Siemens Systeme 6·000
AMBOSS 1
Betriebssystem BS1 MP
und Dienstprogramme

Bedienungsanleitung

August 1982

Vorwort

In diesem Handbuch werden Funktionen und Anwendungen des Betriebssystems BS1MP und seiner Dienstprogramme beschrieben.

Außerdem sind auch Beschreibungen für das Verwaltungssystem für Direktzugriffsdateien (RFM bzw. DVS), für die Dienstprogramme EDIT, TXT und SORT und für den Monitor enthalten.

Die Einstellung und Bedeutung des Menüs ist der Bedienungsanleitung "Bildschirmcomputer 6.611" zu entnehmen. Hier sind nur Hinweise zur Programmierung der Software-Schalter und der Bildschirm-Steuerung enthalten.

Einleitung

Das Betriebssystem BS1MP wird im Siemens System 6.611 - A, 6.611 - B und 6.611 - C verwendet.

Das BS1MP stellt die Verbindung zwischen dem Computer und der Außenwelt her. So wird zum Beispiel die Anwahl eines Anwenderprogrammes über Kommandos vorgenommen, die dem Betriebssystem sagen, welches Programm es in den Hauptspeicher laden soll und auf welcher Diskette sich dieses Programm befindet.

Viele Funktionen des Betriebssystems lernt der Bediener gar nicht kennen, da sie von den Anwenderprogrammen benutzt werden, ohne daß dies nach außen hin sichtbar ist.

Das BS1MP besteht aus folgenden Teilen:

- dem Monitor mit dem Dateinamen ZZMON
- dem E/A-System für sequentielle Dateien
- dem E/A-System für Direktzugriffsdateien RFM (= Random File Manager) bzw. DVS (wahlweise)
- dem Kommando- und Assignment-Interpreter, durch den die Bedienoberfläche geschaffen wird
- Dienstprogramme
- SPOOL-Funktion (wahlweise) zum Drucken von Listdateien als Hintergrundprozeß (s. 2.19)
- Assembler

Der Monitor muß beim Start der Anlage je nach Menüeinstellung von einer Diskette bzw. von Festplatte geladen werden. Eine seiner Funktionen besteht darin, nach Anschalten der Anlage den ladbaren Teil des Betriebssystems von der Diskette bzw. Festplatte zu lesen und im RAM-Speicher abzulegen.

Das sequentielle E/A- (Eingabe/ Ausgabe-)system des BS1MP regelt die Zugriffe von Anwender- und Dienstprogrammen auf die sequentiellen Dateien der unterschiedlichen Disketten-Formate bzw. der Festplatte. Das sequentielle E/A-System wird auch als Handler bezeichnet.

Die Behandlung von indexsequentiellen und direkten Dateien wird durch einen Baustein vorgenommen, der Teil des Betriebssystems ist, nach Bedarf jedoch von Anwenderprogrammen überlagert werden kann. Die detaillierte Beschreibung befindet sich in Kapitel 7 bzw. 8.

Der Kommandointerpreter schafft eine Verbindung zwischen dem Bediener und dem Anwendungs- oder Dienstprogramm. Er prüft die Eingaben des Bedieners auf ihre Richtigkeit und übergibt dem aufgerufenen Programm die Parameter, die es für seine Arbeit benötigt.

Um den Wünschen und der Anlagenausstattung der Anwender möglichst gerecht zu werden, wird das BS1MP in verschiedenen Konfigurationen ausgeliefert. Es besteht die Möglichkeit, zwischen einem BS1MP mit residentem Handler, RFM, DVS oder SPOOL-Funktion zu wählen. Die Größe des BS1MP ist abhängig von den gewählten residenten bzw. transienten Teilen und den Zusatzfunktionen . Eine nähere Beschreibung der Generiervarianten des BS1MP befindet sich in Kapitel 6.1.

Neben der Beschreibung des Betriebssystems enthält das Handbuch eine Anleitung für die Benutzung der verschiedenen Dienstprogramme, die zum großen Teil für die Dateiverwaltung (Sichern von Dateien, Erstellen von Dateilisten, Kopieren von Datenträgern usw.) benötigt werden.

In Abschnitt 3 werden die Komponenten des ASSEMBLER-Sprachsystems beschrieben, die standardmäßig mit der System-Diskette geliefert werden.

In Abschnitt 4 befindet sich die Beschreibung des Texteditors EDIT, der ebenfalls Bestandteil des BS1MP ist.

Lesehinweise

In diesem Handbuch werden einige Symbole und Abkürzungen benutzt, welche die folgenden Bedeutungen haben:

< > Steht ein Text in spitzen Klammern, so bedeutet dies, daß der Bediener an dieser Stelle einen Namen, eine Zahl, ein Symbol usw. eingeben soll.

Steht im Text z.B. < Programmname >

so muß der Bediener an dieser Stelle einen existierenden Programmnamen - z.B. DIR - eingeben.

CR CR ist eine Taste (CARRIAGE RETURN = Wagenrücklauf, auf der Tastatur mit  dargestellt), mit der grundsätzlich alle Kommandos an das Betriebssystem, die auf der Tastatur eingegeben werden, abgeschlossen werden müssen. Die Eingabe hat außerdem zur Folge, daß die Schreibmarke (Cursor) auf dem Bildschirm an den Anfang der nächsten Zeile springt.

Die Taste gibt den Code 0DH aus.

LF LINE FEED, Code 0AH. Auf der Tastatur wird LF durch  dargestellt.

SP Leerzeichen (Space). Die Taste  gibt den Code 20H aus.

{ } Geschweifte Klammern werden gelegentlich bei der Beschreibung von Programmaufrufen benutzt. Eine der untereinanderstehenden Varianten zwischen den Klammern muß vom Anwender beim Aufruf ausgewählt werden.

