ist (in beiden Faellen kehrt EDIT in die Kommandoebene zurueck).

Wenn kein ⟨inc⟩ in das Kommando einbezogen wurde, gilt als Standardannahme die Standardschrittweite.
⟨inc⟩ spezifiziert eine neue Schrittweite, die dann statt der Standardschrittweite eingesetzt wird.
;⟨inc⟩ spezifiziert eine zeitweilige Schrittweite, die fuer die Verwendung mit dem aktuellen Kommando vorgesehen ist, es veraendert aber nicht die Standardschrittweite.

Fuer den Fall, dass kein Argument zusammen mit dem Insert-Kommando zur Verfuegung gestellt wurde, knuepft die Einfuegung daran an, wo das letzte Insert-Kommando abgeschlossen wurde, wobei die letzte zeitweilige (temporaere) Schrittweite angewendet wird. Beim ersten Insert-Kommando in der Abarbeitung einer vorhandenen Datei fuehrt dies zur Fehlerausschrift "Out of order".

Wird lediglich ⟨position⟩ zur Verfuegung gestellt (I⟨position⟩), wird die Standardschrittweite verwendet.
EDIT laesst nicht zu, an einer Stelle etwas einzufuegen, wo bereits eine Zeile vorhanden ist.
Wenn ⟨position⟩ eine bereits existierende Zeilennummer ist, addiert das Kommando I⟨position⟩ die Standardschrittweite (oder auch die temporaere Schrittweite, wenn sie spezifiziert wurde) zur ⟨position⟩ und gestattet das Einfuegen bei der Zeilennummer ⟨position⟩ + ⟨inc⟩.

Ist die Zeile ⟨position⟩ + ⟨inc⟩ ebenfalls vorhanden oder sind Zeilennummern zwischen ⟨position⟩ und ⟨position⟩ + ⟨inc⟩ vorhanden, wird eine Fehlernachricht ausgegeben.

Beispiel der Verwendung des Insert-Kommandos:

```
*I7000,10
07000      M = M + 1
07010      GOTO  750
07020 *
*_
```

Es ist zu beruecksichtigen, dass der Einfuegevorgang mit einem ⟨ESC⟩ abgeschlosssen wird. ⟨ESC⟩ wird per Taste am Ende der letzten Zeile eingegeben, die eingefuegt werden soll (anstelle von ⟨ENTER⟩) oder am Anfang der naechsten Zeile. Eine Zeile wird nicht gerettet, wenn ⟨ESC⟩ das erste Zeichen ist, das auf der Zeile erscheint.

5.2.2. Loeschen (Delete)

Das Loesch-Kommando loescht eine Zeile oder einen Bereich von Zeilen aus der Datei. Das Format des Kommandos ist:

D⟨range⟩

Nachdem ein Loesch-Kommando ausgefuehrt wurde, wird die erste Zeile des geloeschten Bereiches zur aktuellen Zeile bestimmt.

Beispiele fuer das Loesch-Kommando:

D1000	Loeschen der Zeile 1000
D.	Loeschen der aktuellen Zeile
D300:900	Loeschen der Zeilen 300 bis 900
D1000:*	Loeschen aller Zeilen von der Zeile 1000 an bis zur letzten Zeile

5.2.3. Ersetzen (Replace)

Das Replace-Kommando vereinigt die Wirkungen der Kommandos Loeschen und Einfuegen. Das Format des Kommandos ist:

R⟨range⟩[,⟨inc⟩|;⟨inc⟩]

Das Replace-Kommando loescht alle Zeilen innerhalb des Bereiches ⟨range⟩, danach gestattet es dem Anwender einen neuen Text einzugeben, als ob ein Insert-Kommando eingegeben wurde (EDIT gibt die Zeilennummern an).
Die Zusaetze fuer das Auswaehlen der Schrittweite zwischen den Zeilennummern sind die gleichen wie diejenigen, die fuer das Insert-Kommando gelten (siehe Pkt. 5.2.1.).

Beispiel fuer die Anwendung des Replace-Kommandos:

```
*R300
00300       ld a,b
*_

*R500:600;50
00500      do 80 I = 1,7
00550      Y(I) = ALOG(Y(I))
00600      80  CONTINUE
*_
```

Im zweiten Beispiel wurden die Zeilen im Bereich von 500 bis 600 geloescht. Sie werden durch drei neue Zeilen (500,550,600) ersetzt, wobei eine temporaere Schrittweite von 50 verwendet wird.
Ein Abschluss mit ⟨ENTER⟩ fuehrt zum automatischen Abschluss, da die Zeile 650 ausserhalb der Bereichsvereinbarung liegt.

5.2.4. Anzeige-Kommando auf Bildschirm oder Drucker

Anzeige-Kommando auf Bildschirm (Print)

Das Anzeige-Kommando zeigt die Zeilen auf dem Bildschirm des BC an. Das Format des Kommandos ist:

P⟨range⟩

Beispiele fuer das Anzeige-Kommando:

P.:700	Zeigt die Zeilen von der aktuellen Zeile bis zur Zeile 700 an.
P800:*	Zeigt alle Zeilen von der Zeile 800 bis zum Ende der Datei an.

Das Bedienen von ⟨ESC⟩ in der Kommandoebene bewirkt, dass die Zeile vor der aktuellen Zeile angezeigt wird. Die Eingabe von P ⟨ENTER⟩ bewirkt die Standardanzeige von 20 Zeilen ab der aktuellen Kursorposition.

Druck-Kommando (List)

Das Kommando List ist das Gleiche wie das Kommando Print, die Ausgabe erfolgt nur auf den Drucker. Das Format des Kommandos ist:

L⟨range⟩

5.2.5. Umnumerieren (Number)

Das Number-Kommando ordnet den Zeilen des Textes neue Nummern zu (Umnumerieren). Es kann notwendig sein, den Zeilen neue Nummern zuzuordnen, damit Platz zum Einfuegen geschaffen wird oder um Zeilennummern innerhalb der Datei organisiert zu ordnen. Das Format des Kommandos ist:

N[⟨start⟩][,⟨inc⟩|;⟨inc⟩][=⟨range⟩]

Es gilt:

- ⟨start⟩ ist die erste Nummer der neuen Folge. Von einer Startzeile, die keine Seitennummer spezifiziert, wird angenommen, dass sie sich auf der gleichen Seite befindet, auf der der Bereich (⟨range⟩) beginnt. Falls ⟨start⟩ weggelassen aber ⟨range⟩ einbezogen wurde, wird ⟨start⟩ auf die erste Zeile von ⟨range⟩ gesetzt. Wurden ⟨start⟩ und ⟨range⟩ weggelassen, aber ⟨inc⟩ einbezogen, wird ⟨start⟩ auf ⟨inc⟩ gesetzt.
  Wurde ⟨start⟩ weggelassen und ⟨inc⟩ einbezogen und wird durch ⟨range⟩ lediglich eine Seitennummer spezifiziert (z.B. =/2, siehe Pkt. 5.5.1.), wird ⟨start⟩ ebenfalls auf ⟨inc⟩ auf dieser Seite gesetzt. Fuer den Fall, dass ⟨start⟩, ⟨range⟩ und ⟨inc⟩ weggelassen wurden, wird ⟨start⟩ auf die Standardschrittweite eingestellt.

- ⟨inc⟩ ist die Schrittweite zwischen den Zeilennummern in der neuen Folge. Die Zusaetze fuer die Auswahl der Schrittweite sind die gleichen, wie die fuer das Insert-Kommando (siehe Pkt. 5.2.1.).

- (range) ist der Bereich von Zeilennummern, der umnumeriert werden soll. Falls (range) weggelassen wurde, wird die gesamte Datei umnumeriert.

Wurde die aktuelle Zeile umnumeriert, wird "." neu auf die gleiche physische Zeile eingestellt.
Wenn sich ein Number-Kommando auf Zeilennummern bezieht, die sich ausserhalb der Sequenz befinden, oder EDIT kann nicht alle Zeilen unterbringen, die die gegebene Schrittweite verwenden , wird die Fehlernachricht "Out of order" als Antwort gegeben.

Infolge des Speicherbedarfes von EDIT fuer das Ausfuehren eines Number-Kommandos kann es geschehen, dass ein Versuch, eine sehr grosse Datei umzunumerieren, zum Fehler "Insufficient memory" (unzureichender Speicherplatz) fuehrt.
In dieser Situation ist es empfehlenswert, dass ein kleinerer Abschnitt der Datei umnumeriert wird und dieser auf Diskette ausgegeben wird. Dann ist der naechste Abschnitt umzunumerieren und auf Diskette auszugeben usw. (siehe dazu das Write-Kommando im Pkt. 5.6.3.).

Beispiele fuer das Number-Kommando:

N7000;100=200:1000	Die "alten" Zeilen 200 bis 1000 sollen umnumeriert werden, so dass der neue Anfang bei 7000 und die Schrittweite bei 100 liegen soll, wobei N innerhalb der Datei liegen muss.
N,10=400:*	Die Zeilen 400 bis zum Ende der Datei werden umnumeriert, sie beginnen nun mit einer Schrittweite von 10 bei 400.
N9000=10000:*	Unter Verwendung der Standardschrittweite werden die Zeilen von 10000 bis zum Ende der Datei umnumeriert. Sie beginnen nun bei 9000.
N,100	Die gesamte Datei ist mit einer Schrittweite von 100 umzunumerieren.
N,5=2350!10	Dieses Kommando kann genutzt werden, wenn Platz fuer das Einfuegen geschaffen werden soll, indem die 10 Zeilen , die mit 2350 beginnen, in der Schrittweite von 5 verdichtet werden.

5.3. Aufbereiten innerhalb der Zeile - die Eingabebetriebsart

Die Kommandos zum Aufbereiten der Zeilen, soweit sie beschrieben wurden, ermoeglichen ein Aufbereiten durch Einfuegen, Loeschen oder Ersetzen ganzer Zeilen. Es gibt aber auch zahlreiche Situationen waehrend einer Aufbereitung, wo es

erforderlich ist, Veraenderungen in einer bereits existierenden Zeile vorzunehmen, wobei es aber nicht erforderlich ist, die gesamte Zeile neu einzugeben. Das Aufbereiten einer Zeile, ohne dass sie neu eingegeben werden muss, wird "intraline editing" (Aufbereiten innerhalb der Zeile) genannt, und dies wird in der Eingabebetriebsart ("Alter mode") durchgefuehrt.

5.3.1. Eingabebetriebsart (Alter)

Das Alter-Kommando wird in der Eingabebetriebsart benutzt. Das Format dieses Kommandos ist:

A(range)

In der Alter-Betriebsart muss die Zeilennummer der Zeile eingegeben werden, die geaendert werden soll, und es wird auf ein Subkommando der Eingabebetriebsart gewartet.

5.3.2. Subkommandos der Eingabebetriebsart

Die Subkommandos der Eingabebetriebsart werden zum Bewegen des Kursors benutzt, um nach einem Text zu suchen oder Text in eine Zeile einzufuegen, zu loeschen oder zu ersetzen. Die Subkommandos werden nicht auf dem Bildschirm angezeigt.

Zahlreiche Subkommandos der Eingabebetriebsart koennen vor dem Kommando selbst eine ganze Zahl stehen haben, die bewirkt, dass dieses Kommando so oft ausgefuehrt wird, wie es diese Zahl angibt (wenn keine ganze Zahl definiert wurde, lautet die Standardannahme immer 1).

In zahlreichen Faellen ist es moeglich, dass dem gesamten Kommando ein Minuszeichen (-) vorangestellt wird, das die Richtung der Wirkungsweise des Kommandos umwandelt.

Zum Beispiel:

D Loeschen des naechsten Zeichens, auf dem der Kursor steht.

6D Loeschen der naechsten 6 Zeichen ab Kursorposition.

-D Loeschen des Zeichens links vom Kursor.

-12D Loeschen von 12 Zeichen links vom Kursor.

Jedes Subkommando der Eingabebetriebsart ist unten beschrieben. Eine Zusammenfassung aller Subkommandos wird in Pkt.5.8. gegeben.

Beachte:
In der nachfolgenden Beschreibung repraesentiert das Zeichen "x" (ESC); (ch) repraesentiert ein beliebiges Zeichen, (text) eine Zeichenkette in einer willkuerlichen Laenge und i repraesentiert eine beliebige ganze Zahl.

5.3.2.1. Kursorposition

Die nachfolgend aufgefuehrten Kommandos oder Tasten der Tastatur des BC werden angewendet, wenn die Position des Kursors innerhalb der Zeile veraendert werden soll. Die Lage des Kursors wird als "aktuelle Position" bezeichnet.

(space) bewegt den Kursor nach rechts.
i(space) bewegt den Kursor um i Zeichen nach rechts.
-i(space) bewegt den Kursor um i Zeichen nach links.
Die Zeichen, ueber die der Kursor hinweggeht, werden angezeigt.

(bs) bewegt den Kursor nach links.
i(backspace) bewegt den Kursor um i Zeichen nach links.
Die uebergangenen Zeichen werden angezeigt.
i(backspace) ist aequivalent zu -i(space).
Fuer Tastaturen, die nicht ueber eine Rueckstelltaste verfuegen, hat das "^H" die gleiche Wirkung.

L zeigt den Rest der Zeile an und positioniert den Kursor an den Zeilenanfang. Er wird mit dem naechsten Subkommando der Eingabebetriebsart vorwaerts bewegt.

P zeigt den Rest der Zeile an und setzt den Kursor auf die aktuelle Position zurueck. Er wird mit dem naechsten Subkommando der Eingabebetriebsart vorwaerts bewegt.

W bewegt den Kursor zum Anfang des naechsten Wortes. Ein Wort ist als eine aneinandergrenzende Ansammlung von Buchstaben, Zahlen sowie Zeichen wie ".", "x" oder "%" definiert.
iW bewegt den Kursor ueber die naechsten i Worte hinweg.
-iW bewegt den Kursor ueber i Worte hinweg nach links.

5.3.2.2. Text Einfuegen (Insert text)

I fuegt Text ein. I(text)(ENTER) setzt den gegebenen Text ein, wobei bei der aktuellen Position begonnen wird.

B fuegt Leerzeichen (blanks) ab der aktuellen Position ein. Dem B-Kommando kann eine ganze Zahl vorangestellt werden, die angibt, wie viele Leerzeichen einzusetzen sind.
Die Leerzeichen werden ausschliesslich rechts vom Kursor eingefuegt.

G fuegt Zeichen ein.
iG(ch) setzt i Ausfertigungen von (ch) ein.

X erweitert eine Zeile. Das Subkommando X zeigt den Rest der Zeile an, geht zur Betriebsart "Text einfuegen" ueber und gestattet es, Text am Ende der Zeile einzufuegen.
Das Subkommando -X fuehrt dies am Anfang der Zeile aus (Betriebsart "Text Einfuegen"). Abschluss des Einfuegens erfolgt mit (ENTER).

5.3.2.3. Text Loeschen (Delete text)

D loescht das Zeichen an der aktuellen Position.
iD loescht i Zeichen ab der aktuellen Position nach rechts,
-iD loescht i Zeichen ab der aktuellen Position nach links.
Die geloeschten Zeichen werden von doppelten Schraegstrichen eingefasst.

H loescht den Rest der Zeile rechts vom Kursor (oder links vom Kursor bis zur 1. Position der Zeile, wenn -H eingegeben wurde) und geht zur Betriebsart "Text Einfuegen" ueber.
Das Texteinfuegen verlaeuft genauso, als waere ein I-Kommando eingegeben worden.

K loescht Zeichen.
K(ch) loescht alle Zeichen bis zu, aber nicht einschliesslich (ch) in der Zeile.
iK(ch) loescht alle Zeichen bis zum i- ten Auftreten von (ch).
-ik(ch) loescht alle Zeichen bis zum und einschliesslich des zum i-ten Mal auftretenden Zeichens (ch) nach links in der Zeile.
Wurde (ch) nicht gefunden, wird das Kommando nicht ausgefuehrt. Es wird ^G angezeigt.

O loescht Text.
O(text)x loescht den Text bis zu, aber nicht einschliesslich dem naechsten Auftreten von (text).
iO(text)x loescht den Text bis zum i-ten Auftreten von (text).
-iO(text)x loescht saemtlichen Text bis zum und einschliesslich des zum i-ten Mal auftretenden (text).

T loescht den Text des verbleibenden Teiles der Zeile rechts vom Kursor (oder aber links vom Kursor fuer den Fall, dass -T eingegeben wurde) und geht aus der Eingabebetriebsart heraus.

Z loescht Woerter.
iZ loescht die naechsten i Woerter rechts.
-iZ loescht i Woerter links vom Kursor.

5.3.2.4. Text Ersetzen (Replace text)

R ersetzt Text.
iR(text) loescht die naechsten i Zeichen und ersetzt sie durch (text).
-iR(text) ersetzt Text links vom Kursor.
Das Kommando ist mit (ENTER) abzuschliessen. Die geloeschten Zeichen werden zwischen rueckwaerts gerichteten Schraegstrichen wiedergegeben.

C tauscht Zeichen aus.
C(ch) ersetzt das Zeichen an der Kursorposition mit (ch).

5.3.2.5. Text Finden (Find text)

S sucht ein bestimmtes Zeichen.
S(ch) sucht das naechste Auftreten von (ch) ab aktueller Position, nach rechts und es positioniert den Kursor vor dieses Zeichen.
iS(ch) sucht das i-te Auftreten von (ch).
-S(ch) und -iS(ch) suchen das naechste bzw. zum i-ten Mal vorher aufgetretene (ch) und positionieren den Kursor unmittelbar davor. Das Zeichen, das sich an der Kursorposition befindet, wird nicht in die Suche einbezogen. Wurde (ch) nicht gefunden, wird das Kommando uebergangen.

F findet Text.
F(text)x findet das naechste Auftreten von (text) und positioniert den Kursor an den Anfang der Zeichenkette.
iF(text)x findet das i-te Auftreten von (text).
-F(text)x und -iF(text)x finden das (i-te Mal) vorherige Auftreten von (text) und positionieren den Kursor davor.

5.3.2.6. Beenden und Neustarten der Eingabebetriebsart

(ENTER) Zeigt den verbleibenden Rest der Zeile an, gibt die Aenderungen ein und bringt das Aendern auf dieser Zeile zum Abschluss.

A ist identisch zu (ENTER).

E gibt die Aenderungen ein und bringt das Aendern auf dieser Zeile zum Abschluss, es zeigt aber nicht den Rest der Zeile an.

N speichert die urspruengliche Zeile ab (die Aenderungen werden nicht uebernommen), und es wird entweder zur naechsten Textzeile gegangen (fuer den Fall, dass noch ein A(range)- Kommando in Bearbeitung ist), oder es erfolgt die Rueckkehr zur Kommandoebene.