[ ] Eckige Klammern werden bei der Beschreibung von Programmaufrufen benutzt. In eckigen Klammern stehende Teile eines Kommandos können, müssen jedoch nicht verwendet werden.

Inhaltsverzeichnis

1 Betriebssystem BS1MP

In diesem Abschnitt wird beschrieben, in welcher Weise Kommandos an das Betriebssystem aufgebaut sind. Die Kenntnis des allgemeinen Kommandoaufbaus und der damit verbundenen Möglichkeiten ist die Voraussetzung, um die Dienstprogramme, die in Abschnitt 2 beschrieben werden, sinnvoll und möglichst effektiv einsetzen zu können.

1.1 Laden des BS1MP

Beim Einschalten der Netzspannung wird zunächst der Monitor ZZMON und dann das System-Vorspannprogramm SYSTEM von dem im Menü angegebenen Laufwerk geladen. Ist auf dem System-Laufwerk die Datei SYSDAT nicht vorhanden, wird die Datei BS611.11 (BS1MP mit SPOOL und DD=resident) geladen. Ansonsten werden die in SYSDAT angegebenen Programmaufrufe durchgeführt. SYSDAT ist eine mit dem Editor erstellte Datei, die verschiedene Dienstprogramme, Cobol-Programme oder eine Emulation enthalten kann. Auch das Laden der gewünschten BS1MP Generiervariante (s. 6.1) ist möglich. Beim Auftreten eines Fehlers wird das Vorspannprogramm sofort unterbrochen, der letzte Programmaufruf und die Systemfehlermeldung werden ausgegeben. Nach einer Tastatureingabe wird das jeweilige BS1MP gestartet.

Auf dem Bildschirm ist nicht der Ladeaufruf, sondern sind nur eventuelle Ausführungsmeldungen sichtbar. Diese Meldungen werden beim Start des BS1MP überschrieben.

Betriebssystemaufruf als Gerade Eingabe bei Datei "Sysdat"

Die Betriebssystemmeldung lautet:

SIEMENS BS1MP REV A.xy[Konfig.] / ZZ:ZZ:ZZ

REV A.xy ist der aktuelle Ausgabebestand des Systems.
"Konfig." kennzeichnet Zusatzfunktionen der geladenen Generiervariante des BS1MP. ZZ:ZZ:ZZ gibt die aktuelle Uhrzeit an, falls die entsprechenden Zellen beim Systemhochlauf versorgt wurden.

Anschließend erscheint das Eingabezeichen "*" als Anzeige, daß Kommandos durch Tastatureingabe erwartet werden.

BS1mp.doc = Datei Information die noch nicht dokumentiert sind.

Hinweis: Bei der 6.611-C muß das BS1MP zuerst mit dem Dienstprogramm CONFIG auf der Festplatte installiert werden.

1.2 Programmaufruf

Die Programme, welche unter BS1MP ablaufen sollen, sind entweder auf Festplatte oder auf Disketten im BS1M-Format (1 MByte-Floppy) oder im INTEL-Format (256 KByte-Floppy) abgespeichert.

Sie werden durch Namen identifiziert.

Der Ablauf eines Programmes wird durch Eingabe seines Namens und |CR| angestoßen. Falls sich die Diskette mit dem Programm nicht in dem Laufwerk befindet, von dem ursprünglich das Betriebssystem geladen wurde oder wenn sie in einem anderen Format beschrieben ist, muß beim Aufruf die Laufwerksbezeichnung ebenfalls angegeben werden.

Programmnamen können selbst dann in Kleinbuchstaben aufgerufen werden, wenn der Name bei Anlage des Programms groß geschrieben wurde. Voraussetzung ist aber, daß nicht bereits Programme mit demselben - jedoch kleingeschriebenen - Namen existieren. Das BS1MP sucht zunächst nach dem kleingeschriebenen Namen, danach, falls unauffindbar, nach dem großgeschriebenen.

In den meisten Fällen genügt es jedoch nicht, ein Programm einfach aufzurufen, normalerweise benötigt es, um arbeiten zu können, irgendwelche weiteren Informationen. Die Art, in der diese Informationen dem Programm übergeben werden, soll zunächst an Beispielen gezeigt werden.

Wir nehmen an, wir hätten eine Textinformation (z.B. einen Brief oder ein Source-Programm) auf einer Diskette gespeichert. Diese (physikalische) Datei enthält also eine Anzahl von Zeilen, jede abgeschlossen durch die beiden Zeichen CR LF (Wagen-Rücklauf/Zeilenvorschub).

Auf einer anderen Diskette (einer System-Diskette), befinde sich ein Programm namens COPY (ein Dienstprogramm des BS1MP), mit dem die Datei auf eine andere Diskette kopiert werden soll.

Wir starten den Aufruf, indem wir eingeben:

```
COPY
```

Im nächsten Schritt müssen wir die Datei angeben, die wir kopieren wollen. Sie möge den Namen TEXT haben und sich auf einer INTEL-formatierten Diskette im Laufwerk 1 befinden. Wir geben also zur Zuweisung der Eingabedatei weiter ein:

```
SI=:F1:TEXT
```

Hierbei bedeutet:

SI= daß die logische Einheit Standard-Eingabedatei zugewiesen wird

:F1: daß sich die Eingabedatei physikalisch auf der Diskette im Laufwerk 1 befindet (physikalische Einheit) und daß diese Diskette das sog. INTEL-Format hat (für andere Datenträger und andere Formate werden andere Buchstaben verwendet, s. 1.2.1)

TEXT ist der Name, unter dem die Eingabedatei gespeichert ist

Das COPY-Programm benötigt außerdem eine Ausgabedatei, in die kopiert werden soll. Da wir auf die Diskette im BS1M-Format im Laufwerk 0 kopieren wollen und die Kopie TEXT.ALT heißen soll, geben wir ein:

```
,SO=:D0:TEXT.ALT
```

Das Komma ist notwendig, um die beiden Zuweisungen voneinander zu trennen.

Weiter bedeutet

SO= die Zuweisung der logischen Einheit Standard-Ausgabe-datei

:DO: daß die Ausgabedatei auf einer BS1M-formatierten Diskette (D) im Laufwerk 0 angelegt werden soll

TEXT.ALT Name, unter dem die Ausgabedaten gespeichert werden sollen

Insgesamt sieht unser Kommando also folgendermaßen aus:

```
COPY SI=:F1:TEXT,SO=:DO:TEXT.ALT      (1)
```

Wenn wir jetzt die [CR]-Taste drücken, wird das Kommando ausgeführt. Dazu veranlaßt das BS1MP die folgenden Aktivitäten:

1. Prüfung, ob eine Datei namens COPY (d.h. das Programm) auf der Diskette vorhanden ist, von der das Betriebssystem geladen wurde. Wird es gefunden, so wird es in den Speicher geladen, andernfalls erfolgt eine Fehlermeldung.

2. Prüfung, ob eine Datei namens TEXT auf der INTEL-Diskette im Laufwerk 1 vorhanden ist. Wenn nicht, erscheint eine Fehlermeldung.

3. Prüfung, ob keine Datei namens TEXT.ALT auf der BS1M-formatierten Diskette im Laufwerk 0 vorhanden ist (denn TEXT.ALT soll ja erst durch COPY erstellt werden). Wenn vorhanden, erscheint die Meldung

   ```
   :DO:TEXT.ALT   EXISTS. DELETE (Y/N):
   ```

   Die bereits existierende Datei kann durch Eingabe "Y" gelöscht oder das Programm mit "N" beendet werden.
   War im Laufwerk 0 keine BS1M-formatierte Diskette, wird eine Fehlermeldung ausgegeben.

4. Start des Programmes COPY.

Nach dem Ablauf des Programmes geht das System bis zur Eingabe eines neuen Kommandos in den Wartezustand.

Dasselbe Programm COPY kann auch für die Ausgabe einer Datei auf dem Drucker verwendet werden. Die physikalische Gerätezuweisung für den Drucker lautet :LP:.

Das Kommando für dieses Beispiel hieße

```
COPY SI=:F1:TEXT,SO=:LP:          (2)
```

Kommandos können auf dem Bildschirm korrigiert werden, solange sie nicht mit CR abgeschlossen worden sind. Mit den Tasten → und ← kann der Cursor beliebig in der Kommandozeile positioniert werden, die Zeichen werden dabei nicht gelöscht.

Mit DEL kann die gesamte Eingabezeile gelöscht werden.

Mit → CHAR kann der Teil des eingegebenen Kommandos, der rechts neben dem Cursor steht, um eine Stelle nach rechts geschoben werden. Dabei werden neben dem Cursor Leerzeichen eingefügt.

Mit ← CHAR kann der rechte Teil des Kommandos um eine Stelle nach links verschoben werden. Das Zeichen neben dem Cursor wird dabei ausgefügt.

Bei Eingabe von CR wird die Zeile unabhängig von der Cursorposition in voller Länge an das Betriebssystem zur Ausführung übergeben.

Das letzte abgeschickte Kommando kann durch die Eingabe von ↖ wieder auf dem Bildschirm ausgegeben und dann - gegebenenfalls verändert - wieder abgeschickt werden.

Die allgemeine Form einer Kommandoeingabe hat folgendes Aussehen:

```
Programmname   Geräte- bzw.Dateizuweisung-1 ,...
Geräte- bzw. Dateizuweisung-n $ Parameter-1 ,...
Parameter-i
```

n ist hier die Anzahl der Zuweisungen, i die Anzahl der Parameter.

Das Kommando muß mit CR abgeschlossen werden und darf nicht über eine Zeile hinausgehen.

Welche Datei- bzw. Programmnamen zulässig sind, hängt u.a. vom Format des Dateiträgers ab. Dies wird detailliert im Abschnitt 1.2.1 erklärt. Dort findet man auch die Liste der möglichen logischen und physikalischen Einheiten.

Parameter sind in den obigen Beispielen nicht aufgetreten, siehe dazu 1.2.2, Beispiel (3).

1.2.1 Zuweisung logischer Einheiten zu Geräten und Dateien

Die meisten Dienstprogramme benötigen Informationen über Dateien und Geräte, mit denen sie arbeiten sollen. Die Programme arbeiten zunächst mit logischen Einheiten, denen in einem Kommando ein physikalisches Gerät bzw. eine Datei zugeordnet wird.

In Beispiel 1 sind SI und SO logische Einheiten, denen :F1:TEXT und :DO:TEXT.ALT zugeordnet wurden, in Beispiel 2 ist die Ausgabedatei der Drucker.

Alle Zuweisungen eines Programmes werden sequentiell von links nach rechts abgearbeitet.

Es gibt folgende logische Einheiten:

```
    CI - Tastatureingabe
    CO - Bildschirmausgabe
    SI - Standardeingabe
    SO - Standardausgabe
    SL - Standardauflistung
    AI - Hilfseingabe
    AO - Hilfsausgabe
    AL - Hilfsauflistung
 I0-I3 - zusätzliche Eingabedateien
 L0-L3 - zusätzliche Listdateien
 O0-O3 - zusätzliche Ausgabedateien
    RP - Antworteingabe
```

Ein Programm kann über diese logischen Einheiten also maximal 7 Eingabe- und 12 Ausgabedateien benutzen, wenn man von CI und CO absieht, die praktisch nie eine andere als die oben genannte Bedeutung haben (mit Ausnahme der Verwendung in EXEC, s. 2.12). Es sei hier schon erwähnt, daß man auf Dateien auch auf eine andere Weise zugreifen kann, bei der es nahezu keine Beschränkung ihrer Anzahl gibt (Abschnitt 6.3).