Q speichert die urspruengliche Zeile ab (die Aenderungen werden nicht uebernommen), verlaesst die Eingabebetriebsart, und es erfolgt die Rueckkehr zur Kommandoebene.

^U Speichert die urspruengliche Zeile ab, verbleibt in der Eingabebetriebsart und positioniert den Kursor erneut an den Zeilenanfang.

5.3.2.7. Erweitern (Extent)

Das Extent-Kommando wird in der Kommandoebene angegeben und zum Erweitern von Zeilen verwendet. Das Format des Kommandos ist:

X(range)

Die Wirkung eines X-Kommandos ist aequivalent zur Eingabe eines A-Kommandos, welchem ein X-Subkommando nachgestellt wird. Nach der Eingabe eines X- Kommandos ist mit der Eingabe des Textes zu beginnen, der am Ende der Zeile eingefuegt werden soll.

Das Extent-Kommando ist besonders nuetzlich, wenn Kommentare in Assemblersprachprogramme eingefuegt werden sollen.

5.4. Finden und Substituieren

Wenn es notwendig ist, einen bestimmten Abschnitt eines Textes auszutauschen, wird nicht immer unmittelbar bekannt sein, wo sich innerhalb der Datei dieser Text befindet. Sogar fuer den Fall, dass eine tabellarische Uebersicht der Datei vorhanden ist, ist es eine ermuedende Aufgabe, die tabellarische Uebersicht der Datei zu durchsuchen, wenn die Zeilennummer einer speziellen Position gefunden werden soll, die auszutauschen ist.

Die Kommandos zum Finden und Substituieren von EDIT erlauben es dem Anwender, schnell den Text zu finden und die erforderlichen Aenderungen auszufuehren.

5.4.1. Finden (Find)

Das Find-Kommando lokalisiert eine gegebene Zeichenkette innerhalb der Datei und zeigt die Zeilen an, die diese Zeichenkette enthalten. Das Format des Kommandos ist:

F[(range)][,(limit)](ENTER)|*(Zeichenkette)*

* repraesentiert (ESC) und (limit) ist die Anzahl der Zeilen, die die Zeichenkette ((string)) enthaelt, die gefunden werden soll.
Die nachfolgend aufgefuehrten Regeln beziehen sich auf das Format des Find-Kommandos:

- Wenn *(Zeichenkette)* weggelassen wurde, wird die letzte Zeichenkette verwendet, die in einem Find-Kommando angegeben wurde.

- Wenn (limit) weggelassen und *(Zeichenkette)* einbezogen wurde, wird (limit) mit 1 angenommen.

- Wenn (limit) und *(Zeichenkette)* weggelassen wurden, wird die zuvor angewendete Grenze angenommen.

- Wenn (range) weggelassen und *(Zeichenkette)* einbezogen wurden, wird der gesamte Bereich von dem zuvor ausgefuehrten Find-Kommando angewendet.

- Wenn (range) und *(Zeichenkette)* weggelassen wurden, wird die Suche bezueglich der vorigen Zeichenkette von der Zeile ab fortgesetzt, bei der die Zeichenkette zuletzt gefunden wurde.

- War die Suche nicht erfolgreich, wird eine Fehlernachricht ausgegeben.

Beispiel fuer die Editierung mit einem Find-Kommando:

```
*F^:**WHI(I)*        Finde die erste Zeile, die WHI(I) enthaelt.

01100  WHI(I)=0      Anzeigen der Zeile 1100.

*F(ENTER)            Finden des naechsten Vorkommens.
01400  IF ...        Anzeigen der Zeile 1400.
*A.                  Fehler auf dieser Zeile .
01400  .             Fehler beseitigen.
       .
       .

*F,2*LD A,B*         Finde die ersten beiden Zeilen, die
01200 LD A,B         LD A,B enthalten (Bereich ist noch .:*).
01500 LD A,B         Anzeigen der Zeilen 1200 und 1500.
*A.                  Aktuelle Zeile.
01500  .             Aendern der Zeile 1500.
       .
```



```
*F.:**AVG*           Finde die erste Zeile in der Datei, die
                     AVG enthaelt.
Search fails         Es existiert keine solche Zeile.

*F*MEAN*             Versuche MEAN zu finden.
03700 MEAN=SUM/40    Anzeigen der Zeile 3700.

*F,0                 Finde alle anderen Zeilen, die MEAN enthalten (die Suche beginnt nach der Zeile
04200 MEAN=MEAN+1    3700).

06700 MEAN=SUM/M     Findet 2 mehr (4200 und 6700).
*A4200               Aendern der Zeile 4200 usw.
04200  .
       .
       .
```

5.4.2. Substituieren (Substitute)

Das Substitute-Kommando lokalisiert eine gegebene Zeichenkette, ersetzt sie durch eine neue Zeichenkette und zeigt die neuen Zeile(n) an. Das Format des Kommandos ist:

S[(range)][,(limit)](ENTER)|*(alte Zeichenkette)*(neue Zeichenkette)*

* repraesentiert (ESC), und (limit) ist die Nummer der Zeile ((position)), in der (alte Zeichenkette) durch (neue Zeichenkette) zu ersetzen ist.

Eine Grenze von Null wird saemtliche Vorkommen von (alte Zeichenkette) durch (neue Zeichenkette) ersetzen. (neue Zeichenkette) kann eine Nullzeichenkette sein und fuehrt damit automatisch zum Loeschen von (alte Zeichenkette).
Es gelten die folgenden Regeln fuer das Format des Substitute-Kommandos:

- Wenn *(alte Zeichenkette)*(neue Zeichenkette)* weggelassen wurden, werden die im letzten Substitute-Kommando angegebenen Zeichenketten benutzt.

- Wenn (limit) weggelassen und *(alte Zeichenkette) *(neue Zeichenkette)* einbezogen wurden, wird (limit) als Null angenommen.

- Wenn (limit) und *(alte Zeichenkette)*(neue Zeichenkette)* weggelassen wurden, wird die vorherige Grenze angenommen.

- Wenn (range) weggelassen wurde und *(alte Zeichenkette) *(neue Zeichenkette)* einbezogen wurden, wird der gesamte Bereich des vorherigen Substitute-Kommandos verwendet.

- Wenn (range) und *(alte Zeichenkette)*(neue Zeichenkette)* weggelassen wurden, wird die Substitution an der Stelle

fortgesetzt, an der die letzte Substitution aufgehoert hatte.

Falls <alte Zeichenkette> nicht gefunden wurde, wird eine Fehlernachricht ausgegeben.

Beispiel:

```
*S^:5000^ALPHA^BETA^          Von der ersten Zeile bis zur
                              Zeile 5000 ist ALPHA durch BETA
                              zu ersetzen.
00950 BETA=500                EDIT zeigt die Zeilen an, in
01750 WRITE BETA(K)           denen Substitutionen durchge-
04100 IF BETA=100 GOTO 9000   fuehrt wurden.
```

5.5. Seiten

Es ist moeglich eine EDIT-Datei in Abschnitte zu unterteilen, die man als Seiten bezeichnet. Diese Seiten werden durch Seitenmarken getrennt.

Die erste Seite einer Datei ist immer die Seite 1, und EDIT fuehrt die Kommandoebene immer auf der Seite 1 einer aus mehreren Seiten bestehenden Datei ein.
Jede nachfolgende Seite beginnt mit einer Seitenmarke und ist sequentiell numeriert. Auf einer beliebig angegebenen Seite kann der vollstaendige Bereich von Zeilennummern (00000 bis 99999 oder ein Teilbereich davon) verwendet werden.
Wenn EDIT einen Formularvorschub realisiert, waehrend in der Datei gelesen wird, wird an dieser Stelle in der Datei eine Seitenmarke eingefuegt. Realisiert EDIT eine Zeilennummer die, niedriger ist als die vorhergehende Zeilennummer, wird automatisch eine Seitenmarke eingesetzt, so dass die ordnungsgemaesse Reihenfolge der Zeilennummern gewaehrleistet bleibt. Schreibt EDIT eine Datei auf die Diskette, wird ein Formularvorschub mit jeder Seitenmarke ausgegeben. Dann wird jede neue Seite der Datei auf einer neuen physischen Seite gestartet, wenn die Datei gedruckt wird.

5.5.1. Spezifizieren der Seitennummern

In einer aus einer Seite bestehenden Datei wird lediglich eine Zeilennummer benoetigt, wenn die Zeile angezeigt werden soll.
In einer aus mehreren Seiten bestehenden Datei muss EDIT die Seitennummer ebenso wie die Zeilennummer kennen, wenn es die <position> bestimmen soll.
Das heisst, die <position> wird angezeigt durch

<zeile>[/<seite>],

wobei <zeile> ".", "^", "*" oder eine aus bis zu 5 Stellen bestehende Zahl sein kann.

<seite> ist ".", "^", "*" oder eine aus bis zu 5 Stellen bestehende Zahl.
Wird eine Seite spezifiziert, beziehen sich die Zeichen ".", "^" und "*" auf die aktuelle Seite, auf die erste bzw. letzte Seite. Falls <seite> weggelassen wurde, wird die aktuelle Seite zugrundegelegt.
Folglich kann in einer aus mehreren Seiten bestehenden Datei ein <range>, das gekennzeichnet sein kann durch

<position>:<position>

oder

<position>!<number>

ebenfalls Seitennummern enthalten.

Wurde die Seitennummer der ersten Zeilennummer in dem Bereich weggelassen, wird angenommen, dass dies die laufende Seite betrifft.
Falls die Seitennummer der zweiten Zeilennummer in dem Bereich weggelassen wird, wird vorausgesetzt, dass diese sich auf der gleichen Seite wie die erste Zeilennummer in dem Bereich befindet.

Einige Beispiele von Zeilennummern und von Bereichen, die eine Seitennummernspezifikation einschliessen:

P100/2:*/*	Zeile 100 auf der Seite 2 bis zur letzten Zeile auf der letzten Seite
P100/2:*	Zeile 100 auf der Seite 2 bis zum Ende von dieser Seite
P100:*/5	Zeile 100 auf der laufenden Seite bis zur letzten Zeile auf der Seite 5
P100/*	Zeile 100 auf der letzten Seite
P100/.:*/3	Zeile 100 auf der laufenden Seite bis zur letzten Zeile auf der Seite 3

Siehe dazu auch Pkt. 5.9. fuer weitere Beispiele der Bereichsspezifikation.

5.5.2. Einsetzen von Seitenmarken

Die Seitenmarken koennen in die Datei nach Ermessen des Anwenders eingefuegt werden. Wenn eine Seitenmarke eingefuegt werden soll, ist das Kommando Mark (Marke) zu verwenden. Das Format des Kommandos ist:

M<position>

Die Seitenmarke wird unmittelbar nach <position> eingefuegt. <position> muss existieren, sonst wird eine Fehlernachricht ausgegeben.
Die Bezugnahme auf die laufende Zeile (".") bleibt erhalten, nachdem ein Mark-Kommando ausgefuehrt wurde. Das heisst, wenn <position> vorhanden war, bevor "." ausgegeben wurde, wird "." zur naechsten Seite transportiert und noch auf der gleichen physischen Zeile angezeigt.

5.5.3. Loeschen von Seitenmarken

Seitenmarken koennen mit H'lfe des Kommandos K (Kill) geloescht werden. Das Format des Kommandos ist:

K/<seite>

Das K-Kommando loescht die Seitenmarke, die nach <seite> kommt. Beispielsweise wuerde in einer aus 4 Seiten bestehenden Datei K/2 die zweite Seitenmarke geloescht (das ist die Seitenmarke, die als Startpunkt der Seite 3 steht), und die Seiten wuerden dann mit 1, 2 und 3 numeriert. Die letzte Zeilennummer auf <seite> muss niedriger als die erste Zeilennummer auf <seite>+1 sein, bevor ein K/<seite>-Kommando ausgefuehrt werden kann.

5.5.4. Anfang einer Seite

Wenn an den Anfang einer Seite zurueckgekehrt werden soll, ist das Kommando Begin zu verwenden. Das Format des Kommandos ist:

B[/<seite>]

Falls <seite> weggelassen wurde, vollzieht das Begin-Kommando die Rueckkehr an den Anfang der Seite eins.

5.5.5. Zusaetzliche Kommandos und Seitenmarken

1. Ein Loeschkommando, das ueber eine Seitengrenze hinweggeht, loescht alle Zeilen innerhalb des Bereiches; es loescht aber nicht die Seitenmarke.

2. Ein Anzeige-(Print-)Kommando, das sich aus der aktuellen Seite herausbewegt, zeigt zuerst die neue Seitennummer an, bevor die erste Zeile angezeigt wird, die im Kommando spezifiziert wurde.

3. Wenn die Ausgabe mit Hilfe des Kommandos "List" ausgefuehrt werden soll, wird mit jeder Seitenmarke ein Formularvorschub ausgefuehrt und es wird die Seitennummer auf jeder Seite gedruckt.

4. Ein Bereich, der mit Hilfe eines Ausrufezeichens spezifiziert wurde, kann auch mehrere Seiten lang sein.

5. Wurde der Bereich mit Hilfe eines Number-Kommandos spezifiziert und geht der Bereich ueber Seitengrenzen hinweg, wird die Numerierung auf jeder Seite neu gestartet; die erste Zeilennummer wird gleich der Schrittweite sein. Folglich muessen sich das Kommando Number <start> und die erste Zeile von <range> auf der gleichen Seite befinden.

5.6. Beenden von EDIT

Der Pkt 5.1.3. hat die Kommandos Exit und Quit fuer das Beenden von EDIT eingefuehrt. Die beiden Kommandos werden in diesem Kapitel ausfuehrlicher beschrieben. Es wird ebenfalls noch ein zusaetzliches Kommando, das Kommando Schreiben (Write) vorgestellt.

5.6.1. Exit

Das Exit-Kommando wird zum Schreiben der Datei auf die Diskette und zur Rueckkehr zum Systemgrundzustand verwendet. Das Format des Kommandos ist:

E[d:][<dateiname>][/<schalter>]

Ein E-Kommando, das nicht ueber ein Argument verfuegt, schreibt die editierte Datei unter ihrer urspruenglichen Dateibezeichnung auf die Diskette und rettet eine Kopie der urspruenglichen Datei unter dem alten Dateinamen mit dem Dateityp ".BAK".

Das wahlweise anwendbare d:, wobei d ein Laufwerk kennzeichnet, wird dann benutzt, wenn ein anderes Laufwerk zugewiesen werden soll. Diese Zuweisung hat mit Grossbuchstaben zu erfolgen.
<dateiname> ist ebenfalls wahlweise anzuwenden, es wird benutzt, die editierte Datei unter einem neuen Namen auf die Diskette zu schreiben.
Wurde ein neuer Dateiname spezifiziert, wird keine BAK-Datei erzeugt, weil die Originaldatei noch unter ihrer urspruenglichen Dateibezeichnung existiert.
Unter SCP wird fuer den Fall, dass ein Dateiname ohne Dateityp nach Verlassen von EDIT spezifiziert wurde, der Dateityp verwendet, der bei Beginn von EDIT angegeben wurde (dies wird ".MAC" sein, wenn kein anderer Dateityp angegeben wurde). Der wahlweise anwendbare <schalter> steuert das Format der Ausgabe (siehe Pkt. 5.6.5.).

5.6.2. Quit

Das Quit-Kommando wird zur Rueckkehr zum Systemgrundzustand verwendet, ohne dass die editierte Datei auf die Diskette geschrieben wird. Das Format des Kommandos ist:

Q[<dateiname>]

Quit rettet nur die Originalkopie der Datei. Der Dateiname kann geaendert werden, indem <dateiname> in das Quit-Kommando einbezogen wird.

5.6.3. Schreiben (Write)

Das Write-Kommando schreibt die editierte Datei auf die Diskette und kehrt in die EDIT-Kommandoebene zurueck. Es bewirkt nicht das Verlassen des Editors, die aktuelle Position in der Zeile wird nicht veraendert. Das Format des Kommandos ist:

W[d:][<dateiname>][/<schalter>]

d: wird angewendet, wenn ein anderes Laufwerk zugewiesen werden soll, wobei d ein Grossbuchstabe sein muss.
Falls <dateiname> spezifiziert wurde, wird die editierte Datei unter ihrem neuen Namen auf die Diskette geschrieben. Ein nachfolgendes E- oder W-Kommando (ohne einen Dateinamen) rettet die neue Datei unter dem neuen Dateinamen, der mit W spezifiziert wurde. Die Datei vom letzten W-Kommando wird unter diesem Namen mit dem Dateityp ".BAK" gerettet.

Der wahlweise anwendbare <schalter> steuert das Format der Ausgabe (siehe Pkt. 5.6.5.).

5.6.4. Indexdateien

Wenn in einer Datei, die editiert werden soll, gelesen wird, erzeugt EDIT die Information, die es bezueglich eines jeden Blockes der Diskettendatei benoetigt. Bei einer kleinen Datei wird diese Information innerhalb weniger Sekunden erzeugt, immer dann, wenn die Datei eingelesen wird. Bei groesseren Dateien (5 K oder mehr) steigt der Zeitbedarf fuer das Einlesen der Datei betraechtlich.
Deshalb rettet EDIT fuer den Fall, dass es eine Datei von 42 oder mehr Saetzen auf die Diskette schreibt, ebenfalls eine andere, kleinere, Datei, die von der Textdatei gesondert zu betrachten ist und die die benoetigte Information bezueglich der Textdatei enthaelt.

Diese kleine Datei wird Indexdatei genannt, sie kann schneller als die Textdatei eingelesen werden. EDIT rettet die Indexdatei unter dem gleichen Dateinamen, wie die Textdatei, wobei

den ersten beiden Buchstaben des Dateityps noch ein *-Zeichen vorangestellt wird. So wird beispielsweise fuer eine mit FOO.MAC bezeichnete Textdatei die Indexdatei mit FOO.*MA benannt.

Wenn EDIT beauftragt wird, eine Datei zu editieren, wird als erstes die Indexdatei ueberprueft. Wenn eine Indexdatei existiert, liest EDIT die Indexdatei anstelle der Textdatei. Dabei muss beachtet werden, ob die Textdatei bereits durch einen anderen EDITOR modifiziert oder in BASIC geaendert und gerettet wurde. In einem solchen Fall muss der Anwender die Indexdatei loeschen, bevor die Textdatei wieder mit EDIT editiert werden kann. Wurde die Indexdatei nicht geloescht, enthaelt EDIT eine fehlerhafte Information bezueglich der Textdatei.