Jeder logischen Einheit kann eine der im folgenden beschriebenen physikalischen Einheiten zugeordnet werden. Sämtliche physikalischen Einheiten können auch klein geschrieben werden, also z.B. :bb: statt :BB:.

Liste der physikalischen Einheiten

```
:BB:  Ausgabeleerdatei
:NF:  Eingabeleerdatei
:CI:  Tastatur (zeichenweise)
:CE:  wie :CI:, Eingabe von  ETX  bewirkt End-of-File-
      Behandlung durch das Betriebssystem.
:CL:  Tastatur (zeilenweise)
:RP:  Antwort von Tastatur
:CO:  Bildschirm
:LP:  Sequentielles Ausgabegerät (Drucker)
:XM:  Sendung über die Asynchronschnittstelle
:RC:  Empfang von der Asynchronschnittstelle
:RE:  wie :RC:, Empfang von  ETX  bewirkt End-of-File-Be-
      handlung durch das Betriebssystem
```

Den logischen Einheiten CI und RP ist automatisch die physikalische Einheit :CI: zugeordnet, außerdem werden Eingaben automatisch auf :CO: geechot (d.h. CI=:CI:/:CO:, siehe Doppelzuweisungen 1.2.5). Der logischen Einheit CO wird automatisch :CO: zugeordnet.

:CP: weggefallen stattdessen: *(*(:CI:/:CO:)/:LP:)

Erläuterungen

:BB: ist eine Ausgabedatei, die immer eröffnet ist. Ausgaben auf :BB: werden nicht gespeichert. (Für Testzwecke)

:NF: ist eine Eingabedatei, die immer eröffnet ist und bei jedem Lesezugriff Dateiende meldet. (Für Testzwecke)

:CI: bedeutet zeichenweise Eingabe von der Tastatur. Da beim Systemstart CI=:CI:/:CO: angenommen wird, werden Eingaben automatisch auf dem Bildschirm wiedergegeben.

:CO: Beim Systemstart wird der logischen Einheit CO automatisch :CO: zugeordnet, d.h. Ausgaben erfolgen standardmäßig auf dem Bildschirm. Es kann entweder ein Zeichen angezeigt oder ein Kontrollzeichen der Bildschirm- Steuerung übergeben werden.

:CL: ist eine editierfähige Eingabedatei, die normalerweise :CI: zugeordnet ist und auf :CO: geechot wird. Die Zeile muß mit CR abgeschlossen werden. LF wird automatisch angefügt. Die Zeilenlänge beträgt 73 Zeichen.

:RP: ist eine editierfähige Eingabedatei, die normalerweise :CI: zugeordnet ist und auf :CO: geechot wird. Die Zeile muß mit CR abgeschlossen werden. LF wird automatisch angefügt. Die Zeilenlänge beträgt 8 Zeichen.

:XM: Übertragung über die Asynchronschnittstelle. Die Übertragung erfolgt zeichenweise ohne darüberliegende Prozedur. Übertragungsrate, Zeichenrahmen, Parity-Sicherung und Anzahl der Stop-Bits können durch das Menü beeinflußt werden.

:RC: Zeichenweiser Empfang über die Asynchronschnittstelle.

Logische Einheiten können nicht nur Geräten, sondern auch Dateien zugewiesen werden, die sich auf einer Diskette bzw. Festplatte befinden.

Da das BS1MP drei Disketten-Formate unterstützt (1 MByte: BS1M-Format, 256 KByte: INTEL- und ECMA 54-("IBM") Format)und die Festplatte, muß bei der Zuweisung die Art des Formates gekennzeichnet werden.
Zudem muß angegeben werden, in welchem Laufwerk sich der Datenträger mit der entsprechenden Datei befindet.

Folgende Zuweisungen von Dateien sind möglich:

1) <Dateiname>

Wird <Dateiname> eingegeben, um ein Programm aufzurufen, so wird es automatisch vom System-Laufwerk geladen. Das System-Laufwerk ist im Menü wählbar und lädt beim Systemstart das Betriebssystem.
Ist Dateiname nicht der Name eines Programmes, das aufgerufen werden soll, sondern eine Datei, die einer der logischen Einheiten zugewiesen wird, so geht das System davon aus, daß sie sich auf einer BS1M-formatierten Diskette im Laufwerk # bzw. auf der Festplatte befindet.
Der Dateiname darf aus max.6 Buchstaben und Ziffern bestehen und um einen Punkt sowie max. 3 weitere Buchstaben und Zahlen ergänzt werden.

Beispiele für zulässige Namen sind:

1ABCD.3

A66666.UUU

abcdef.ghi

Nicht zulässig ist z.B.:

Kundendatei.001

1/kk.HH

da im ersten Fall der Name zu lang ist, im zweiten Fall das Sonderzeichen "/" vorkommt.

2) :Di:<Dateiname>

<Dateiname> wie unter 1. i kann die Zahl 0 oder 1 sein und gibt das Laufwerk an, in dem sich die Diskette befindet. Vom Betriebssystem wird vorausgesetzt, daß die Diskette im BS1M-Format formatiert ist.

3) :Fi:<Dateiname>

<Dateiname> wie unter 1. i kann die Zahl 0 oder 1 sein und gibt das Laufwerk an, in dem sich die Diskette mit der angegebenen Datei befindet. Vom Betriebssystem wird außerdem vorausgesetzt, daß die Diskette INTEL-formatiert ist.

4) :Hi:<Dateiname> <Dateiname>wie unter 1. Das Betriebssystem setzt eine formatierte Festplatte voraus. i kann die Zahl 0 oder 1 sein, da die Festplatte in zwei logische Einheiten geteilt ist.