5.6.5. Schalter

Wenn EDIT eine Datei einliest, wird erwartet, dass diese Zeilennummern besitzt. Enthaelt die Datei keine Zeilennummern, fuegt EDIT Zeilennummern hinzu und versucht, die Datei auf das Standardformat von EDIT umzuwandeln.
Es gibt jedoch eine Reihe verschiedener anderer Dateiarten, die EDIT fuer den Fall akzeptiert, dass der richtige Schalter ordnungsgemaess an den Dateinamen angehaengt wurde. Den Schaltern ist immer ein Schraegstrich (/) vorangestellt:

dateiname.dateityp[/schalter]

5.6.5.1. BASIC-Schalter

/BASIC

Wenn der BASIC-Schalter an den Eingabedateinamen angehaengt wurde, liest EDIT die Datei nach folgendem Algorithmus:

1. Alle vorangestellten Leerzeichen und Tabulatoren werden von jeder Zeile entfernt.

2. Das erste Zeichen, das kein Leerzeichen ist, muss eine Ziffer sein.

3. Die ersten fuehrenden 5 Stellen werden zu einer Zeilennummer umgewandelt. Wenn mehr als 5 fuehrende Stellen vorhanden sind, ergibt dies einen Fehler (fatal error).

4. Ein Tabulator wird eingefuegt, wenn das erste Zeichen keine Ziffer, Leerzeichen oder Tabulator ist. Ist das erste Zeichen, das keine Ziffer ist, ein Leerzeichen, wird es durch einen Tabulator ersetzt. Ist das erste Zeichen, das keine Ziffer ist, ein Tabulator, bleibt dieser erhalten.

5. Bei der Ausgabe werden fuer den Fall, dass UNSEQ (siehe Pkt. 5.6.5.2.) ausgewaehlt wurde, die fuehrenden Nullen in der Zeilennummer unterdrueckt, der Tabulator wird in ein Leerzeichen umgewandelt.

Auf Grund der Tatsache, dass BASIC die Zeilennummern verwendet, um die Reihenfolge der Befehle zu steuern, sollten die Anwender von BASIC vor einem Umnumerieren mit Hilfe des N-Kommandos gewarnt sein. Das BASIC-Programm ignoriert die Seitenmarken der EDIT-Datei. Auf diese Weise hat eine BASIC-Datei Mehrfachseiten. Es ist zu gewaehrleisten, dass keine Zeilennummer mehr als einmal im Programm erscheint.

5.6.5.2. Schalter SEQ und UNSEQ

/SEQ oder /UNSEQ

Falls der Schalter SEQ an den Eingabedateinamen angehaengt wurde, wendet EDIT den gleichen Algorithmus an, um die Textdatei zu interpretieren, wie es bei BASIC der Fall ist. Wenn die Datei ausgegeben wird, befindet sie sich in dem standardisierten EDIT-Format, falls nicht der Schalter UNSEQ an den Ausgabedateinamen angehaengt wurde.

Der Schalter UNSEQ an der Eingabe teilt EDIT mit, dass es seine eigenen Zeilennummern an die einzugebende Datei anzuhaengen hat, ungeachtet dessen wie sie aussieht. Dieser Schalter muss dann verwendet werden, wenn die ankommende Datei am Anfang der Zeilen Stellen hat, wobei die Bits mit der hoechsten Wertigkeit nicht als Zeilennummern interpretiert werden duerfen.

Bei der Ausgabe muss der Schalter UNSEQ angegeben werden (fuer den Fall, dass dies nicht bereits geschehen ist), wenn eine nichtstandardisierte Datei ausgegeben werden soll.
Das bedeutet, dass die Datei im BASIC-Format dann ausgegeben wird, wenn BASIC bei der Eingabe und UNSEQ bei der Ausgabe spezifiziert wurde.
Falls BASIC nicht bei der Eingabe spezifiziert wurde, aber UNSEQ bei der Ausgabe, wird die Datei ausgegeben, ohne dass Zeilennummern realisiert und ohne dass Tabulatoren nachgezogen werden. Wenn bei der Eingabe der Schalter UNSEQ angegeben wurde und der Anwender wuenscht, dass eine Standarddatei ausgegeben wird, wird sich der Schalter SEQ, wenn er bei der Ausgabe angegeben wird, ueber den Schalter UNSEQ hinwegsetzen und bestimmend sein.

5.7. Alphabetische Zusammenstellung der Kommandos

Kommando	Format und Bedeutung
Alter	A⟨range⟩ Schaltet die Eingabebetriebsart ein.
Begin	B[⟨/seite⟩] Verschiebt den Anfang von ⟨seite⟩; Standardannahme ist die Seite 1.
Delete	D⟨range⟩ Loescht Zeilen.
Exit	E[d:][⟨dateiname⟩][/⟨schalter⟩] Schreibt den editierten Text auf die Diskette und verlaesst den Editor.
Find	F[⟨range⟩][,⟨limit⟩]⟨ENTER⟩|x⟨zeichenkette⟩x Findet vorkommende Ereignisse von ⟨zeichenkette⟩.
Insert	I[⟨position⟩[,⟨inc⟩|;⟨inc⟩]] Fuegt Zeilen ein, wobei bei ⟨position⟩ begonnen wird und die Schrittweite ⟨inc⟩ verwendet wird. Falls kein Argument angegeben ist, wird mit dem vorherigen Insert-Kommando fortgesetzt.
Kill	K/⟨seite⟩ Loescht die Seitenmarke am Ende von ⟨seite⟩.
List	L⟨range⟩ Druckt die Zeilen auf dem Drucker aus.
Mark	M⟨position⟩ Fuegt eine Seitenmarke nach ⟨position⟩ ein.
Number	N[⟨start⟩][,⟨inc⟩|;⟨inc⟩][=⟨range⟩] Versieht die Zeilen, welche sich innerhalb von ⟨range⟩ befinden, mit neuen Nummern, so dass sie bei ⟨start⟩ beginnen und die Schrittweite bei ⟨inc⟩ liegt.
Print	P⟨range⟩ Zeigt die Zeilen auf dem Bildschirm des BC an. Wurde kein Argument angegeben, zeigt es 20 Zeilen ab der aktuellen Position an (Standard).
Quit	Q[⟨dateiname⟩] Verlaesst den Editor, ohne dass der editierte Text auf die Diskette ausgegeben wird.

Kommando	Format und Bedeutung
Replace	R⟨range⟩[,⟨inc⟩|;⟨inc⟩] Ersetzt eine oder mehrere Zeilen mit der Schrittweite ⟨inc⟩.
Substitute	S[⟨range⟩][,⟨limit⟩][⟨ENTER⟩]|*⟨alte Zeichenkette⟩*⟨neue Zeichenkette⟩* Tauscht die alte Zeichenkette gegen die neue aus.
Write	W[d:][⟨dateiname⟩][/⟨schalter⟩] Schreibt den editierten Text auf die Diskette, verlaesst aber den Editor nicht.
Extent	X⟨range⟩ Gestattet das Einfuegen von Text am Zeilenende

5.8. Alphabetische Zusammenstellung der Subkommandos der Eingabebetriebsart (Alten)

Kommando	Format	Wirkung
A	A	Zeigt den verbleibenden Rest der Zeile an, gibt die Aenderungen ein und schliesst die Aenderung dieser Zeile ab.
B	[i]B	Setzt Leerzeichen ein.
C	C⟨ch⟩	Ersetzt Zeichen.
D	[-][i]D	Loescht Zeichen.
E	E	Gibt die Aenderungen ein, schliesst die Aenderung und Eingabe dieser Zeile ab.
F	[-][i]F*⟨text⟩*	Findet ⟨text⟩.
G	[i]G⟨ch⟩	Setzt i mal ⟨ch⟩ ein.
H	[-]H⟨text⟩	Loescht den Rest der Zeile und schaltet die Insert-Betriebsart ein.
I	I⟨text⟩	Fuegt ⟨text⟩ ein.
K	[-][i]K⟨ch⟩	Loescht alle Zeichen bis zu ⟨ch⟩.
L	L	Positioniert den Kursor an den Zeilenanfang.

Kommando	Format	Wirkung
N	N	Speichert die urspruengliche Zeile wieder ein und fuehrt entweder die Bewegung zur naechsten Zeile aus, wenn ein A⟨range⟩-Kommando noch in Bearbeitung ist, oder kehrt zur Kommandoebene zurueck.
O	[-][i]O⟨text⟩*	Loescht alle Zeichen bis ⟨text⟩.
P	P	Fuehrt den Kursor auf die aktuelle Position zurueck.
Q	Q	Verlaesst die Eingabebetriebsart und speichert die Zeile in ihrer urspruenglichen Form wieder ab.
R	[-][i]R⟨text⟩	Ersetzt i Zeichen mit ⟨text⟩.
S	[-][i]S⟨ch⟩	Findet ⟨ch⟩.
T	[-]T	Loescht den verbleibenden Rest der Zeile, schliesst die Aenderung der Zeile ab.
W	[-][i]W	Bewegt den Kursor ueber Worte hinweg.
X	[-]X	Erweitert die Zeile.
Z	[-][i]Z	Loescht Worte.
[i]⟨bs⟩		Bewegt den Kursor rueckwaerts ueber Zeichen hinweg.
[-][i]⟨space⟩		Bewegt den Kursor ueber Zeichen hinweg.
⟨ENTER⟩		Zeigt den Rest der Zeile an, gibt die Aenderungen ein und schliesst die Aenderungen und Eingabe der Zeile ab.
^U		Speichert die Zeile in ihrer urspruenglichen Form wieder ab, verbleibt in der Eingabebetriebsart und positioniert den Kursor wieder an den Anfang der Zeile.

5.9. Steuerzeichen von EDIT

^C Unterbricht das aktuelle Kommando und kehrt in die Kommandoebene von EDIT zurueck. Gilt nicht fuer Einfuegekommandos!

^H Bewirkt einen Rueckschritt (im Gegensatz zu BASIC werden dabei keine Zeichen geloescht).
Es kann z.B. dafuer verwendet werden, einen Rueckschritt ueber die Zeichen hinweg auszufuehren, bevor ein Unterstreichen ausgefuehrt wird.

^I Realisiert einen Tabulator.

^O Unterbricht/nimmt wieder die Ausgabe (auf dem Bildschirm des BC oder Drucker) auf, von einem EDIT-Kommando aus.

^S Stoppt/startet erneut das Ausfuehren des Kommandos.

^U Loescht die Zeile, die ueber Tastatur eingegeben wurde und laesst sie "darueberstarten".
Falls es in der Eingabebetriebsart angewendet wird, speichert das ^U die Zeile in ihrer urspruenglichen Form wieder ab, es verbleibt in der Eingabebetriebsart und positioniert den Kursor wieder an den Zeilenanfang.

5.10. Fehlermeldungen

5.10.1. Allgemeine Fehler

- Disk full: Diskette ist voll.

Zeigt an, dass auf der Diskette kein Speicherplatz mehr vorhanden ist, um die Datei abzuspeichern.

- Directory full: Verzeichnis ist voll.

Diese Fehlernachricht zeigt an, dass auf der Diskette im Verzeichnis kein weiterer Platz mehr vorhanden ist.

- Disk output error: Diskettenausgabefehler (Schreibfehler)

Diese Fehlernachricht zeigt an, dass ein Fehlerverhalten in der Diskettenausgaberoutine zu verzeichnen ist, das entweder auf Hardwareprobleme oder auf Probleme im Betriebssystem zurueckzufuehren ist. Allgemein ist darunter ein Schreibfehler zu verstehen.

- Illegal line format: Unerlaubtes Zeilenformat.

Diese Fehlernachricht tritt auf, wenn EDIT eine Zeile mit fremdem Inhalt oder mit einer fremden Zeilennummer findet.

5.10.2. EDIT-Fehlermeldungen

- Illegal command: Unerlaubtes Kommando.

Teilt dem Anwender mit, dass ein Kommando nicht existiert oder fehlerhaft eingegeben wurde.

- Insufficient memory available: Speicherplatz nicht ausreichend.

Diese Fehlernachricht tritt auf, wenn der Anwender viele Aenderungen ausgefuehrt hat, die den fuer EDIT verfuegbaren Speicherplatz erschoepfen. Es sollte ein W-Kommando ausgefuehrt werden, um den Speicher zu verdichten.

- No string given: Keine Zeichenkette eingegeben.

Teilt dem Anwender mit, dass ein F- oder ein S-Kommando erteilt wurde ohne eine Suchzeichenkette zu spezifizieren.
Dies tritt auf, wenn F- oder S-Kommandos angewendet werden, in denen keine Argumente spezifiziert wurden.

- No such line(s): Keine derartige(n) Zeile(n).

Diese Fehlernachricht wird fuer den Fall ausgegeben, dass sich ein Kommando auf eine Zeile oder auf einen Bereich von Zeilen bezieht, die nicht vorhanden sind.

- Line too long: Zeile zu lang.

Diese Fehlernachricht wird ausgegeben, wenn der Anwender versucht, eine Zeile einzugeben, die laenger als 255 Zeichen ist.

- Out of order: Reihenfolge gestoert (nicht in Ordnung).

Diese Fehlernachricht zeigt an, dass sich die Zeilennummern in der Datei nicht in der natuerlichen aufsteigenden Reihenfolge befinden. Dies ereignet sich dann, wenn versucht wird, an solchen Stellen einzufuegen, an denen nicht genuegend Raum dafuer vorhanden ist oder wenn versucht wird, eine Seitenmarke zu streichen.

- Search fails: Suche nicht erfolgreich.

Diese Fehlernachricht teilt mit, dass ein Suchlauf nicht erfolgreich gewesen ist.

- Wrap around: Zeilennummer zu gross.

Diese Fehlernachricht wird dann ausgegeben, wenn eine Zeile erzeugt wird, deren Nummer groesser als 99 999 ist.

5.10.3. Dateifehler

- File already exists: Datei existiert bereits.

Diese Fehlernachricht wird dann ausgegeben, wenn der Anwender versuchen sollte, einer Datei den Namen einer bereits existierenden Datei zu geben oder wenn er versucht, eine Datei mit einem Namen einer bereits existierenden Datei in einem E- oder in einem W-Kommando umzubenennen.

- File not found: Datei nicht gefunden.

Diese Fehlernachricht wird ausgegeben, wenn in einem Kommando eine Datei spezifiziert ist, die nicht gefunden wurde.

- Illegal file specification: Nicht erlaubte Dateispezifikation.

Diese Fehlernachricht informiert den Anwender darueber, dass die Kommandozeichenkette ein falsches Zeichen enthaelt.

5.11. Ausgabedateiformat

Compiler und Assembler sollten die Zeilennummern und die Seitenmarken ignorieren, die in die Ausgabedateien von EDIT einbezogen sind (mit der Ausnahme, dass sie in Druckdateien integriert sind). BASI, FORTRAN und ASM verhalten sich so. Eine Zeilennummer besteht aus 5 Dezimalstellen, welchen ein Tabulatorzeichen (^I) folgt. Bei allen sechs Bytes ist das hoechste Bit (das Bit 7) gleich einer Eins.
Es ist nicht empfehlenswert, EDIT-Dateien mit Hilfe des SCP-Kommandos TYPE zu drucken. Dabei koennten verstuemmelte Zeichen in den Zeilennummern erscheinen. Statt dessen sollten die Kommandos PRINT und LIST von EDIT angewendet werden.
Eine Seitenmarke ist ein Zeichen eines Formularvorschubes (^I), wobei das Bit der hoechsten Stellenwertigkeit gleich eins ist.

6. LINK 1520 (8CPX)

Um ein in der Assemblersprache geschriebenes Programm unter dem Betriebssystem SCP lauffaehig zu machen, muessen sowohl der Assembler ASM als auch der Programmverbinder LINK benutzt werden.

Der Prozess der Programmentwicklung laeuft in vier Schritten ab:

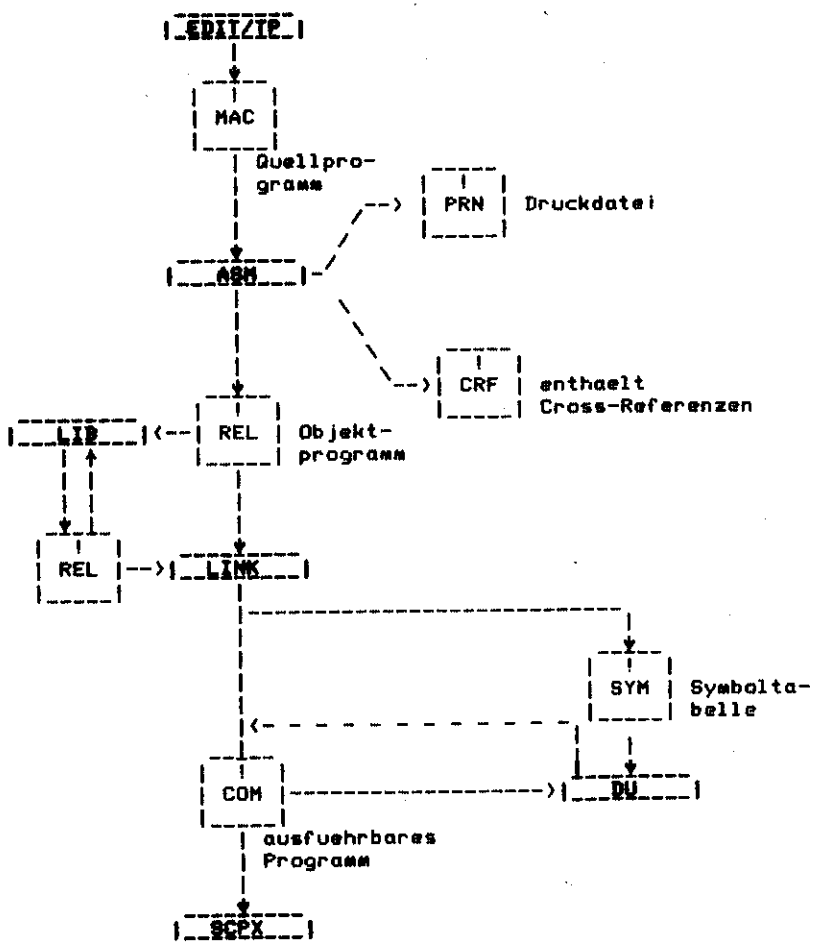
1. Erfassen des in Assemblersprache geschriebenen Programmes unter Benutzung eines Editors (EDIT, TP).

2. Uebersetzen des Quellprogrammes mit dem Assembler ASM, das Ergebnis ist ein Programm (Objektprogramm) in einem dem Maschinencode nahen, aber nicht abarbeitbaren Zwischencode.