5) :Ji:<Dateiname> Das Betriebssystem sucht die Datei <Dateiname> auf der ECMA 54-("IBM-") formatierten Diskette in Laufwerk 0 oder 1. Außerdem erwartet es, daß die Diskette im EBCDI-Code beschrieben ist. Der Dateiname darf aus max. 8 Großbuchstaben und Zahlen bestehen.

1.2.2 Parameterübergabe

Neben der Zuweisung von Geräten bzw. Dateien zu logischen Einheiten kann auch noch die Angabe von weiteren Parametern nötig sein.

Beispiel:

```
DIR SO=:LP:$:D1:          (3)
```

Das Programm DIR erstellt eine Liste aller Dateien einer Diskette. Durch eine Gerätezuweisung wird dem Programm mitgeteilt, wo diese Liste ausgegeben werden soll, in diesem Beispiel auf dem Drucker.

Daneben muß dem Programm gesagt werden, von welchem Datenträger die Liste angefertigt werden soll, in diesem Fall von der Diskette :D1:, d.h. der BS1M-Diskette, die sich im Laufwerk 1 befindet. Durch diese Angabe wird dem Programm DIR eine weitere Information, nämlich der Parameter Laufwerknummer , übergeben.

Parameter werden von den Zuweisungen oder direkt vom Programmnamen, falls keine Zuweisungen nötig sind, durch das $-Zeichen getrennt.

1.2.3 Langfristige Zuweisungen von Geräten bzw. Dateien zu logischen Einheiten

Mit dem Programm ASSIGN (s. 2.3) ist es möglich, eine logische Einheit langfristig, d.h. über mehrere Programme hinweg, einem Gerät bzw. einer Datei zuzuweisen.

Die Zuweisung wird mit dem Programm RELEAS aufgehoben.
Zuweisungen für CI, RP und CO werden durch RELEAS nicht aufgehoben, sondern müssen explizit durch eine neue ASSIGN-Angabe geändert werden.

Beispiel: ASSIGN CO=:LP: (4)

Fortan werden die Ausgaben, die bisher auf dem Bildschirm erschienen sind, auf dem Drucker ausgegeben.

Man kann die Zuweisung (4) in den Kommandos folgendermaßen nutzen: Schreibt man etwa statt Beispiel 3

DIR$:D2: (5)

erhält man die Datei-Liste auf den Drucker, da vom Programm DIR SO=CO standardmäßig angenommen wird, wenn SO im Aufruf nicht zugewiesen wurde und durch (4) die Ausgaben vom Bildschirm auf den Drucker umgeleitet werden. Um die Ausgabe wieder auf :CO: umzuschalten, muß

ASSIGN CO=:CO: (6)

angegeben werden.

Man unterscheide daher zwischen der logischen Einheit CO und der physikalischen Einheit :CO:. :CO: bedeutet immer Bildschirmausgabe, nicht jedoch CO.

1.2.4 Disketten-Wechsel am Beginn eines Dienstprogrammes

Falls ein Programm auf der einen, die Daten aber auf einer anderen Diskette stehen, jedoch nur ein Disketten-Laufwerk vorhanden ist, darf das Programm, nachdem es aufgerufen wurde, nicht gleich starten, da zuvor die Diskette gewechselt werden muß. Dies geschieht folgendermaßen:

<Programmname>/CR (jetzt Disketten-Wechsel)

<Gerätezuweisungen>$<Parameter>CR

Beispiel: DIR/CR (7)
*SO=:LP:$:DO:CR

'*' ist das Quittungszeichen des Systems, mit dem es weitere Eingaben anfordert.

Mit diesem Kommando wird das Inhaltsverzeichnis einer BS1M formatierten Diskette gedruckt, auf der das Programm DIR nicht vorhanden ist. Zunächst wird DIR von einer Diskette geladen, dann wird sie durch die andere (ohne DIR) ersetzt.

1.2.5 Datei-Kombinationen

Zwei Arten von Kombinationen sind bei der Geräte- bzw. Dateizuweisung möglich:

Echo — z.B. CI=(:CI:/XYZ)

Bedeutung — Eingabe aus der ersten Datei (hier :CI:, also Tastatur) mit gleichzeitiger Ausgabe in die zweite (hier: Datei XYZ auf der BS1M-Diskette im Laufwerk 0).

Doppel — z.B. SO=(ABC.X,:CO:)

Bedeutung für Ausgabedateien — Gleichzeitige Ausgabe in beide Dateien (hier:in die Datei ABC.X auf der BS1M-Diskette in Laufwerk 0 bzw. auf der Festplatte und auf den Bildschirm, d.h. man sieht auf dem Bildschirm, was in die Datei ABC.X geschrieben wird).

Bedeutung für Eingabedateien — Verkettung beider Dateien, d.h. nach Abarbeitung der ersten Datei wird aus der zweiten eingelesen. Schachtelung ist erlaubt.
Beispiel:

```
SI=(DAT1,(DAT2,DAT3))
```

Hier wird zunächst aus Datei DAT1, dann aus DAT2 und anschließend aus DAT3 - jeweils auf der BS1M-formatierten Diskette im Laufwerk 0 bzw. auf Festplatte gelesen.

2 Dienstprogramme

Auf der BS1MP-System-Diskette werden eine Reihe von Dienstprogrammen mitgeliefert, die zur Initialisierung von Disketten, zum Kopieren von Dateien und Datenträgern, zum Einrichten und Löschen von Dateien usw. dienen.