3. Laden und Verbinden einzeln uebersetzter Programme ("Moduln") zu einem Gesamtprogramm mit dem Programmverbinder LINK; LINK wandelt die Objektprogramme, die sich auch in einer mit dem Programm LIB gebildeten Bibliothek befinden koennen, zu einem einzigen Programm im Maschinencode um, das unter SCP ausgefuehrt werden kann.

4. Test des lauffaehigen Programmes mit dem Testhilfsprogramm DU.

Nachfolgendes Bild verdeutlicht diesen Ablauf:



Die folgende Beschreibung der Arbeitsweise des Programmverbinders LINK verwendet Begriffe, die im Teil II der Anleitung fuer den Programmierer erklaert sind. Die dort benutzte Syntax-Notation wurde beibehalten.

Die vom Assembler erzeugten Programme sind nicht ausfuehrbar. Um ein REL-Programm ausfuehrbar zu machen, muss es mit dem Programmverbinder LINK geladen und gebunden werden.

Laden bedeutet die physische Ablage des Programmes im Speicher und das Umwandeln von Relativadressen in Absolutadressen. Dies ist einer der notwendigen Schritte, um ein verschiebliches Objektprogramm (REL) in ein ausfuehrbares Programm (COM) zu wandeln.

Binden heisst, dass jedes Programm, das externe Bezuege besitzt (durch ein CALL, ein externes Symbol oder ein INCLUDE) mit dem entsprechenden Programm (Modul) "verbunden" wird.

LINK kann das uebersetzte und gebundene Programm als ein ausfuehrbares Programm (Dateityp .COM) auf Diskette speichern. LINK belegt einen Speicherbereich von 16 K Byte.

Bemerkung:
Fuer die vom Programmverbinder verarbeiteten REL-Dateien bzw. die erzeugte .COM-Datei werden im Text die Bezeichnungen "Programm" bzw. "Modul" verwendet.

6.1. Aufruf des Programmverbinders LINK

Durch die Tastatureingabe von

```
LINK
```

wird das Programm LINK.COM geladen. LINK zeigt durch einen Stern (*) an, dass es eine Kommando-Eingabe erwartet. Die zu verbindenden REL-Programme muessen im Direktzugriff zur Verfuegung stehen. Ist nur ein Laufwerk verfuegbar und die REL-Programme sind auf mehreren Disketten gespeichert, so muss ein Wechsel der Disketten vor den entsprechenden Lade-Schritten von LINK erfolgen.

6.2. LINK-Kommandos

LINK-Kommandos sind Programmnamen und Schalter. Man kann die LINK-Kommandos nacheinander einzeln eingeben oder alle Kommandos (einschliesslich des Aufrufes des Programmverbinders LINK selbst) in einer Zeile.

Eine Kommando-Zeile hat ein flexibles Format, es koennen wahlweise Angaben fuer das Laden und Verbinden der Programme und fuer andere Operationen eingegeben werden.

Die Grundregel ist dabei, dass die Programme in der Reihenfolge geladen werden, in der sie angefuehrt sind, beginnend ab der Startadresse 0103H. Selbst wenn die Programme in der eingegebenen Reihenfolge geladen werden, braucht man sie nicht in der Folge ihrer Abarbeitung anzugeben, denn LINK erzeugt an der Stelle 0100H-0102H einen unbedingten Sprung an den logischen Programmanfang.

LINK kann ueber 11 verschiedene Aufgaben ausfuehren, aber in der praktischen Arbeit werden i. a. nicht mehr als drei oder vier gleichzeitig benoetigt.

Nach der Eingabe eines LINK-Kommandos meldet sich LINK mit einem Stern (*) und zeigt an, dass es eine weitere Kommando-Eingabe erwartet.

Beispiel:

```
A)LINK                  (Kommandos durch (ENTER) beendet)
*/(schalter)
*(programmname)
*/(schalter)
*/E                     (Verlassen von LINK)
```

Alle Kommando-Zeilen koennen als eine Zeile eingegeben werden:

```
LINK /(schalter),(programmname)/(schalter)/E
```

Obwohl die Eingabe eines jeden Kommandos als gesonderte Zeile zeitaufwendig ist, hat sie den Vorteil, dass man zu jedem Zeitpunkt weiss, was LINK gerade durchfuehrt.

6.2.1. Programmnamen

Die von LINK verarbeiteten Programme sind REL-Programme. Ein Programmname bewirkt, dass LINK das gesamte Programm (Modul) laedt. Wenn vorher schon ein Programm geladen wurde, fuehrt LINK das entsprechende Binden der Programme aus.

Normalerweise benoetigt jeder LINK-Lauf **letztlich zwei Programmnamen**. Ein Programmname bezeichnet das zu ladende und zu bindende Programm, waehrend der andere der Name des entstehenden ausfuehrbaren Programmes ist.

Wird waehrend des LINK-Laufes nur ein Programmname eingegeben, so wird entweder das COM-Programm nicht gerettet (in diesem Fall war der LINK-Lauf reine Zeitverschwendung) oder LINK antwortet mit der Fehlermeldung: ?NOTHING LOADED.

Beachte:
Wird nur ein Programmname eingegeben, dem jedoch ein /G-Schalter folgt, so wird zwar das COM-Programm nicht gerettet, aber das Programm wird nach dem Laden und Verbinden ausgefuehrt (siehe Erlaeuterungen in Abschnitt 6.2.2.).

Man kann so viele Programmnamen in einer Kommando-Zeile eingeben, wie es die Kapazitaet der Zeile erlaubt. Die Programmnamen koennen REL-Programme in verschiedenen Sprachen (hoehere Programmiersprachen, Assemblersprache) oder Laufzeit-Bibliotheken aus REL-Programmen fuer irgendeine der hoeheren Programmiersprachen bezeichnen.

Wenn ein LINK-Lauf beendet ist, wird sein Resultat unter dem angegebenen Namen gerettet (Programmname, gefolgt von einem /N-Schalter, siehe 6.2.2.). LINK gibt diesem Programmnamen den Dateityp .COM.

Ein Programmnamen-Kommando bei LINK besteht im allgemeinen aus der Geraetebezeichnung, dem Programmnamen und dem Dateityp.

```
(geraetebezeichnung):(programmname).(typ)
```

LINK nimmt standardmaessig als (geraetebezeichnung): das aktuelle logische Disketten-Laufwerk, als (typ) fuer die Eingabe REL und fuer die Ausgabe COM an.

6.2.2. Schalter

Die Schalter-Kommandos von LINK fuehren Funktionen ohne Laden und Verbinden aus. Schalter sind Buchstaben, denen ein Schraegstrich (/) vorangestellt ist. Man kann in einer Kommando-Zeile so viele Schalter wie benoetigt angeben, aber jedem einzelnen Schalter-Buchstaben muss ein Schraegstrich (/) vorangehen.

Wenn zum Beispiel ein Programm namens TEST geladen und gebunden, das entstandene Programm unter dem gleichen Namen auf Diskette gerettet und dann ausgefuehrt werden soll, benoetigt man zwei Programmnamen und zwei Schalter:

```
TEST,TEST/N/G
```

LINK speichert das geladene und gebundene Programm auf Diskette (/N-Schalter), dann wird es abgearbeitet (/G).
Einige Schalter koennen allein eingegeben werden (/E, /G, /R, /P, /D, /U, /M, /O, /H). Einige Schalter muessen dem entsprechenden Programmnamen folgen (/N, /S).

Der /Y-Schalter wirkt nur, wenn andere Schalter beim LINK-Lauf eingegeben wurden.

Einige Schalter muessen vor dem betreffenden Programmnamen eingegeben werden (/P, /D). Andere Schalter-Kommandos werden erst am Ende des LINK-Laufes ausgefuehrt (/N, /Y). Weitere Schalter-Kommandos werden unmittelbar nach ihrer Eingabe ausgefuehrt (/S, /R). Diese "Regeln" sollten bei Eingabe des LINK-Kommandos beachtet werden (siehe detaillierte Beschreibung der "Schalter").

Beachte:
Die LINK-Schalter wirken anders als die mit dem gleichen Buchstaben bezeichneten ASM-Schalter!

Uebersicht LINK-Schalter

Funktion	Schalter	Wirkung
Ausfuehren	/G	Ausfuehrung COM-Programm, dann zurueck zum Betriebssystem.
	/G:<name>	Beginnt das Abarbeiten des COM-Programmes bei der Adresse, die dem Wert von <name> entspricht.
Beenden	/E	Verlassen LINK und zurueck zum Betriebssystem.
	/E:<name>	Setzt die Startadresse des COM-Programmes gleich dem Wert von <name> und geht ins Betriebssystem.
Retten		Speichert alle geladenen und gebundenen Programme als Gesamtprogramm unter dem vor /N angegebenen Programmnamen ab.
Adresse setzen	/P	Setzt die Anfangs-Ladeadresse fuer Programme und Daten. Wenn auch der /D-Schalter benutzt wird, setzt /P nur die Programm-Ladeadresse.
	/D	Setzt die Anfangsadresse nur fuer Datenbereiche.
	/R	Setzt LINK auf den Anfangszustand.
Bibliotheks-suche	/S	Durchsucht die Bibliothek, deren Name unmittelbar vor /S angegeben ist.
Listen der Globalen	/U	Listet die undefinierten Globalen.
	/M	Listet die gesamte Cross-Referenz-Tabelle.

Funktion	Schalter	Wirkung
Festlegen Zahlendar-stellung	/O	Oktal.
	/H	Hexadezimal (Standard).
Spezieller Code	/Y	Erzeugt eine spezielle Datei fuer den symbolischen Test mit DU, fordert /N- und /E-Schalter, gibt der speziellen Datei die Typbezeichnung .SYM.

Letztlich werden **bei jedem** LINK-Lauf zwei Schalter benutzt. Diese Schalter gehoeren zu den drei ersten Funktionen :

Ausfuehren, Beenden und Retten.

Ausfuehren

/G-Schalter

Der /G-Schalter veranlasst LINK, die in der Kommando-Zeile eingegebenen Programme zu laden, zu binden und das geladene und gebundene Programm auszufuehren. Nach dem Programmlauf erfolgt die Rueckkehr ins Betriebssystem.

Beispiel:

```
LINK TEST,TEST/N/G
```

bindet TEST.REL, rettet das Ergebnis als Disketten-Datei TEST.COM und startet das Programm TEST.COM.

Das Programm wird ausgefuehrt, sobald die Kommando-Zeile interpretiert wurde. Kurz bevor die Ausfuehrung beginnt, gibt LINK drei Zahlen und den Text BEGIN EXECUTION aus.
Die erste Zahl ist die Startadresse des Programmes. Die zweite Zahl gibt die Adresse des naechsten verfuegbaren Bytes an, das ist die Endadresse des Programms + 1. Die dritte Zahl ist die Anzahl der 256-Byte-Seiten, die das Programm belegt.

Soll das COM-Programm nicht gerettt werden, gibt man nur einen Programmnamen und den /G-Schalter in der Kommando-Zeile an,

Beispiel:

```
LINK TEST/G
```

Beachte:
Wird kein COM-Programm erzeugt (kein /N eingegeben), muss fuer einen erneuten Start des Programmes LINK ebenfalls geladen werden.

/G:(name)

Der /G:(name)-Schalter besitzt die gleiche Wirkung wie der /G-Schalter, aber mit einer zusaetzlichen Funktion.

(name) ist ein in einem der geladenen und gebundenen Moduln definiertes Globalsymbol. LINK nimmt (name) als Startpunkt des Programms und traegt die entsprechende Adresse in den Sprungbefehl auf 0100-0102H ein.

Die Bedeutung dieses Schalters liegt darin, dass LINK das Programm an einem vorgegebenen Punkt startet, wenn der Startpunkt in den assemblierten Moduln nicht eindeutig definiert ist.

Normalerweise ist dies kein Problem, wenn ein Programm in einer hoeheren Programmiersprache (das LINK standardmaessig als Hauptprogramm ansieht) eingebunden wird, oder man bindet nur Assembler-Moduln und genau einer hat eine END (name)-Anweisung zur Definition des Startpunktes.

Aber in den Faellen, wo zwei oder mehr END (name)-Anweisungen vorhanden sind oder kein Modul eine solche enthaelt, gibt /G:(name) dem Programmverbinder LINK den logischen Programmanfang fuer das Ausfuehren des gebundenen Programmes.

Wird die Ausfuehrung eines Assembler-Moduls vor der des Programmes in einer hoeheren Programmiersprache gewuenscht, so ist dazu eine CALL- oder INCLUDE-Anweisung am Beginn des Programmes in der hoeheren Programmiersprache anzugeben.

Beenden

/E-Schalter

Der /E-Schalter bewirkt die Rueckkehr zum Betriebssystem, wenn man nicht die sofortige Programmausfuehrung wuenscht.

Ist das Binden beendet, gibt LINK drei Zahlen aus:
Startadresse, Adresse naechstes verfuegbares Byte, Anzahl der 256-Byte-Seiten.

/E:(name)

wirkt genau wie der /E-Schalter, definiert aber zusaetzlich durch (name) einen Startpunkt. Im uebrigen gilt das gleiche wie bei /G:(name) (siehe oben).

Retten

/N-Schalter

Der /N-Schalter bewirkt das Retten des geladenen und gebundenen Programmes auf Diskette. Dabei ist es wichtig, dass fuer den /N-Schalter ein Programmname spezifiziert wird. Ist kein

Typ angegeben, wird standardmaessig .COM angenommen. Der Programmname muss unmittelbar vor /N angegeben werden.

Der /N-Schalter wirkt erst bei einem nachfolgenden (braucht nicht unmittelbar nachfolgend zu sein) /E- oder /G-Schalter. Der haeufigste Fehler bei Verwendung des /N-Schalters ist der, dass vergessen wurde zwei Programmnamen zu spezifizieren: einen als den des zu bindenden Programmes und einen als Name des zu rettenden Programmes.

Deswegen muss die Kommando-Zeile mindestens:

```
LINK TEST,TEST/N/G
```

enthalten.

Der erste Name TEST bezeichnet das zu ladende und zu bindende Programm, der zweite Name ist der Name des als Ergebnis des LINK-Laufes entstehenden und zu rettenden Programmes. Natuerlich ist es moeglich, Programmnamen in beliebiger Reihenfolge anzugeben, z.B.:

```
LINK TEST/N,MOD1,MOD2,PROG/G
```

Hier laedt und bindet LINK die Programme MOD1,MOD2 und PROG, dann wird das entstandene Programm unter dem Namen TEST gerettet.

Aus diesen zwei Beispielen ist zu sehen, dass das Programm mit dem Namen, dem der Schalter /N folgt, nicht geladen wird. /N ist nur eine Spezifikation fuer ein **Ausgabe**-Programm, man muss letztlich auch ein **Eingabe**-Programm angeben.

Man sollte diesen Schalter bei jedem LINK-Lauf benutzen, weil es keine andere Moeglichkeit der Rettung des gebundenen Programmes gibt und Nichtretten bedeutet, dass vor der naechsten Ausfuehrung des Programmes der Lade-Binde-Vorgang wiederholt werden muesste. Einmal auf Diskette gerettet, braucht man nur den COM-Programmnamen als Systemkommando zur Abarbeitung des Programmes anzugeben.

Beachte:
Wird als Name des zu rettenden Programmes ein Name angegeben, der ein schon auf der gleichen Diskette gespeichertes anderes Programm bezeichnet, so wird das vorhandene Programm (Datei) ueberschrieben und ist somit verloren.

Zwei der bisher erlaeuterten Schalter (/N plus entweder /G oder /E) sind die einzigen Schalter, die fuer die meisten LINK-Laeufe noetig sind. Die Benutzung von anderen Schaltern gibt einige zusaetzliche Funktionen, die eine detailliertere Beeinflussung der LINK-Ablaeufe ermoeglichen.

Adressen setzen

Jedes in Assemblersprache geschriebene Programm kann sich aus absoluten (ASEG) und relativen, d.h. verschieblichen (COMMON, DSEG, CSEG) Programmteilen zusammensetzen (s. Beschreibung in der Anleitung fuer den Programmierer Teil II).
Laedt LINK ein solches Programm in den Speicher, so geschieht das standardmaessig wie folgt:

Unabhaengig von der phyischen Anordnung der einzelnen Programmteile (ASEG, COMMON, DSEG, CSEG) im Assemblerprogramm erfolgt die Speicherung in der Rangordnung

1. COMMON - Programmteile
2. DSEG - Programmteile
3. CSEG - Programmteile

beginnend bei Adresse 0103H (0100H-0102H enthaelt unbedingten Sprung an den logischen Programmanfang bzw. den Wert 00H).

Beispiel:

Besitzt ein ASM-Quellprogramm die Struktur

(dseg1,cseg1,common1,cseg2,dseg2,common2),

so erzeugt LINK ab 103H ein wie folgt strukturiertes COM-Programm:

(common1,common2,dseg1,dseg2,cseg1,cseg2).

ASEG - Programmteile werden in den entsprechenden vorgegebenen Speicherbereich geladen, dabei ist darauf zu achten, dass sich keine Ueberlagerungen mit anderen Programmteilen ergeben.

Weitere durch LINK geladene Programme werden lueckenlos an das vorher geladene Programm angefuegt, wobei jedes Programm in sich in obiger Rangordnung gespeichert wird.

Beispiel:

Aus Programmen P1 und P2 mit der Struktur

P1: (dseg1,cseg1,dseg2)
P2: (cseg2,common1,dseg3,cseg3)

erzeugt LINK durch das Kommando

*P1,P2

folgende Struktur:

(dseg1,dseg2,cseg1,common1,dseg3,cseg2,cseg3).

COMMON-Bloecke gleichen Namens werden stets in den gleichen

Speicherbereich geladen, den der erste COMMON-Block dieses Namens durch LINK zugewiesen erhaelt. Er legt somit die Laenge des Bereiches fest, und das entsprechende Programm muss zuerst geladen werden.

Die Verwendung der /P- und /D- Schalter ermoeglicht einen von dieser Standardstrukturierung abweichenden Aufbau des von LINK erzeugten COM-Programmes.

Die /P- und /D-Schalter legen absolute Adressen innerhalb des COM-Programmes fest. Dies kann z.B. zum Trennen von Programm- und Datenbereich innerhalb des erzeugten Gesamtprogrammes oder zum Festlegen von fuer den Test guenstigen Anfangsadressen der gebundenen Einzelprogramme genutzt werden.