In der nach Anwendungen gruppierten Liste der Dienstprogramme ist jeweils das Kapitel angegeben, in dem das Programm genauer beschrieben wird. Diese Beschreibungen sind in alphabetischer Reihenfolge der Programmnamen auf den folgenden Seiten zu finden.

a) Initialisierungsprogramme

FORMAT	Initialisierung und Formatierung von Disketten bzw. Festplatte in unterschiedlichen Formaten (2.13)

b) Kopierprogramme

DCOPY	Kopieren einer ganzen Diskette (2.8)
COPY	Kopieren einer oder mehrerer Dateien (2.7)
CONFIG	Kopieren einer System-Diskette auf/von Festplatte (2.6)
BACKUP	Sichern des Inhalts der Festplatte (2.5)

c) Programme zur Dateipflege

DIR	Auflistung des Inhaltsverzeichnisses (2.10)
REV	Auflistung des Ausgabestandes von Programmen (2.20)
ALLOC	Platzreservierung für eine Datei auf einer ECMA 54 ("IBM"-) formatierten Diskette (2.1)
ATTRIB	Änderung von Dateiattributen (2.4)
RENAME	Umbenennung einer Datei (2.18)
DELETE	Löschen von Dateien (2.9)
RESCUE	Retten einer gelöschten Datei (2.19)
DIRPAC	Packen des Inhaltsverzeichnisses einer Diskette (2.11)
ANALYZ	Prüfen einer Diskette bzw. Datei auf Fehler (2.2)

REMAP Rekonstruktion des Belegungsverzeichnisses einer Diskette (2.17) oder der Festplatte

SORT Sortieren von sequentiellen Dateien (2.21)

d) weitere Dienstprogramme

ASSIGN Ständige Zuweisung einer Datei/eines Gerätes zu einer logischen Einheit (2.3)

RELEAS Freigabe einer vorher mit ASSIGN zugewiesenen Datei (2.16)

EXEC Abarbeitung einer gespeicherten Kommandofolge (2.12)

HELP Auskunft über Systemschalter und Konfiguration (2.14)

SYS Setzen von Systemschaltern (2.23)

KONV Änderung ASCII - EBCDIC Konvertierungstabelle(2.15)

SPOOL Drucken von Listdateien als Hintergrundprozeß (2.22)

2.1 ALLOC

Funktion: Mit ALLOC kann auf einer ECMA 54-("IBM"-) formatierten Diskette ein Bereich für eine Datei reserviert werden.

Aufruf: ALLOC$:Jn:<Dateiname>

n ist die Laufwerknummer.

<Dateiname> darf aus 1 bis 8 Großbuchstaben oder Ziffern bestehen und darf auf der Diskette noch nicht vorhanden sein.

Falls der Aufruf fehlerfrei war, erscheint auf dem Bildschirm eine Liste, die vom Anwender auszufüllen ist.

```
FILENAME: Dateiname
REPLACE/NEW:
RECORD LENGTH:
BOE:
EOE:

REMAINDER DEFAULT (Y/N):

BYPASS:
ACCESS CHAR:
PROTECT:
CREATION DATE:
EXPIRATION DATE:
VERIFY:
EOD:
```

FILENAME: Enthält Dateiname aus dem Aufruf.

REPLACE/NEW: "R" bedeutet, daß eine bereits existierende Datei ersetzt wird, "N",daß die Datei neu angelegt wird. Im ersten Fall muß der Dateiname der Datei, die überschrieben werden soll, dem "R" - mit oder ohne Zwischenraum - folgen. Der Name wird nicht in Hochkomma gesetzt. Die physikalische Einheit wird nicht angegeben. Wird der Name weggelassen, wird DATA angenommen.

RECORD LENGTH (Satzlänge): Dreistellige Angabe der (logischen) Datensatzlänge. Bei "N" (s. vorige Parameter) obligatorisch. Bei fehlender Angabe (nur möglich bei "R") wird die Satzlänge der alten Datei übernommen.

BOE: Beginn des Dateibereichs. Spur- und Sektornummer (dezimal) des ersten Satzes der Datei. Obligatorische Angabe bei NEW, sonst (REPLACE und keine Angabe) werden die Werte der alten Datei übernommen. Die beiden zweistelligen Eingaben müssen mit einer 0 getrennt werden. Zu den Einzelheiten des ECMA-54-Formats siehe Anhang D.

EOE: Ende des Dateibereiches. Spur und Sektornummer (dezimal) des letzten für diese Datei reservierten Sektors. Format und Eingabemöglichkeiten wie bei BOE.

REMAINDER DEFAULT (Y/N): Der restliche Teil der Parameterliste braucht nicht unbedingt ausgefüllt zu werden. Antwort für diesen Fall: "Y" oder CR. Wenn einer der folgenden Parameter nicht weggelassen werden darf, muß mit "N" geantwortet werden.

BYPASS: Bypass-Indikator. Zeigt an, daß die Datei beim Kopieren oder bei Datentransfer übergangen werden soll. Eingabe von "B" für Bypass (überspringen); SP für "nicht überspringen". Bei Fehlen wird der alte Parameter übernommen.

Zur Beachtung: Das BS1MP berücksichtigt BYPASS nicht.

ACCESS CHAR: Zugriffsschutzfeld. SP bedeutet, daß die Datei nicht geschützt ist. Jedes andere Zeichen bedeutet Zugriffsschutz. Bei Fehlen wird der alte Parameter übernommen.

Zur Beachtung: Das BS1MP prüft das Feld nicht.

PROTECT: Schreibschutz. "P" bedeutet Schreibschutz, SP kein Schreibschutz, d.h. die Datei kann gelesen und beschrieben werden.

Zur Beachtung: Das BS1MP prüft das Feld nicht.

CREATION DATE: Erstellungsdatum der Datei. Format: JJMMTT (J= Jahr, M=Monat, T=Tag). Eingabe [SP] : Schreibdatum nicht signifikant. Bei fehlender Angabe wird der alte Wert übernommen.

EXPIRATION DATE: Freigabedatum. Format wie Schreibdatum. Ab dem angegebenen Datum kann die Datei gelöscht werden.

Zur Beachtung: Das BS1MP berücksichtigt das Freigabedatum nicht.

VERIFY: Verify/Copy-Indicator. Das Feld muß [SP], "V" oder "C" enthalten.

Zur Beachtung: Das BS1MP prüft das Feld nicht.