Grundsaetzlich gilt bei Verwendung von /P- und /D-Schalter:

- Die Schalter wirken nur auf nachfolgende Programme, d.h. bereits geladene Programme werden nicht beeinflusst.
- Auf ASEG-Programmteile ueben diese Schalter keine Wirkung aus.
- Waehrend eines LINK-Laufes koennen mehrere /P- und /D-Schalter angegeben werden.
- Luecken zwischen gebundenen Programmen werden durch LINK nicht initialisiert, sondern koennen Daten oder Programmteile eines vorhergehenden Programmes enthalten. Dies kann unter Umstaenden die Ursache fehlerhafter Ergebnisse bei Abarbeitung des erzeugten Programmes sein.
- Das Format der Schalter ist

  /P:(adresse),

  bzw.

  /D:(adresse),

  Die Adresse muss in der aktuellen Zahlendarstellung angegeben werden. Die Standarddarstellung ist hexadezimal.

/P-Schalter

Werden waehrend des LINK-Laufes nur /P-, aber keine /D-Schalter angegeben, so legt ein /P-Schalter die Ladeadresse des nachfolgenden zu bindenden Programmes fest. Innerhalb des Programmes erfolgt die oben angefuehrte Umstrukturierung.

Beispiel (s.o.):

```
*/P:200,P1,/P:300,P2

erzeugt:

(dseg1,dseg2,cseg1)                  ab 0200H
und  (common1,dseg3,cseg2,cseg3)     ab 0300H
```

Damit koennen Programme nicht lueckenlos aufeinanderfolgend ab vorgegebenen Adressen im Gesamtprogramm angeordnet werden.

Werden beide, /P- und /D-Schalter verwendet, so beeinflusst der /P-Schalter nur die CSEG-Teile der nachfolgenden Programme.

/D-Schalter

Der /D-Schalter wirkt nur auf die COMMON- und DSEG-Teile der nachfolgenden zu bindenden Programme.

Beispiel (s.o.):

```
Die LINK-Kommando-Zeile

*/D:200,P1,P2

erzeugt:

(cseg1,cseg2,cseg3)                  ab 0103H
                                     (Standard-Anfangsadresse)

und  (dseg1,dseg2,common1,dseg3)     ab 0200H
```

Damit kann eine Trennung von Programm- und Datenteil erfolgen. Es ist auch moeglich, Programm- und Datenteil ab vorgegebenen Adressen anzuordnen, z.B. Programmteil ab 0400H und Datenteil ab 0200H:

```
*/D:200,/P:400,P1,P2
```

LINK bindet die geladenen Programme entsprechend der angegebenen Anfangsadressen zu einem Gesamtprogramm. Nicht erlaubt ist das Mischen der durch Schalter getrennten Bereiche, d.h. beispielsweise die Kommando-Folge:

```
*/P:200,/D:300,P1,P2
*/P:400,P3
```


wuerde ein CSEG ab 0200H, DSEG ab 0300H und CSEG ab 0400H erzeugen. Dies ist nicht moeglich und fuehrt zu einer Fehlermeldung.

Zusaetzliche Bemerkung zu /P- und /D-Schalter:

Wenn das Programm zu gross fuer LINK ist, ist es in einigen Faellen durch die gemeinsame Benutzung von /D- und /P-Schalter doch moeglich, die zu ladenden und zu verbindenden Moduln in den Speicher zu laden. Waehrend LINK laedt und bindet, bildet es eine Lade-Tabelle, die aus fuenf Byte fuer jede Eintragung besteht. Durch das Setzen beider Schalter, /D und /P, entfaellt das Bilden dieser Tabelle und somit steht ein zusaetzlicher Arbeitsspeicherbereich zur Verfuegung.

/R-Schalter

Der /R-Schalter setzt LINK auf den Anfangszustand zurueck. LINK prueft vor Ausfuehrung der Kommandos die eingegebene Kommandozeile. Wird ein /R-Schalter erkannt, werden alle geladenen Programme ignoriert. LINK initialisiert sich selbst und meldet sich als bereit zur naechsten Kommandoeingabe.

Bibliothekssuche

/S-Schalter

Der /S-Schalter durchsucht die Datei (REL-Programm(e)), deren Name unmittelbar vor dem Schalter angegeben ist, nach Routinen, Subroutinen, Definitionen fuer Globale usw.

Bei der Eingabe der Kommando-Zeile muss der Dateiname mit dem angehaengten /S-Schalter vom Rest der Zeile durch ein Komma getrennt werden, z.B.:

```
LINK TEST/N,ULIB/S,TEST/G
```

Der /S-Schalter wird benutzt, um Programme zu suchen, die durch das Programm LIB zu einer Bibliothek zusammengefasst wurden.

Listen der Globalen

/U-Schalter

Der /U-Schalter bewirkt das Listen aller undefinierten Globalen. Er wirkt nur in einer Kommando-Zeile, die weder einen /G-, noch einen /E-Schalter enthaelt.

Beachte, dass LINK automatisch alle undefinierten Globalen listet, ausser wenn die Kommando-Zeile auch einen /S-Schalter enthaelt. In diesem Fall gibt man den /U-Schalter ein und erhaelt so diese Liste.

Das Listen kann durch die Betaetigung der Taste ^S unterbrochen und mit ^Q fortgesetzt werden.
Zusaetzlich zum Listen der undefinierten Globalen werden durch den /U-Schalter der Anfangsladepunkt, das Ende und die Groesse des gemeinsamen Programm- und Datenbereiches ausgegeben. Sind beide Schalter, /P und /D, gesetzt, werden Anfangspunkt, Ende und Groesse beider Bereiche gesondert gelistet.

/M-Schalter

Durch den /M-Schalter erfolgt das Listen aller Globalen, der definierten und der undefinierten. Dabei steht neben den definierten Globalen ihr Wert (Adresse), die undefinierten sind durch einen Stern (*) gekennzeichnet.

Auch hier werden wie beim /U-Schalter Anfang, Ende und Groesse von Programm- und Datenbereich zusaetzlich gelistet.

Festlegen der Zahlendarstellung

/O-Schalter

Der /O-Schalter bewirkt das Umstellen der aktuellen Zahlendarstellung auf Oktal.

/H-Schalter

Der /H-Schalter setzt die aktuelle Zahlendarstellung auf Hexadezimal zurueck.
Hexadezimal ist die Standard-Darstellung bei LINK, deshalb braucht man den /H-Schalter nur anzugeben, wenn man vorher /O eingegeben hatte und zur hexadezimalen Darstellung zurueck will.

Spezieller Code

/Y-Schalter

Der /Y-Schalter erzeugt eine Spezial-Ausgabe-Datei, die von dem Testhilfsprogramm DU fuer den symbolischen Test benutzt wird.

Der /Y-Schalter verlangt den /N- und den /E- (nicht /G!) Schalter in der Kommandozeile, z.B.

```
LINK TEST,TEST/Y/N/E
```


Die durch den /Y-Schalter gerettete Datei hat den Dateityp .SYM. Ein COM-Programm wird ebenfalls gerettet. So werden in obigem Beispiel beide, COM- und SYM-Datei, erzeugt.
Die SYM-Datei enthaelt die Namen und Adressen von allen Globalen.

6.3. Fehlermeldungen

Bei waehrend des LINK-Laufes auftretenden Fehlern erfolgt eine entsprechende Ausschrift, der meist das Zeichen ? oder % vorangestellt ist.

?No Start Address

Es wurde der /G-Schalter benutzt, aber kein Hauptprogramm geladen.

?Loading Error

Keine von LINK verarbeitbare, korrekt formatierte REL-Datei (Programm, Modul) geladen.

?Out of Memory

Verfuegbarer Arbeitsspeicherplatz reicht zum Laden der Programme nicht aus.

?Command Error

Nicht vorhandenes LINK-Kommando.

?<dateiname> Not Found

Datei (Programm) mit dem im Kommando angegebenen Namen existiert nicht.

?Start Symbol - <name> - Undefined

Der im /E:<name> oder /G:<name> angegebene Startpunkt existiert nicht als definierte Globale.

?Nothing Loaded

Ein <name>/S oder /E oder /G wurde angegeben, aber kein Objektprogramm (REL-Datei) geladen. Das heisst, es wird versucht, eine Bibliothek zu durchsuchen, LINK zu verlassen oder ein Programm auszufuehren, obwohl nichts vorher geladen wurde, z.B.

```
TEST/N/E
```

ergibt ein "?Nothing Loaded", weil TEST/N ein Programm TEST.COM benennt, aber das Laden eines REL-Programmes

fehlt.
Um ein Programm zu laden, muss nur sein Name eingegeben werden. Um ein Programm abzuspeichern, muessen sein Name gefolgt von dem /N-Schalter und entweder einem /E- oder einem /G-Schalter eingegeben werden.

So ist jede der Kommandofolgen zulaessig:

```
LINK TEST,TEST/N/E
```

oder

```
LINK
*TEST
*TEST/N/E
```

oder

```
LINK TEST/N,TEST/E
```

?Can't Save Object File

Diskettenfehler waehrend des Schreibens des zu rettenden Programmes. Normalerweise bedeutet dies, dass die Diskette gefuellt oder schreibgeschuetzt ist.

X2nd COMMON larger /xxxxxx/

Wenn Module geladen werden, die COMMON-Bloecke (siehe Assemblerhandbuch) enthalten, reserviert LINK einen Speicherbereich entsprechend der Laenge des ersten geladenen COMMON-Blockes.

Enthaelt nun ein nachfolgender Modul einen COMMON-Block mit einer groesseren Laenge, so antwortet LINK mit dieser Fehlerausschrift. Das bedeutet, dass der erste geladene Block mit der Bezeichnung |xxxxxx| nicht der groesste Block gleichen Namens war. Man muss entweder die Ladereihenfolge vertauschen oder die COMMON-Bloecke umbenennnen.

XMult. Def. Global YYYYYY

Der Global- (PUBLIC) Symbol-Name YYYYYY ist in den geladenen Moduln mehrfach definiert.

&Overlaying Program Area ,Start = xxxx
,Public =<symbolname> (xxxx)
,External =<symbolname> (xxxx)

Normalerweise erfolgt diese Ausschrift, wenn entweder /D oder /P eine Adresse innerhalb des durch LINK belegten Bereiches definieren. Man sollte dann die Schalter-Adressen ruecksetzen.

Diese Ausschrift erfolgt auch, wenn nach bereits geladenen Programmen ein Schalter mit einer zu niedrigen Adresse fuer ein noch zu ladendes Programm angegeben wird, so dass eine Ueberlagerung und somit ein Zerstoeren von Teilen des bereits geladenen Programmes erfolgen wuerde.

Sei zum Beispiel die Laenge von SUM groesser als 147 Bytes und man gibt die Kommandos

```
SUM,/P:150,SUBR1,TEST/N/E
```

ein, so wird man diese Fehlerausschrift erhalten.

XOverlaying Data Area ,Start = xxxx
,Public = <symbolname> (xxxx)
,External = <symbolname> (xxxx)

Die /D- und /P-Schalter waren so gesetzt, dass eine Ueberlagerung der Bereiche erfolgen wuerde. Wenn z.B. bei /D eine hoehere Adresse als bei /P angegeben wird, die aber noch im Programmbereich liegt oder wenn die /D-Adresse kleiner als die /P-Adresse ist, aber der Startpunkt des Programmes noch im Datenbereich liegt, dann erfolgt diese Fehlerausschrift.

?Intersecting Program Area

oder

Intersecting Data Area

Programm- und Datenbereiche ueberschneiden sich und einer im Ueberschneidungebereich liegenden Adresse kann kein aktueller Wert zugewiesen werden.

Origin Above Loader Memory, Move Anyway (Y or N)?

oder

Origin Below Loader Memory, Move Anyway (Y or N)?

Diese Ausschrift erfolgt nur bei einem /E- oder /G-Schalter. Steht LINK nicht genuegend Speicherplatz zum Laden eines Moduls zur Verfuegung, aber ein /E oder /G wurde nicht eingegeben, wird man die "?Out of Memory"-Ausschrift erhalten.

Erfolgt ein Speicherueberlauf, weil die zu ladenden Module zu gross fuer den verfuegbaren Speicherplatz sind oder ein /D- oder /P-Schalter zu hoch fuer das Programm gesetzt wurde, gibt LINK die "Origin Above Loader Memory"-Ausschrift.

Wird ein /D- oder /P- Schalter unterhalb von 100H gesetzt, erfolgt die "Origin Below Loader Memory"-Ausschrift. Damit wird verhindert, dass das Program in den

fuer das Betriebssystem bestimmten Speicherbereich geladen wird.
Wird Y (ENTER) eingegeben, erfolgt ein Transport des Bereiches und Weiterarbeit. Wird irgend etwas anderes eingegeben, so wird LINK beendet. In jedem Fall erfolgt, wurde der /N-Schalter eingegeben, eine Rettung des Programmes auf Diskette.

7.____LIB 1520 (SCPX)

7.1.___Aufruf des Bibliothekars

LIB ist ein Dienstprogramm zum Bilden und Verwalten einer Bibliothek aus in Assemblersprache geschriebenen Programmen. LIB kann auch als Laufzeit-Bibliotheksverwalter fuer hoehere Programmiersprachen eingesetzt werden.

Anmerkung:
Vor der Benutzung von LIB ist die Anfertigung einer Sicherheitskopie der zu behandelnden Bibliothek zu empfehlen, da man auch leicht Bibliotheken zerstoeren kann.

LIB 1520 (SCPX) wird durch das Kommando

```
LIB
```

aufgerufen. LIB meldet sich mit einem Stern (*), um seine Bereitschaft zur Kommandoeingabe anzuzeigen. Jedes Kommando in LIB fuegt Moduln zu der sich im Aufbau befindlichen Bibliothek hinzu.

LIB kann Laufzeit-Bibliotheken aus Programmen in Assemblersprache bilden. Diese Programme koennen Unterprogramme fuer Compiler hoeherer Programmiersprachen (z.B. FORTRAN) oder fuer andere Assembler-Programme sein.

Die durch LIB zu einer Bibliothek zusammengefassten Programme koennen einzelne Programme oder Programme, im folgenden als "Moduln" bezeichnet, einer bereits existierenden Bibliothek sein.

Der Vorteil der Bildung einer Bibliothek liegt darin, dass alle fuer die Ausfuehrung eines Programmes benoetigten Routinen mit ihm zu einem ausfuehrbaren Gesamtprogramm (COM-Datei) vom Programmverbinder LINK gebunden werden koennen.
Dazu sind nur der Bibliotheksname, gefolgt von einem /S in der LINK-Kommando-Zeile einzugeben.

Beispiel:

```
LINK HPRG,BIB1/S,PROG/N/G
```

Dies ist bequemer als die einzelne Eingabe der Routinen, besonders wenn ihre Anzahl gross ist. Bei Verwendung einer Bibliotheksdatei kann man sicher sein, dass alle benoetigten Moduln in die COM-Datei gebunden werden, ausserdem besteht keine Gefahr, dass die Kommandozeile nicht ausreicht. Schliesslich steht die zu einer Bibliothek zusammengefasste Sammlung von Unterprogrammen fuer das Einbinden in beliebige Programme zur Verfuegung.

LIB belegt einen Speicherbereich von etwa 5 K Byte.

7.2.___Anwendungsfaelle

Die zwei meist benoetigten Anwendungsfaelle von LIB sind **Bilden** einer Bibliothek und **Listen** einer Bibliothek. Die folgenden Beispiele zeigen die Grundkommandos fuer diese beiden Aufgaben.

7.2.1._Bilden einer Bibliothek

Beispiel:

```
A>LIB
*MATLIB=SIN,COS,TAN,COTAN
*EXP
*/E
A>
```

In diesem Beispiel wird durch das SCP-Kommando LIB der Bibliothekar LIB 1520 (SCPX) aufgerufen, der sich mit einem Stern (*) als bereit zur Kommandoeingabe meldet.

MATLIB ist der Name der Bibliothek, die gebildet werden soll. Auf Diskette wird eine leere Datei MATLIB.LIB erzeugt. SIN, COS, TAN, COTAN sind die Namen der Programme, die zu MATLIB verkettet werden sollen.

EXP ist der Name eines anderen Programmes, das zu MATLIB gehoeren soll. (EXP koennte auch in der vorhergehenden Kommandozeile angegeben werden; dieses Beispiel zeigt, dass die Programmnamen einzeln oder auch mehrere zusammengefasst eingegeben werden koennen.)

/E bewirkt das Umbenennen von MATLIB.LIB in MATLIB.REL und die Ausgabe auf Diskette, wobei die vorher gebildete leere Datei MATLIB.LIB ueberschrieben wird und das Verlassen von LIB.

7.2.2. Listen einer Bibliothek

Beispiel:

```
A>LIB
*MATLIB.LIB/U
*MATLIB.LIB/L
.
.
.
(Liste der Symbole in MATLIB.LIB)
.
.
.
*^C  (CTRL C)
A>
```

In diesem Beispiel erfolgt ebenfalls der Aufruf des Bibliothekars durch das Kommando LIB.

MATLIB.LIB/U bewirkt, dass MATLIB.LIB nach solchen Cross-Referenzen durchsucht wird, die beim ersten Durchlauf durch die Bibliothek nicht aufgeloest werden koennen ("rueckwaerts" gerichtete Referenzen).

MATLIB.LIB/L listet die Namen und Symboldefinitionen der in MATLIB.LIB enthaltenen Moduln.
(^C) kehrt zu SCP zurueck, ohne eine Bibliothek zu beeinflussen.

Anmerkung:
Wird keine neue Bibliothek gebildet, sollte LIB stets mit ^C verlassen werden, denn durch /E wird in diesem Fall eine leere Bibliothek FORLIB.REL gebildet, die eine bereits vorhandene FORTRAN-Laufzeit-Bibliothek FORLIB.REL zerstoeren wuerde.

7.3. LIB-Kommandos

Die Kommandos in LIB bestehen aus einem wahlfreien Zielfeld, einem Quellfeld und einem wahlfreien Schalter-Feld.
Das Format eines LIB-Kommandos ist

```
ziel=quelle/schalter
```

7.3.1. Zielfeld

Dieses Feld wird wahlfrei angegeben. Es erfolgt die Angabe des Namens (auch mit Dateityp moeglich) der zu bildenden Bibliothek:

```
dateiname=
```


Wird das Zielfeld weggelassen, erzeugt LIB standardmaessig den Dateinamen FORLIB.
Der Standard-Typ ist .LIB.

Anmerkung:
Der Standard-Dateiname FORLIB.LIB darf nicht mit FORLIB.REL, der FORTRAN-Laufzeit-Bibliothek verwechselt werden.

Diese beiden Bibliotheken werden nur identisch sein, wenn man durch ein LIB-Kommando alle Moduln der FORTRAN-Laufzeit-Bibliothek in eine neue Bibliothek kopiert.

Ueberdies erhaelt, wenn man LIB verlaesst, der Standard-Bibliotheksname den Typ .REL, das bedeutet, dass man die FORTRAN-Bibliothek FORLIB.REL ueberschreibt.