EOD: Ende des beschriebenen Teils des Dateibereiches, erster freier Sektor. Format wie BOE oder EOE. Bei Fehlen und REPLACE wird der alte Wert übernommen, bei Fehlen und NEW wird EOD=BOE angenommen.

Bei der Eingabe kann zur Fehlerkorrektur innerhalb einer Zeile die Taste ← benutzt werden. Bei Betätigung der ↖ -Taste beginnt die Parametereingabe von neuem.

Nach Angabe dieser Werte wird die Datei - sofern keine Fehler auftreten - angelegt. Anschließend kann die nächste Datei (auf dem gleichen Laufwerk) angelegt oder das Programm beendet werden. Bei der Anforderung des Namens mit

FILENAME:

darf kein Laufwerk angegeben werden.

Verzeichnis der Meldungen

<Dateiname> ESTABLISHED	normales Programmende
FILENAME ERROR	unzulässiger Dateiname im Programmaufruf
UNRECOGNIZED LABEL ID	Diskrepanz zwischen dem Formatkennzeichner im Aufruf und der Datenträger-Kennsatzinformation auf der Diskette.
<Dateiname> ALREADY EXISTS	bei NEW wurde eine bereits existierender Dateiname angegeben.

<Dateiname> NOT FOUND	der angegebene Dateiname wurde nicht gefunden
DIRECTORY PACKED	kein Platz mehr für weitere Dateinamen im Inhaltsverzeichnis der Diskette
IMPROPER RECORD LENGTH	unzulässige Satzlänge
DEFAULT PARAMETER NOT ALLOWED	Es muß ein Parameter angegeben werden
IMPROPER PARAMETER LIST	unzulässige Parametereingaben
BOE ≥ 73/26 NOT ALLOWED	Dateibeginn muß vor Spur 73, Sektor 26 liegen
EOE ≥ 73/26 NOT ALLOWED	Dateiende muß vor Spur 73, Sektor 26 liegen
BOE ≥ EOE NOT ALLOWED	Dateibeginn muß vor Dateiende liegen
EOD ≤ BOE NOT ALLOWED	Datenende muß hinter dem Dateibeginn liegen
EOD > EOE+1 NOT ALLOWED	Datenende darf nicht größer als Dateiende + 1 sein
IMPROPER TRACK VALUE	Unzulässige Spurangabe
IMPROPER SECTOR VALUE	Unzulässige Sektorangabe

2.2 ANALYZ (noch nicht verfügbar)

Funktion: Das Programm prüft, ob Disketten, Dateien oder Programme schadhaft sind.

Aufruf: ANALYZ$<Parameter>

oder: ANALYZ SI=<Eingabedatei>,SO=<Ausgabedatei>$
<Parameter>

<Parameter> können sein Kommandoart, Ausgabeart, Quell-laufwerk, Ziellaufwerk, Anzahl der Leseoperationen, Dateiname und Druckformat. Eingabedatei ist nur für die Kommandos PD und FC nötig. Ausgabedatei bestimmt die Ausgabedatei für den Diagnosebericht. Standardausgabedatei ist :CO:.

Kommandos:
- DD - Device Diagnostic (physikalische Prüfung der Diskette bzw. Festplatte)
- FD - File Diagnostic (Dateiprüfung in bezug auf Doppelbelegung)
- PD - Program Diagnostic (Prüfung des Ladeformates eines Programmes)
- VC - Volume Compare (blockweiser Vergleich zweier Disketten)
- FC - File Compare (byteweiser Vergleich zweier Dateien)

Zur Zeit stehen nur die Kommandos DD und PD zur Verfügung. Nicht alle Kommandos benötigen alle Parameter.

Ausgabeart:
- E - Error (nur Fehleranzeige)
- S - Status (Statusanzeige jedes Sektors bzw. Segments)
- D - Dump (Ausgabe des Inhalts jedes Sektors bzw. Segments in Verbindung mit E oder S)

Weitere Parameter:

PWnnn	Papierbreite (Standard 110)
PLnnn	Papierlänge (Standard 48)
:Xn:	Dateibezeichner und Laufwerksnummer (Standard ist :D0: bzw. :H0:)
RCnnn	Anzahl der Lesewiederholungen (Standard ist 1)
Dateiname	nach BS1MP Konventionen

1. Kommando DD

Dieses Kommando überpfüft alle Sektoren einer Diskette bzw. Festplatte auf Beschädigung. Tritt bei einer Diskette ein Fehler auf, darf sie nicht mehr verwendet werden.

Beispiele:
```
ANALYZ SO=:LP:$DD,:D1:,RC10,S,PW132,PL80
ANALYZ$DD,E
ANALYZ SO=Dump$DD,S,D
```

2. Kommando PD

Dieses Kommando überprüft das Ladeformat und das korrekte Ende einer einzelnen ladbaren Datei. Dabei wird die Länge, die Startadresse und bei Parameter D der Inhalt jedes einzelnen Segments ausgegeben, sowie die Ladeadresse des Programms.

Beispiele:
```
ANALYZ SI=PROGR,SO=:LP:$PD,S,D,PW80,PL66
ANALYZ SI=:D1:PROGR$PD,S
```

2.3 ASSIGN

Funktion: ASSIGN macht eine Zuweisung von Dateien bzw. Geräten zu logischen Einheiten, die für mehr als ein Kommando gültig bleibt.

Üblicherweise wird z.B. eine Datei beim Programmstart eröffnet und bei Programmende geschlossen. Wurde ASSIGN benutzt, dann bleibt die Datei eröffnet, bis ein RELEAS-Kommando gegeben wird.

Aufruf: ASSIGN <Zuweisungen>

Mehrere Zuweisungen werden durch Kommas getrennt.

Beispiel: ASSIGN SO=:LP:

Programme können jetzt ohne vorherige Zuweisung von SO gestartet werden.