Aus diesem Grunde wird empfohlen, wenn man die Zerstoerung der FORTRAN-Laufzeit-Bibliothek nicht ausdruecklich beabsichtigt, bei Bildung einer Bibliothek stets einen bestimmten Bibliotheksnamen anzugeben.

7.3.2. Quellfeld

Fuer dieses Feld sind einige Eintragungen notwendig. Alle Quellfelder muessen REL-Dateien (Programme) sein.
Die Quellfeld-Eintragungen bezeichnen die Programme oder Moduln, die man zu einer bestimmten Bibliotheksdatei zusammenfuegen will. Es gibt zwei Eintragungsmoeglichkeiten:

- nur Dateiname(n)
- eine Kombination aus Dateiname(n) und Modul-Name(n)

Folgende Syntaxregeln sind zu beachten:

- Besteht ein Kommando nur aus Dateiname(n), so werden die Eintragungen durch Komma getrennt, z.B.:

  ```
  PROG1,PROG2,PROG3
  ```

- Besteht ein Kommando aus Datei - und Modulnamen, muessen die Modulnamen in spitze Klammern (<>) eingeschlossen sein. Die Module folgen unmittelbar dem Namen der Datei (Bibliothek), in der sie enthalten sind. Jede Kombination dateiname <modul> wird durch ein Komma von anderen Eintragungen in der Kommando-Zeile getrennt, z.B.:

  ```
  PROG1,PROG2<UP1>,PROG3<SUM>,PROG4
  ```

- Werden mehr als ein Modul der gleichen Datei angegeben, muessen die in spitze Klammern (<>) eingeschlossenen Modul-Namen voneinander durch ein Komma getrennt werden, z.B.:

  ```
  PROG1,PROG2<UP1,UP2>,PROG3
  ```

(Siehe "Ergaenzungen zu den Quell-Moduln".)

Dateien und Moduln sind typische Unter- oder Hauptprogramme in hoeheren Programmiersprachen (z.B. FORTRAN, BASIC) oder Programme in der Assemblersprache, die ENTRY- bzw. GLOBAL- oder PUBLIC-Anweisungen enthalten (Eintrittspunkte).

LIB erkennt einen Modul durch seinen Programmnamen, der ein Dateiname sein kann oder ein Name, der entweder durch die .TITLE- oder NAME-Pseudooperation der Assemblersprache definiert wurde (siehe Anleitung fuer den Programmierer Teil II).

Alle Quelldateien muessen REL-Dateien sein, LIB kettet eine Datei oder einen Modul an den anderen. So gibt es keinen Unterschied zwischen dem Beispiel unter Syntaxregel 2 und der Folge:

```
PROG1
PROG2(UP1)
PROG3(SUM)
PROG4
```

Es ist zu empfehlen, die Moduln so zu verketten, dass alle Cross-Referenzen "vorwaerts" gerichtet sind. Das heisst, die Moduln, die externe Bezuege enthalten, sollten physisch vor den Moduln angeordnet werden, die die Eintrittspunkte (die Definitionen) enthalten. Im anderen Fall kann LINK, wenn eine Bibliothek durchsucht werden soll, nicht in einem einzigen Durchlauf alle Cross-Referenzen aufloesen.

Ergaenzungen zu den Quell-Moduln

LIB kann Moduln aus vorhandenen Bibliotheken oder anderen REL - Dateien herausloesen.

Es gibt fuer die Definition einer solchen Modulkette folgende Moeglichkeiten:

- **Nur ein einziger Modul:**
  Angabe des Modulnamens, z.B. enthaelt

  ```
  PROG1(UP1)
  ```

  Nur den Modul UP1 von PROG1.

- **Einige nicht unmittelbar aufeinanderfolgende Moduln einer Datei:**
  Angabe der Modulnamen, getrennt durch Komma, z.B.

  ```
  PROG1(UP1,UP2,UP3)
  ```

  umfasst die Moduln UP1, UP2 und UP3.

  Allgemein koennen die Moduln in beliebiger Reihenfolge in der Ursprungdatei angegeben werden, aber guenstigerweise in einer solchen, wie sie spaeter fuer eine geeignete Ein-

Durchlauf-Suche benoetigt werden.

- **Vom ersten Modul der Datei bis zum angegebenen Modul:**

  Angabe zweier Punkte (..) und den Namen des letzten Moduls der Kette. So schliesst z.B.

  ```
  PROG1(..UP1)
  ```

  Alle Moduln ein, angefangen vom ersten Modul in PROG1 bis zum Modul UP1.

- **Von einem angegebenen Modul bis zum letzten Modul der Datei:**
  Angabe des Namens des Anfangsmoduls der Kette, gefolgt von zwei Punkten (..). So definiert

  ```
  PROG1(UP1..)
  ```

  eine Kette aller Moduln von UP1 bis zum letzten Modul von PROG1.

- **Aufeinanderfolgende Moduln:**
  Angabe des Namens des ersten Moduls der Kette, gefolgt von zwei Punkten, gefolgt vom Namen des letzten Moduls der Kette, so enthaelt

  ```
  PROG1(UP1..UP4)
  ```

  alle Moduln von UP1 bis UP4.

- **Relative Kette:**
  indirekte Bezeichnung eines Moduls durch Angabe eines Modul-Namens und zum Abstand dem zu benennenden Modul.
  Der Abstand ist eine ganze Zahl im Bereich 1-255.

  Beispiele:

  Sei die Bibliothek PROG1 in der angegebenen Reihenfolge aus den Moduln UP1, UP2, UP3, UP4, UP5, SUM zusammengesetzt, dann bezeichnen

  ```
  PROG1(UP2+2) den Modul UP4
  PROG1(UP5-3) den Modul UP2.
  ```

  Modulreihen und relative Angaben koennen auch gemischt erfolgen, z.B. definiert

  ```
  PROG1(UP1+1..SUM-1)
  ```

  eine Kette aller Moduln zwischen UP1 und SUM, ausser UP1 und SUM selbst.

- alle Moduln einer Datei:
  Angabe nur eines Dateinamens, z.B. schliesst

```
PROG1
```

  alle Moduln der Datei PROG1 ein.

7.3.3. Schalterfeld

Eine Eintragung im Schalterfeld bewirkt zusaetzliche LIB-Funktionen. Eine Schalterfeld-Eintragung besteht aus einem Buchstaben, dem ein Schraegstrich (/) vorangeht.

Schalter	Wirkung
/E	Verlaesst LIB und bewirkt die Rueckkehr zum Betriebssystem. Wird keine neue Bibliothek gebildet oder eine vorhandene geaendert, sollte ^C (CTRL C) anstelle /E zur Beendigung benutzt werden (s. Anmerkung unter 7.2.2.). Erfolgt keine Angabe des Dateitypes im Zielfeld bei der Kommandoeingabe, erzeugt LIB als Standard den Dateityp .LIB. Durch /E wird dieser standardmaessig erzeugte Dateityp in .REL umbenannt und die neu gebildete Bibliothek auf Diskette ausgegeben. Ein bei der Kommandoeingabe direkt angegebener Dateityp im Zielfeld wird dagegen nicht veraendert, auch wenn dieser .LIB ist! Es wird nochmals nachdruecklich empfohlen, stets einen Dateinamen im Zielfeld der LIB-Kommandozeile anzugeben!
/R	/R hat die gleiche Wirkung wie /E, aber bewirkt keine Rueckkehr zum Betriebssytem. Auch hier ist die gleiche Vorsicht wie beim Gebrauch von /E angebracht! /R wird anstelle /E nur dann benutzt, wenn unmittelbar im Anschluss die naechste Bibliothek gebildet werden soll.
/L	Erzeugt eine Liste der Moduln einer angegebenen Datei und der in ihnen enthaltenen Symboldefinitionen (Cross-Referenzen).

Schalter	Wirkung
/U	Es werden die Symbole gelistet, die bei einem einfachen Durchlauf durch die Bibliothek undefiniert bleiben wuerden, d.h., alle "rueckwaerts" (zu einem vorhergehenden Modul) weisenden Referenzen.
/C	/C setzt LIB zurueck, d.h., die eingegebenen Kommandos werden ignoriert, ohne dass LIB verlassen wird. Die sich im Aufbau befindliche Bibliothek wird geloescht und der LIB - Lauf beginnt erneut. LIB meldet sich mit einem Stern (*). /C wird benutzt, wenn falsche Moduln spezifiziert oder eine falsche Reihenfolge eingegeben wurden und ein Neueingabe erfolgen soll.
/O	setzt Anzeigemodus (Zahlendarstellung) auf Oktalbasis, /O wird gemeinsam mit /L angegeben. Beachte: Werden mehrere Schalter angegeben, muss jedem Schalter ein Schraegstrich vorangehen, z.B. MATLIB/L/O
/H	setzt Anzeigemodus auf Hexadezimal-Basis zurueck (Standard-Zahlendarstellung).

8. DU 1520 (SCPX)

8.1. Einfuehrung

DU (Debugger) unterstuetzt das dynamische Testen von Programmen, die in Z80-Mnemonik geschrieben sind. Bei Vorhandensein einer Symboltabelle (SYM-Datei) fuer das zu testende Programm ist ein symbolisches Testen moeglich.

DU wird durch Eintasten eines der nachfolgenden SCP-Kommandos:

(1) DU
(2) DU p.COM
(3) DU p.COM p.SYM

geladen, wobei p den Namen des mit DU zu ladenden und zu testenden Programmes bezeichnet, bei (3) wird zusaetzlich die Symboltabelle geladen. Anstatt einer COM-Datei kann auch eine beliebige andere Datei geladen werden.

Nach dem Start meldet sich DU mit:

```
DU 1520 (SCPX) V x.y
U880-Debugger
SYMBOLS
NEXT  PC  END
nnnn pppp eeee
```

,wobei x.y die Versionsnummer, nnnn die Adresse des naechsten freien Speicherplatzes nach dem zu testenden Programm, pppp den durch das zu testende Programm initialisierten Befehlszaehlerstand und eeee die Adresse des letzten freien Speicherplatzes hexadezimal angibt.

Falls keine Symboltabelle geladen wird, entfaellt "SYMBOLS". Wird DU ohne das zu testende Programm geladen, so werden nur die ersten beiden Zeilen ausgeschrieben.

Anschliessend wird durch "*" angezeigt, dass DU eine Kommandoeingabe erwartet.

Jedes DU-Kommando besteht aus dem Kommando-Typ (ein Buchstabe) und optionalen Spezifikationen, die durch Leerzeichen oder Komma getrennt sind.

Alle Kommando-Zeilen koennen 64 Zeichen lang sein und werden durch (ENTER) beendet. Danach wird das Kommando ausgefuehrt.

DU wird mit Warmstart (^C) beendet.

Folgende Kommando-Typen enthaelt DU:

A	(Assemble)	Assemblieren
C	(Call)	Aufruf Unterprogramm
D	(Display)	Anzeige Speicherinhalt
F	(Fill Memory)	Fuellen Speicher
G	(Go)	Echtzeitabarbeitung
H	(Hex)	Hexa-Rechnung
I	(Input Line)	Vorbereitung Datei-Eingabe
L	(List Mnemonics)	Reassemblieren
M	(Move)	Transport Speicherbereich
P	(Pass Point)	Durchlauf-Punkt
R	(Read)	Dateieingabe
S	(Set Memory)	Speichermanipulation
T	(Trace)	Protokollierung
U	(Untrace)	Kein Protokollieren
X	(Examine)	CPU-Registermanipulation

8.2. DU-Kommandos

Die in den Kommandospezifikationen enthaltenen Zahlen koennen in verschiedener Form dargestellt werden, wobei der Zahlenbereich 0 - 65535 betraegt. Zahlen, die diesen Bereich uebersteigen, werden modulo 65536 interpretiert.

- Hexadezimale Darstellung (Standarddarstellung)
  Verwendet werden die Ziffern 0-9 und die Buchstaben A-F. Gross- und Kleinbuchstaben sind gleichberechtigt, z.B. 3f, FF3e.

- Dezimale Darstellung
  Dezimalzahlen werden durch ein vorangestelltes Zeichen "#" gekennzeichnet und bestehen aus den Ziffern 0-9. Beispiele fuer Dezimalzahlen sind: #48, #9873,#0.

- String-Darstellung
  Verwendet werden ASCII-Zeichen, die in Apostrophe eingeschlossen werden. Der Code des am weitesten rechts stehenden Zeichens wird dabei als niederer Wert angesehen.

  Beispiele:

  'A' ist aequivalent mit dem Hexawert 41 bzw. #65,
  'xB' entspricht 2A42.

  Ein Apostroph wird im String als Apostrophenpaar dargestellt: '#''' entspricht dem Hexawert 2327.

- Symbol-Darstellung
  Ist eine Symboltabelle geladen, koennen auch Symbol-Bezuege verwendet werden.

Beispiele:

```
.s    Adresse des Symbols s
@s    2-Byte-Wert von .s
=s    1-Byte-Wert von .s
```

Es sei zum Beispiel der Adresse Ø1Ø4 das Symbol T1 in der Symboltabelle zugeordnet und der Inhalt des entsprechenden Speicherplatzes (Ø1Ø4)=14 und (Ø1Ø5)=5A, so entsprechen in Hexa-Darstellung

```
.T1 = Ø1Ø4
@T1 = 5A14
=T1 = 14
```

- Ausdruecke

Ausdruecke bestehen aus einer von links nach rechts verlaufenden Folge von Hexazahlen, Dezimalzahlen, Strings, und Symbol-Bezugnahmen, verbunden durch die Operanden + oder -. Die einzelnen Werte werden entsprechend der verwendeten Operatoren addiert oder subtrahiert und ergeben den Zahlenwert.

Ein fuehrendes Minus, wie in -x, wird wie Ø-x verrechnet.
Ein fuehrendes Plus, wie +x, wird verrechnet als x'+x, wobei x' der Wert des vorhergehenden Ausdrucks ist.

Beispiele:

```
1ØØ
$1ØØ+2-'A'
.T1+5
```

Weitere Beispiele werden zu den einzelnen Kommandos gegeben.

9.2.1. A-Kommando

Das A-Kommando gestattet ein befehlsweises Sofortuebersetzen. Es ist zu beachten, dass die Quellanweisungen in Z80-Mnemonik erfolgen muessen (Unterschiede zur U880-Mnemonik!).

Moegliche Formen des A-Kommandos:

```
(1) As
(2) A
(3) -A
```

Bei (1) beginnt die Assemblierung ab der Speicheradresse s, die nach der Uebersetzung des Befehls um die Laenge des Befehls erhoeht wird.
(2) beginnt bei der aktuellen Adresse entsprechend der letzten Assembler-, Trace- oder Listadresse.
Bei Form (3) werden Assembler- und Disassembler-Teil von DU ueberschrieben, damit wird der freie Speicherbereich um etwa

4K Byte erhoeht. Nachfolgend sind keine A- und L-Kommandos mehr moeglich, bei Protokollierung (T-, U-Kommandos) wird nur der Befehlscode angezeigt.

Das A-Kommando wird durch die Eingabe eines Punktes "." anstelle des naechsten Befehls beendet.

Beispiele:

```
A1ØØ
A$1ØØ
A.T1+5
A@T1+@T2-=T3
A+1Ø
```

Nachfolgend ein Beispiel fuer die Befehlseingabe. Die Operanden koennen aus symbolischen Adressen bestehen, wenn die zugehoerige Symboltabelle eingegeben wurde.

Angenomen, es wurde

```
A1ØØ
```

eingegeben und die Symbole T1 und T2 existieren, so sind folgende Eingaben gueltig:

```
LD   A,B
LD   A,ØFC
LD   B,$149
LD   (HL),'a'
LD   HL,'ab'
JP   2ØØ
CALL .T1
JP   z,@T2
LD   HL,@T1+=T2
```

Es ist zu beachten, dass keine Pseudo-Operationen erlaubt sind und Marken nicht eingefuegt werden koennen. LD HL,'ab' fuellt H mit 'a' und L mit 'b'.

Bei Relativspruengen muss die Angabe der Zieladresse folgen.

9.2.2. C-Kommando

Das C-Kommando bewirkt einen direkten Unterprogrammsprung an die Speicheradresse s. Nach Abarbeitung des Unterprogrammes erfolgt die Rueckkehr zu DU (Kommandoeingabe).

Form des C-Kommandos:

```
(1) Cs
(2) Cs,b
(3) Cs,b,d
```

(1) startet das Unterprogramm ab Adresse s mit BC = 0000 und DE = 0000, (2) mit BC = 0000 und (3) mit BC = b, DE = d.

Der CPU-Status des zu testenden Programmes wird durch das C-Kommando jedoch nicht veraendert.

Beispiele:

```
C.T1
C20A
C103,$14
C.T1+4,0'A'
```

8.2.3. D-Kommando

Das D-Kommando bewirkt die Anzeige des Inhaltes eines zusammenhaengenden Speicherbereiches in Hexa-Darstellung und als ASCII-Zeichen. In einer Zeile wird eine Speichergruppe bis zur Endadresse f (max. 16 Byte) angezeigt, darunter in der Folgezeile das entsprechende ASCII-Zeichen. Nichtdruckbare Zeichen werden als Punkt dargestellt.

Formen des Kommandos:

```
(1)  Ds
(2)  Ds,f
(3)  D
(4)  D,f
```

Bei (1) beginnt die Anzeige ab Adresse s, es werden sechs aufeinanderfolgende Speichergruppen (12 Zeilen) angezeigt.
Bei (2) endet die Anzeige bei Adresse f.

Bei (3) erfolgt die Anzeige entsprechend (1), aber ab dem zuletzt angezeigten Speicherplatz oder dem Inhalt des HL-Registers nach vorherigem T- oder U-Kommando.
Bei (4) erfolgt die Anzeige mit der Anfangsadresse wie bei (3) und endet bei der Adresse f.

Ausserdem besteht die Moeglichkeit der Anzeige im Adress-Format, d.h. zwei aufeinanderfolgende Bytes werden als Adresse interpretiert und (gedreht) in der Darstellung H-, L-Teil angezeigt. Die zugehoerigen ASCII-Zeichen werden speichergerecht angezeigt.

Formen:

```
(5) DWs
(6) DWs,f
(7) DW
(8) DW,f
```

(5) bis (8) wirken entsprechend (1) bis (4).

Beispiele:

```
DF3F
D$100,$200
D.T1,.T2+$20
d,.T1
DW@T1,+$96
```

8.2.4. F-Kommando

Das F-Kommando bewirkt, dass einem zusammenhaengenden Speicherbereich, beginnend mit Adresse s und endend mit Adresse f der Wert d (1 Byte) zugewiesen wird.

Formen des Kommandos:

```
Fs,f,d
```

Beispiele:

```
f200,+$10,41
f200,+3,'B'
F.T1,.T2,=T1
```

8.2.5. G-Kommando

Mit dem G-Kommando erfolgt die Abarbeitung des zu testenden Programmes im Echtzeitbetrieb.

```
(1) G
(2) Gp
(3) G,a
(4) Gp,a
(5) G,a,b
(6) Gp,a,b
```

(1) startet das zu testende Programm ab der aktuellen Befehlsadresse.

Bei (2) erfolgt der Programmstart ab Befehlsadresse p.
(3) wirkt wie (1), es wird aber ein temporaerer Haltepunkt bei a gesetzt.

Bei (4) wird das zu testende Programm von p bis zur Adresse a abgearbeitet.
(5) und (6) wirken wie (3) und (4), es kann aber ein zweiter Haltepunkt b angegeben werden.

Beim Erreichen eines Haltepunktes (oder der Abarbeitung eines RST7 im zu testenden Programm) wird seine Adresse in der Form

```
*nnnn
```

angezeigt und die Haltepunkte werden geloescht,d.h. es ist gegebenenfalls eine Neueingabe erforderlich. Der bei der Haltepunkt-Adresse beginnende Befehl ist dabei noch nicht ausgefuehrt.

Sind Pass-Punkte gesetzt (s. P-Kommando), so erfolgt bei (1)-(6) bei jedem Durchlauf eines Pass-Punktes eine Protokollierung. Hat der Pass-Zaehler den Wert 1, so erfolgt ein Programmstop und damit das Beenden des G-Kommandos. Soll bei gesetzten Pass-Punkten eine Protokollierung nur erfolgen, wenn der zugehoerige Pass-Zaehler den Wert 1 besitzt, so ist analog (1)-(6)

```
(7)-G...
```

als Kommando einzugeben.
Ist ein Haltepunkt in einem Unterprogramm erreicht, so setzt

```
G,^
```

einen naechsten Haltepunkt an der Rueckkehradresse in dem das Unterprogramm aufrufenden Programm.

Beispiele:

```
G100
G100,103
G.T1,.T2,#1416
G@T1+=T2,.A1,.A2
-G100,+20,+20
```

8.2.6. H-Kommando

Das H-Kommando unterstuetzt das Rechnen mit Hexawerten.

Formen des Kommandos:

```
(1) Ha,b
(2) Ha
(3) H
```

(1) liefert die hexadezimale Summe (a+b) und die Differenz (a-b) der Operanden a und b.
(2) liefert den Wert von a im Format

```
hhhh .ssss #ddddd
```

,wobei hhhh den Hexawert von a, .ssss die symbolische Adresse (falls vorhanden) und #ddddd den Dezimalwert bezeichnen.
(3) gibt eine Liste der Adressen jedes Elements der Symboltabelle (falls vorhanden) hexadezimal aus.

Beispiele:

```
H200,300
H#2000,#1617
H.T1+=T2,@T3-#10
H#67
H@T1+=T2-5
```

8.2.7. I-Kommando

Mit dem I-Kommando kann eine Zeichenfolge c1c2...cn angegeben werden, die wie eine vom CCP bereitgestellte Kommandozeile behandelt wird.

Form des Kommandos:

```
Ic1c2...cn
```

Die Zeichenfolge wird mit vorangestellter Zeichenanzahl n ab 80H abgelegt und der Standard-FCB initialisiert. Ist als Zeichenfolge eine Datei spezifiziert, kann sie mit einem nachfolgenden R-Kommando in den Speicher gelesen werden.
Speziell kann ein zu testendes Programm (COM-/SYM-Datei) nachgeladen werden.

Beispiele:

```
Ix.dat
ix.inp y.out
Ia:x.inp  b:y.out
ITEST.COM
ITEST.COM TEST.SYM
```

8.2.8. L-Kommando

Das L-Kommando wird benutzt, um Speicherinhalte in Assemblersprachmnemonik aus einem bestimmten Programmbereich aufzulisten.

Formen des Kommandos:

```
(1) Ls
(2) Ls,f
(3) L
(4) -L...
```

(1) listet den disassemblierten Maschinencode, beginnend bei s, (2) von s bis f. (3) beginnt ab der letzten gelisteten, assemblierten oder getracten Adresse.

(4) wird wie (1)-(3) angewandt, aber Marken und symbolische Operanden werden nicht ausgegeben.

Nicht-Z80-Mnemoniks werden als

```
??=hhhh
```

ausgegeben, wobei hhhh der Hexawert des entsprechenden Speicherplatzes ist.

```
L100
L$1213,$1304
L.TEST
L@T1,+20
-L.PUFF+=T1,+'B'
```

8.2.9. M-Kommando

Mit dem M-Kommando koennen zusammenhaengende Speicherbereiche verschoben werden.

Form des Kommandos:

```
Ms,h,d
```

Das Kommando transportiert den Inhalt des Speicherbereiches mit der Anfangsadresse s und der Endadresse h byteweise nach dem bei der Adresse d beginnenden Bereich.

Beispiele:

```
M100,20F,400
M.A,.B,.C
M.AB,+FF,.CD
M@A+=B,+$30,+200
```

8.2.10. P-Kommando

Mit dem P-Kommando koennen bestimmte Stellen im zu testenden Programm als "Pass-Punkte" definiert werden. Jeder Pass-Punkt erhaelt einen Zaehler, den Passzaehler (1-FF), der jedesmal dann dekrementiert wird, wenn das Programm den am Pass-Punkt beginnenden Befehl ausfuehrt.

Hat der Zaehler den Wert 1 erreicht, wird der Pass-Punkt zum staendigen Protokollier- und Haltepunkt und behaelt den Zaehler-Wert 1.

Im Gegensatz zum temporaeren Unterbrechungspunkt beim G-Kommando unterbrechen die Pass-Punkte mit dem Pass-Zaehler 1 das Abarbeiten des zu testenden Programms nach Abarbeiten des am Pass-Punkt beginnenden Befehls.

Formen des P-Kommandos:

```
(1) Pp
(2) Pp,c
(3) P
(4) -Pp
(5) -P
```

(1) setzt einen Pass-Punkt auf Adresse p mit dem Zaehler-Anfangswert 1, (2) initialisiert den Zaehler-Anfangswert c (1-FF).
Form (3) bringt die aktivierten Pass-Punkte mit den zugehoerigen Zaehlerwerten zur Anzeige.
(4) loescht den Pass-Punkt auf p, waehrend bei (5) alle Pass-Punkte geloescht werden.

Es koennen bis zu 8 Pass-Punkte gleichzeitig aktiviert werden. Beim Erreichen eines Pass-Punktes erfolgt bei Protokollierung eine Ausschrift der Art

```
nn PASS hhhh .ssss
```

,wobei nn den Pass-Zaehlerwert, hhhh die Pass-Punkt-Adresse und ssss das entsprechende Symbol des Pass-Punktes (falls vorhanden) angeben. Dazu werden die CPU-Register-Inhalte angezeigt, ausser bei den Kommandos -G oder -U, falls der Pass-Zaehler groesser 1 ist.

Beispiele:

```
P200,ff
P.T1
P@T1+30,$11
-P.T1
```

8.2.11. R-Kommando

Das R-Kommando laedt die beim I-Kommando angegebenen Dateien.

Formen des Kommandos:

```
(1)  R
(2)  Rd
```

(1) liest die mit dem I-Kommando spezifizierte Datei und speichert sie entsprechend ihrer Ladeadresse, mit (2) wird zur Ladeadresse d addiert.

Mit der Kommandofolge

```
Ix.COM x.SYM
R
```

werden sowohl die COM-Datei x als auch die zugehoerige Symbol-

tabelle in den Speicher geladen.

Soll die Symboltabelle nachgeladen werden, ohne die bereits gespeicherte COM-Datei zu ueberschreiben, dann muss die Kommandofolge

```
I* x.SYM
R
```

eingegeben werden.

Wenn eine Symbol-Datei spezifiziert wurde, zeigt die Ausschrift

SYMBOLS

den Start der Leseoperation. Ein Fragezeichen "?" vor der Ausschrift SYMBOLS kennzeichnet einen Fehler beim Lesen des Maschinencodes, waehrend ein "?" nach der Ausschrift einen Fehler beim Lesen der Symbol-Datei anzeigt.

Beispiele:

```
ITEST.COM
R
ITEST.COM TEST.SYM
R1000
I* TEST.SYM
R-$256
```

9.2.12. S-Kommando

Mit dem S-Kommando koennen Speicherplaetze geprueft und beliebig geaendert werden.

Formen des Kommandos:

```
(1) Ss
(2) SWs
```

Form (1) gibt den Speicherplatz s im Byte-Format an, (2) im Adress-Format, d.h. sei:

```
S100 - 34
101 - 12
```

,so erfolgt bei unveraenderten Speicherinhalten mit Form (2) folgende Anzeige:

```
SW100 - 1234 .
```

Wenn keine Eingabe erfolgt, wird der Inhalt nicht veraendert und die naechste Adresse angezeigt. Wird ein Wert eingegeben, so ueberschreibt er den vorherigen Inhalt und die naechste Adresse wird angezeigt.

Das Kommando endet mit irgendeiner fehlerhaften Eingabe oder der Eingabe eines Punktes "." .

Mit (1) koennen auch ASCII-Zeichenketten eingegeben werden. Dies geschieht durch Eingabe eines Anfuehrungszeichens ("), gefolgt von der Zeichenkette. Beendet wird die Eingabe durch <ENTER>.

In den folgenden Beispielen werden die Tastatureingaben durch Unterstreichung gekennzeichnet.

```
S100
0100 C3 34
0101 24 $253
0102 BD _
0103 4A "Test
0107 2E =I1+5
0108 14 .

$W.I1+$10
1200 005A 33F
1202 4A12 @TEST
1204 1101 _
1206 AB1C @+.I1+=I2-$10
1208 186B .
```

9.2.13. T-Kommando

Das T-Kommando gestattet wahlweises Protokollieren der Programmabarbeitung von 1 bis 65535 Programmschritten.

Formen:

```
(1) Tn
(2) T
(3) Tn,c
(4) -T...
(5) TW...
(6) -TW...
```

Form (1) protokolliert n Programmschritte, wobei der CPU-Status bei jedem Schritt angezeigt wird, waehrend (2) nur einen Schritt protokolliert.

Form (3) wird in Verbindung mit den DU-Zusatzfunktionen HIST und TRACE benutzt und "ruft" die Zusatzfunktion c bei jedem Schritt.

Form (4) wirkt wie (1)-(3), aber ohne Anzeige der Symbole. Form (5) wirkt ebenfalls wie (1)-(3), aber ohne Protokollierung der durch "call" aufgerufenen Unterprogramme. (6) wirkt wie (5), aber ohne Symbolanzeige.

Beispiele:

```
T200
T#10,.COLLECT
```

Das Protokollieren erfolgt bis zum Erreichen von BDOS und wird nach Rueckkehr aus dem BDOS-Teil ins aufrufende Programm fortgesetzt.

8.2.14. U-Kommando

Das U-Kommando wirkt wie das T-Kommando mit der Ausnahme, dass grundsaetzlich Zwischenprogrammschritte nicht protokolliert werden.

Formen des Kommandos:

```
(1) U...
(2) -U...
(3) UW...
(4) -UW...
```

U fuehrt die gleichen Funktionen wie T aus, es wird aber nur der CPU-Status vor Ausfuehrung des ersten Befehls angezeigt. Bei (2) und (4) jedoch erfolgt keine Protokollierung dazwischenliegender Pass-Punkte, wenn deren Zaehler den Wert 1 noch nicht erreicht hat.

Beispiele:

```
Ufa24
U#100,.COLLECT
UW=SUMME,.COLLECT
```

8.2.15. X-Kommando

Mit dem X-Kommando kann der CPU-Status angezeigt und veraendert werden.

Formen des Kommandos:

```
(1) X
(2) Xf
(3) Xr
```

(1) zeigt den CPU-Status im Format:

```
f A=a B=b D=d H=h S=s P=p i y
  A'  B'  D'  H'  X   Y
```

an, wobei f den Flag-Status, a den Inhalt des Akkumulators A, b den Inhalt des Doppelregisters BC, d den DE-Inhalt, h den Inhalt von HL, s den Wert des Stack-Pointers und p den PC-Wert

bezeichnet. i stellt die Mnemonik des bei p beginnenden Befehls und y die symbolische Adresse dar.
A', B', D', H' bezeichnen den Zweitregistersatz und X, Y die Doppelregister IX und IY.
Der Flag-Status wird durch Striche "-" bei Wert 0 dargestellt oder den entsprechenden Kennbuchstaben (Carry Zero Minus Even parity Interdigit carry) bei Wert 1. (M,E und I entsprechen den ASM-Bezeichnungen S,P und H).

(2) ermoeglicht eine Aenderung des Flag-Status, f ist einer der Buchstaben C, Z, M, E oder I. Der aktuelle Status wird angezeigt (entweder "-" oder der entsprechende Buchstabe) und eine 1 oder 0 eingegeben bzw. bei einer Leereingabe der Status beibehalten.

(3) ermoeglicht ein Aendern des Registerinhaltes, wobei r einer der Buchstaben A, B, D, H, S, A', B', D', H', X, Y oder P ist. Die Form "=mm" wird bei speicherbezogenen Befehlen (z.B. ADD A,(HL)) angezeigt, wobei mm der Speicherinhalt vor Ausfuehrung des Befehls ist.

Beispiele:

(Eingabewerte unterstrichen)

```
XM
M       0
XB
2A04    1B23
XP
0100    .IEST+3
```

8.3. DU-Zusatzfunktionen

Die Zusatzfunktionen von DU gestatten weitere Testmoeglichkeiten. Eine Zusatzfunktion wird zusammen mit DU durch die Eingabe von

```
DU x.UTL
```

geladen, wobei x den Namen der Zusatzfunktion bezeichnet. Nach dem Laden ist die Zusatzfunktion zur Ausfuehrung mit DU bereit und antwortet mit

```
.INITIAL=iiii
.COLLECT=cccc
.DISPLAY=dddd
```

,wobei iiii, cccc und dddd die Absolutadressen der Eintrittspunkte der Zusatzfunktion fuer die Initialisierung, das Sammeln von Testdaten und die Anzeige der gesammelten Informationen darstellen. Eine interne DU-Symbol-Tabelle enthaelt diese drei symbolischen Eintrittsnamen (auch wenn keine SYM-Datei geladen wurde).

Eine Zusatzfunktion wird durch die Eingabe von

    Ciiii oder G.INITIAL

neu initialisiert. Die Anzeige der gesammelten Informationen erfolgt durch

    Cdddd oder C.DISPLAY ,

waehrend das Sammeln der Daten bei Ausfuehrung eines T- oder U-Kommandos erfolgt, bei dem als zweites Argument die COLLECT-Adresse angegeben ist.

Beispiele:

```
Uffff,.collect
U@1000,AA24
TW500,.COLLECT
UW@TEST,.COLLECT
```

Die spezielle Initialisierung, Datensammlung und Anzeige ist von der jeweiligen Zusatzfunktion abhaengig.

8.3.1. HIST

Die Zusatzfunktion HIST erzeugt ein Histogramm der Programmausfuehrung zwischen zwei bei der Initialiserung vorgegebenen Punkten. Die Daten werden waehrend der Ausfuehrung eines U- oder T-Kommandos gesammelt und koennen zu beliebiger Zeit angezeigt werden.
Nach dem Einlesen oder Neuinitialisieren meldet sich HIST mit der Ausschrift

    TYPE HISTOGRAM BOUNDS:

Es wird die Eingabe zweier Speicheradressen aaaa und bbbb erwartet, zwischen denen ein Histogramm erzeugt werden soll.

Beispiel:

```
DU HIST.UTL
TYPE HISTOGRAM BOUNDS  100.A00
.INITIAL = AA21
.COLLECT = AA24
.DISPLAY = AA27
#I_TEST.COM_TEST.SYM
#R
SYMBOLS
#Uff..collect
(Anzeige  Register  und  Stop  nach  Kommando-
Ausfuehrung)
#C.DISPLAY
(Anzeige Histogramme)
#C.INITIAL
```



```
(Histogramm-Grenzen neu festgelegt)
..........
```

8.3.2. TRACE

Die Zusatzfunktion TRACE ermoeglicht ein Rueckprotokollieren von maximal 256 Befehlen, ab dem aktuellen Unterbrechungspunkt. Die Sammlung der entsprechenden Adressen erfolgt nur mit dem T- oder U-Kommando.

Es koennen Pass-Punkte waehrend der Datensammlung gesetzt sein. Sie fuehren zur Unterbrechung, wenn der Pass-Zaehler den Wert 1 erreicht hat.

Beim Initialisieren werden die gesammelten Informationen geloescht, beim Sammeln der Daten erfolgt das Einspeichern der Adressen der abgearbeiteten Befehle in einen zyklischen Puffer und bei der Anzeige die Rueckprotokollierung in mnemonischer Form mit Symbolbezuegen, wenn sie vorhanden sind.

Wurde vor Aktivierung von TRACE das Kommando -A abgearbeitet, so werden nur die Befehlsadressen protokolliert. In diesem Fall ist TRACE durch folgende Eingabe zu laden:

```
DU
#-A
#I_TRACE.UTL
#R
"-A" IN EFFECT, ADRESS BACKTRACE
...
```

Beispiel:

(fuer die normale Anwendung)

```
DU_TRACE.UTL
READY FOR SYMBOLIC BACKTRACE
#I_TEST.COM_TEST.SYM
#R
#Uff..collect
(Registeranzeige, Stop nach Kommandoausfuehrung)
#C.DISPLAY
(symbolische Rueckprotokollierung wird angezeigt)
..........
```

8.4. Abschlussbemerkungen

DU benoetigt einen Speicherbereich von etwa 10K Bytes und verschiebt sich selbst direkt unter BDOS (ueberlagert den CCP-Bereich). Die Symboltabelle wird speicherabwaerts ab DU-Beginn angelegt. Entsprechend der Tabellenbelegung veraendert sich die BDOS-Sprungadresse, die den reduzierten freien Bereich kennzeichnet.

Die Programme, die einen grossen Speicherbereich benoetigen und die BDOS-Sprungadresse benutzen, sollten nicht vor dem Laden der Symboltabelle gestartet werden.
Das "-A"-Kommando erhoeht den freien Bereich um etwa 4K Bytes.

Anlage A

Steuerzeichen fuer Bildschirm

Dezimal	Hexadez.	Wirkung, wenn das Zeichen auf den Bildschirm gegeben wird
1	01H	Kursor an Anfang des Bildschirmes
7	07H	akustisches Signal von ca. 1,25 s Dauer (nur bei Verwendung der Tastatur K 7637)
8	08H	Kursor ein Zeichen nach links
10	0AH	Kursor eine Zeile nach unten
12	0CH	Bildschirm loeschen und Kursor an den Anfang des Bildschirms
13	0DH	Kursor an den Zeilenanfang
20	14H	Loeschen bis Ende des Bildschirms ab aktueller Kursorposition
21	15H	Kursor ein Zeichen nach rechts
22	16H	Loeschen bis zum Ende der Zeile ab der aktuellen Kursorposition
24	18H	Loeschen gesamte Zeile und Setzen Kursor an den Zeilenanfang
26	1AH	Kursor eine Zeile nach oben
27	1BH	Einleiten einer Kursorpositionierung. Die anschliessenden beiden Bytes geben an: (Y + 80H, X + 80H) Y = Zeilennummer (0 ... 23) X = Zeichenposition (0 ... 79)
127	7FH	an die Kursorposition 00H ausgeben und Kursor ein Zeichen nach links
128	80H	LED auf Tastatur ein, Warten auf Taste, LED aus, Auswertung Tastencode: bei ^C Aufruf Warmstart, sonst RET

130	82H	Kursor einschalten
131	83H	Kursor ausschalten
132	84H	normal
133	85H	invers
134	86H	intensiv
135	87H	intensiv invers.

Anlage B

Steuerzeichen fuer Drucker

Befehl Kurzzeichen	Steuerfolge (X=dez, Posit.)	SD1157 PIO	SD1157 SIO	K6311 K6312 K6316	SD1152 PIO	SD1152 SIO
NOP	00 -- -- -- -- --	x	x	x	x	x
		ohne Funktion				
HTS	1B 48 -- -- -- --	x	x	-	-	x
		Horizontale Tabulationsmarken setzen				
HTC	1B 5B 33 67 -- --	x	x	-	-	x
		Horizontale Tabulationsmarken loeschen				
HTCO	1B 5B 30 67 -- --	-	-	-	-	x
		Loeschen eines Tabulators auf aktueller horizontaler Position				
SP	20 -- -- -- -- --	-	-	-	x	x
		Space				
HT	09 -- -- -- -- --	x	x	-	-	x
		Horizontale Tabulationsmarken anlaufen				
HPRV	1B 5B 3X 3X 3X 61	x	x	x	-	x
		Horizontale Positionierung relativ vorwaerts				
HPRR	1B 5B 3X 3X 3X 71	x	x	x	-	x
		Horizontale Positionierung relativ rueckwaerts				
HPA	1B 5B 3X 3X 3X 60	x	x	x	-	x
		Horizontale Positionierung absolut				
CR	0D -- -- -- -- --	x	x	x	x	x
		Wagenruecklauf				
BS	08 -- -- -- -- --	x	x	x	x	x
		Rueckschritt				

6 LPI	1B 5B 30 20 4C --	x	x	-	-	x
		6 Zeilen / Zoll				
8 LPI	1B 5B 35 20 4C --	x	x	-	-	x
		8 Zeilen / Zoll				
24 LPI	1B 5B 36 20 4C --	-	-	-	-	x
		24 Zeilen / Zoll				
LF 1	0A -- -- -- -- --	x	x	x	x	x
		Zeilenvorschub Bahn 1				
LF 2	11 -- -- -- -- --	x	x	-	x	x
		Zeilenvorschub Bahn 2				
VTS 1	1B 4A -- -- -- --	x	x	-	-	-
		Vertikale Tabulationsmarken fuer Bahn 1 setzen				
VTS 2	1B 51 -- -- -- --	x	x	-	-	-
		Vertikale Tabulationsmarken fuer Bahn 2 setzen				
VTC 1	1B 5B 34 67 -- --	x	x	-	-	-
		Vertikale Tabulationsmarken fuer Bahn 1 loeschen				
VTC 2	1B 5B 35 67 -- --	x	x	-	-	-
		Vertikale Tabulationsmarken fuer Bahn 2 loeschen				
Vt 1	0B -- -- -- -- --	x	x	-	-	-
		Vertikale Tabulationsmarken fuer Bahn 1 anlaufen				
VT2	12 -- -- -- -- --	x	x	-	-	-
		Vertikale Tabulationsmarken der Bahn 2 anlaufen				
VPRV 1	1B 5B 3X 3X 3X 65	x	x	x	-	x
		Vertikale Positionierung, relativ vorwaerts, Bahn 1				
VPRV 2	1B 5B 3X 3X 3X 76	x	x	-	-	x
		Vertikale Positionierung relativ vorwaerts, Bahn 2				
VPRR 1	1B 5B 3X 3X 3X 75	x	x	x	-	x
		Vertikale Positionierung, relativ rueckwaerts, Bahn 1				
VPRR 2	1B 5B 3X 3X 3X 72	x	x	-	-	x
		Vertikale Positionierung, relativ rueckwaerts, Bahn 2				

Befehl		SD1157 PIO	SD1157 SIO	K6311 K6312 K6316	SD1152 PIO	SD1152 SIO
Kurz-zei-chen	Steuerfolge					
VPA 1	1B 5B 3X 3X 3X 64	x	x	x	-	-
		Vertikale Positionierung, absolut, Bahn 1				
VPA 2	1B 5B 3X 3X 3X 74	x	x	-	-	-
		Vertikale Positionierung, absolut, Bahn 2				
LPF 1	1B 5B 3X 3X 3X 7D	x	x	x	-	x
		Formularlaengen fuer Bahn 1				
LPF 2	1B 5B 3X 3X 3X 7E	x	x	-	-	x
		Formularlaengen fuer Bahn 2				
FF 1	0C -- -- -- -- --	x	x	x	x	x
		Formularvorschub Bahn 1				
FF 2	13 -- -- -- -- --	x	x	-	x	x
		Formularvorschub Bahn 2				
LLFS 1	1B 5B 3X 3X 3X 7A	x	x	x	-	-
		Formularendezeile fuer Bahn 1 setzen				
LLFS 2	1B 5B 3X 3X 3X 7B	x	x	-	-	-
		Formularendezeile fuer Bahn 2 setzen				
LLFC 1	1B 30 -- -- -- --	x	x	x	-	-
		Formularendezeile fuer Bahn 1 loeschen				
LLFC 2	1B 31 -- -- -- --	x	x	-	-	-
		Formularendezeile fuer Bahn 2 loeschen				
FTB	1B 34 -- -- -- --	x	x	-	-	x
		Formulartransportbefehl fuer robotron 1164				
ZB 1	1B 35 -- -- -- --	x	x	-	-	x
		Zeilenschaltbefehl fuer robotron 1164, Bahn 1				
ZB 2	1B 36 -- -- -- --	x	x	-	-	x
		Zeilenschaltbefehl fuer robotron 1164, Bahn 2				

Befehl Kurz-zei-chen	Steuerfolge	SD1157 PIO	SD1157 SIO	K6311 K6312 K6316	SD1152 PIO	SD1152 SIO
FSB	1B 37 -- -- -- --	x	x	-	-	x
	Formularsprungbefehl					
ZLB	1B 35 -- -- -- --	x	x	-	-	x
	Zeilenschaltbefehl fuer robotron 1161					
KAB	1B 34 -- -- -- --	x	x	-	-	x
	Kartenauswurfbefehl					
BDE	1B 5B 31 6D -- --	x	x	x	-	-
	Breitschrift Ein					
SDE	1B 5B 33 6D -- --	x	x	x	-	-
	Schraegschrift Ein					
FBU	14 -- -- -- -- --	-	-	-	x	x
	Farbbandumschaltung					
NDR	1B 5B 30 6D -- --	x	x	x	-	-
	Normaldruck					
SO	0E -- -- -- -- --	x	x	-	-	-
	Umschalten ext.Zeichenbereich					
SI	0F -- -- -- -- --	x	x	-	-	-
	Umschalten int.Zeichenbereich					
SLZE	1B 39 -- -- -- --	x	x	-	x	x
	Automatisches Sichtbarmachen letzte Zeile EIN					
SLZA	1B 38 -- -- -- --	-	-	-	x	x
	Automatisches Sichtbarmachen letzte Zeile AUS					
DEL	7F -- -- -- -- --	x	x	x	x	x
	Ruecksetzen					
NL	1B 45 -- -- -- --	x	x	-	-	-
	Neue Zeile Kombination von LF 1 und CR					
CPI 80	1B 5B 30 20 4B --	-	-	x	-	-
	Zeichenbreite 1/10"					
CPI 100	1B 5B 32 20 4B --	-	-	x	-	-
	Zeichenbreite 1/12,5"					

Befehl Kurz-zei-chen	Steuerfolge (N=Zeile)	SD1157 PIO	SD1157 SIO	K6311 K6312 K6316	SD1152 PIO	SD1152 SIO
CPI 120	1B 5B 33 20 4B --	-	-	x	-	-
	Zeichenbreite 1/15"					
10 CPI	1B 5B 30 20 4B --	-	-	-	-	x
	10 Zeichen / Zoll					
12 CPI	1B 5B 31 20 4B --	-	-	-	-	x
	12 Zeichen / Zoll					
60 CPI	1B 5B 36 20 4B --	-	-	-	-	x
	60 Zeichen / Zoll					
UDL	1B 5B 34 6D -- --	-	-	x	-	-
	Unterstreichstrich					
BEZ	1B 5B 3N 3N 3N 74	-	-	x	-	-
	Belegeinzug					
ZBV	1B 5B 3N 3N 3N 76	-	-	x	-	-
	Zeilenschaltung vorwaerts					
ZBR	1B 5B 3N 3N 3N 72	-	-	x	-	-
	Zeilenschaltung rueckwaerts					
BAW	1B 34 -- -- -- --	-	-	x	-	-
	Belegauswurf					
SNE	1B 38 -- -- -- --	-	-	x	-	-
	Schneiden					
DA	1B 5B 30 63 -- --	x	x	x	-	x
	Geraetekennungsabfrage					
DSR	1B 5B 35 6E -- --	x	x	x	-	x
	Sende Statusfolge					
GRA	1B 25 30 -- -- --	-	x	-	-	-
	Grafikmodus einschalten					
GRAA	7F -- -- -- -- --	-	x	-	-	-
	Grafikmodus ausschalten					
GRNL	40 -- -- -- -- --	-	x	-	-	-
	Grafik - Neue Zeile					
STX	02 -- -- -- -- --	-	-	x	-	-
	Start des Textes					

Befehl Kurz-zei-chen	Steuerfolge	SD1157 PIO	SD1157 SIO	K6311 K6312 K6316	SD1152 PIO	SD1152 SIO
ETX	03 -- -- -- -- --	-	-	x	-	-
		Ende des Textes				
STX und ETX sind Steuerzeichen die einen Datenblock einrahmen. Sie werden durch den Drucker beantwortet mit:						
ACK	06 -- -- -- -- --	-	-	x	-	-
		Positive Antwort				
NAK	15 -- -- -- -- --	-	-	x	-	-
		Negative Antwort				
10 CPI	01 -- -- -- -- --	-	-	-	x	-
		10 Zeichen / Zoll				
12 CPI	02 -- -- -- -- --	-	-	-	x	-
		12 Zeichen / Zoll				
FZA	0F -- -- -- -- --	-	-	-	x	-
		Festen Zeichenabstand				
/FZA	0E -- -- -- -- --	-	-	-	x	-
		Variabler Zeichenabstand				
HPRV	09 xx xx -- -- -- L H	-	-	-	x	-
		Horizontale Positionierung relativ , vorwaerts				
HPRR	1B 09 xx xx -- -- L H	-	-	-	x	-
		Horizontale Positionierung relativ , rueckwaerts				
6 LPI	05 -- -- -- -- --	-	-	-	x	-
		6 Zeilen / Zoll				
8 LPI	06 -- -- -- -- --	-	-	-	x	-
		8 Zeilen / Zoll				
LF1R	1B 0A -- -- -- --	-	-	-	x	-
		1 ZS rueckwaerts Bahn 1				
LF2R	1B 11 -- -- -- --	-	-	-	x	-
		1 ZS rueckwaerts Bahn 2				
VPRV 1	0B -- -- -- -- --	-	-	-	x	-
		Vertikale Positionierung relativ vorwaerts, Bahn 1				

Befehl Kurz-zei-chen	Steuerfolge	SD1157 PIO	SD1157 SIO	K6311 K6312 K6316	SD1152 PIO	SD1152 SIO
VPRV 2	12 xx -- -- -- --	-	-	-	x	-
		Vertikale Positionierung relativ vorwaerts , Bahn 2				
VPRR 1	1B 0B xx -- -- --	-	-	-	x	-
		Vertikale Positionierung relativ rueckwaerts, Bahn 1				
VPRR 2	1B 12 xx -- -- --	-	-	-	x	-
		Vertikale Positionierung relativ rueckwaerts, Bahn 2				
LPF 1	1B 0C xx -- -- --	-	-	-	x	-
		Formularlaenge Bahn 1				
LPF 2	1B 13 xx -- -- --	-	-	-	x	-
		Formularlaenge Bahn 2				
FTB	16 -- -- -- -- --	-	-	-	x	-
		1164 Formulartransportbefehl				
KAB	16 -- -- -- -- --	-	-	-	x	-
		1161 Kartenauswurfbefehl				
ZB1	15 -- -- -- -- --	-	-	-	x	-
		1164 Zeilenschaltbefehl				
ZLB	15 -- -- -- -- --	-	-	-	x	-
		1161 Zeilenschaltbefehl				
ZB 2	17 -- -- -- -- --	-	-	-	x	-
		1164 Zeilenschaltbefehl				
FSB	18 -- -- -- -- --	-	-	-	x	-
		Formularsprungbefehl				

Anlage_C

Codetabelle ASCII-Zeichen

Dez.	Hex.	Zeichen	Dez.	Hex.	Zeichen	Dez.	Hex.	Zeichen
000	00H	NUL	043	2BH	+	086	56H	V
001	01H	SOH	044	2CH	,	087	57H	W
002	02H	STX	045	2DH	-	088	58H	X
003	03H	ETX	046	2EH	.	089	59H	Y
004	04H	EOT	047	2FH	/	090	5AH	Z
005	05H	ENQ	048	30H	0	091	5BH	[
006	06H	ACK	049	31H	1	092	5CH	\
007	07H	BEL	050	32H	2	093	5DH	]
008	08H	BS	051	33H	3	094	5EH	^
009	09H	HT	052	34H	4	095	5FH	_
010	0AH	LF	053	35H	5	096	60H	`
011	0BH	VT	054	36H	6	097	61H	a
012	0CH	FF	055	37H	7	098	62H	b
013	0DH	CR	056	38H	8	099	63H	c
014	0EH	SO	057	39H	9	100	64H	d
015	0FH	SI	058	3AH	:	101	65H	e
016	10H	DLE	059	3BH	;	102	66H	f
017	11H	DC1	060	3CH	<	103	67H	g
018	12H	DC2	061	3DH	=	104	68H	h
019	13H	DC3	062	3EH	>	105	69H	i
020	14H	DC4	063	3FH	?	106	6AH	j
021	15H	NAK	064	40H	@	107	6BH	k
022	16H	SYN	065	41H	A	108	6CH	l
023	17H	ETB	066	42H	B	109	6DH	m
024	18H	CAN	067	43H	C	110	6EH	n
025	19H	EM	068	44H	D	111	6FH	o
026	1AH	SUB	069	45H	E	112	70H	p
027	1BH	ESCAPE	070	46H	F	113	71H	q
028	1CH	FS	071	47H	G	114	72H	r
029	1DH	GS	072	48H	H	115	73H	s
030	1EH	RS	073	49H	I	116	74H	t
031	1FH	US	074	4AH	J	117	75H	u
032	20H	SPACE	075	4BH	K	118	76H	v
033	21H	!	076	4CH	L	119	77H	w
034	22H	"	077	4DH	M	120	78H	x
035	23H	#	078	4EH	N	121	79H	y
036	24H	$	079	4FH	O	122	7AH	z
037	25H	%	080	50H	P	123	7BH	{
038	26H	&	081	51H	Q	124	7CH	|
039	27H	'	082	52H	R	125	7DH	}
040	28H	(	083	53H	S	126	7EH	~
041	29H	)	084	54H	T	127	7FH	DEL
042	2AH	*	085	55H	U			

Anlage_D:

Uebersicht der Funktionstastencodes und deren Wirkung fuer die Tastaturen K 7637, K 7636/7634, K 7606/7604 (unter CCP- Regie)

Taste K 7637	Taste K 7636/34 K 7606/04	entspricht	Code	Wirkung
		CTRL I	09	Tabulatortaste
		CTRL Z	1A	Kursor 1 Zeile nach oben
		CTRL J	0A	Wirkung wie ENTER
		CTRL H	08	Kursor 1 Position nach links und Zeichen loeschen
		CTRL U	15	Eingabe wird ungueltig, Eingabe auf neuer Zeile wird erwartet
		CTRL X	18	Loeschen der Zeile, Kursor an Zeilenanfang
		CTRL M	0D	Eingabeende, Kursor auf Anfang neue Zeile
		CTRL L	0C	Bildschirm loeschen und Kursor an Bildschirmanfang
		CTRL C	03	Warmstart

Taste K 7637	Taste K 7636/34 K 7606/04	entspricht	Code	Wirkung
		--	7F	Loeschen des links vom Kursor stehen-Zeichens und Ausgabe des geloeschten Zeichens nach rechts (wirkt nur bei K 7636/34 und K 7606/04)
		CTRL S	13	Zeitweises Unterbrechen/Starten der "rollenden" Ausgabe auf LIST- und KONSOL-Geraet
			1B	---
		CTRL P	10	Start / Stop der Druckausgabe

Anhang E

Belegte Toradressen und ihre Bedeutung

00H ... 0BH - von Zentraleinheit (ZRE) und Tastatur belegt

0CH ... 0FH - CTC auf ZRE, von SCPX nicht verwendet

10H ... 1BH - fuer FD-Adapter belegt

40H - bei Adapter K7025 (BAB) belegt

48H ... 4FH - belegt, falls Adapter K6028.40 fuer serielle Tastatur K7637 verwendet wird

50H ... 5FH - belegt vom Drucker-Adapter

60H ... 6FH - reserviert

90H ... 97H - reserviert fuer LBL/LBS

F0H ... FFH - reserviert fuer Programmier- und Pruefeinrichtungen

Die anderen E/A-Tore sind frei und koennen in Anwenderprogrammen fuer die Arbeit mit anderen Geraeten / Anschluessen verwendet werden.