Zuweisungen von CI und CO können nur mit ASSIGN explizit zurückgesetzt werden.

2.4 ATTRIB

Funktion: Änderung von Dateiattributen auf INTEL- oder BS1M-formatierten Disketten bzw. Festplatte.

Folgende Dateiattribute existieren:

S - System
F - Format
W - Schreibschutz
I - Unsichtbarkeit
X - Direktzugriffsdatei (RFM)
D - Direktdatei (DVS)
M - (reserviert)
N - Neu auf Festplatte

Die Attribute haben folgende Wirkungen:

S-Attribut — steht bei System- oder Dienstprogrammen. Diese werden durch das Programm FORMAT bei der Einrichtung einer System-Diskette automatisch kopiert.

F-Attribut — steht bei Dateien,auf die normalerweise nur das Betriebssystem zugreift, z.B. Belegungsliste, Inhaltsverzeichnis, Fehlermeldungs-Datei.

Das Attribut wird nur auf INTEL-formatierten Disketten und Festplatte verwendet.

W-Attribut — schützt eine Datei vor Löschung (DELETE) oder Umbenennung (RENAME).

I-Attribut — bewirkt, daß eine Datei nicht in einem mittels DIR erzeugten Inhaltsverzeichnis erscheint.

X-Attribut — steht bei Direktzugriffsdateien und bewirkt u.a., daß diese Dateien nicht mit COPY kopiert werden können. Die Übertragung auf andere Disketten geschieht mit dem Programm REORG (s. Kapitel 7).

D-Attribut — wird bei Direktdateien des DVS verwendet werden.

N-Attribut — kennzeichnet auf der Festplatte seit der letzten Datensicherung neu geschriebene Dateien.

Mit ATTRIB können nur die Attribute S, W und I geändert werden.

Aufruf: ATTRIB$ <Dateiname>, <Attributliste>

<Attributliste> ist eine Folge von Attributen (nur S, W oder I) mit Vorzeichen + oder -, getrennt durch Kommas.

Beispiel: ATTRIB$:F1:FIL.X,+I,-W

setzt für die Datei FIL.X auf der INTEL-Diskette 1 das I-Attribut und löscht das W-Attribut.

Bemerkung: Im Aufruf muß <Dateiname> in der gleichen Weise (groß oder klein) geschrieben sein wie auf der Diskette.

2.5 BACKUP

Dieses Programm ist noch in Vorbereitung und wird in einer späteren Ausgabe zur Verfügung stehen.
Es dient zur Datensicherung der Festplatte.

2.6 CONFIG

Dieses Programm ist noch in Vorbereitung und wird in einer späteren Ausgabe zur Verfügung stehen.
Es dient zum Einrichten des BS1MP auf Festplatte.

2.7 COPY

Funktion: COPY dient zur Übertragung von Daten von einer Datei in eine andere Datei. Die Dateien können sich auf dem gleichen oder auf unterschiedlichen Disketten befinden, die auch von unterschiedlichem Format sein können.

Darüber hinaus können Daten auch auf Drucker oder Bildschirm ausgegeben werden.

RFM-Dateien werden mit dem Programm REORG kopiert.

Folgende Arten des COPY-Aufrufes gibt es:

- Datei-Kopie
- Dateiverkettung
- Kopie mit einem Laufwerk
- Mehr-Dateien-Kopie

Die verschiedenen Varianten werden im folgenden getrennt beschrieben.

1. Datei-Kopie

Diese Variante des COPY-Aufrufs wird zur Kopie einer einzelnen Datei in eine andere Datei (bzw. Drucker oder Bildschirm) benutzt.

Aufruf: COPY SI=<Eingabe>,SO=<Ausgabe>$<Infoliste>

<Eingabe> ist die Quell- , <Ausgabe> die Zieldatei. <Infoliste> ist eine Liste, die folgende Parameter - getrennt durch Kommas - enthalten kann:

A - Kopieren mit Attributen (nur bei Dateien)

H - Ausgabe hexadezimal

Nn - Mehrfache Kopie (nur auf Drucker). n ist die Anzahl der gewünschten Kopien.

L - Ausgabe mit Zeilennummerierung (nur auf Drucker).

P - Ausgabe mit Seitennummerierung (nur auf Drucker).

PWnn - Papierbreite. nn ist die Anzahl der Zeichen pro Zeile, Standard ist 128, (nur auf Drucker).

PLnn - Papierlänge. nn ist die Anzahl der Zeilen pro Seite, Standard ist 48, (nur auf Drucker).

U - unformatierte Ausgabe (nur auf Drucker).

start - Nummer der ersten Zeile der Quelldatei, die kopiert werden soll.

stop - Nummer der letzten Zeile der Quelldatei, die kopiert werden soll.

H und L können nicht gleichzeitig angegeben werden, da H bereits eine Nummerierung der Zeilen durchführt.

Beispiele:

1) COPY SI=A1,SO=A2$A,98,1004

Datei A1 auf der Diskette :D0: wird auf eine neue Datei A2 auf der gleichen Diskette kopiert. Attribute werden übertragen (A). Die Datei wird von Zeilennummer 98 bis (inklusive) Zeilennummer 1004 kopiert.

2) COPY SI=A1,SO=:LP:$P,N5,PW80,PL72

Datei A1 wird mit 4 Kopien (also fünfmal) (N5) auf den Drucker ausgegeben.

Pro Seite werden 72 (PL72) Zeilen gedruckt, jede Zeile enthält maximal 80 Zeichen (PW80). Die Seiten werden auf dem Ausdruck durchnummeriert (P).

Hinweis: Kopiert man in eine Ausgabedatei, deren Name bereits existiert, erscheint die Frage:

Dateiname EXISTS, DELETE (Y/N)

Darf die bereits bestehende Datei gelöscht werden, kann man jetzt ein "Y" eingeben, andernfalls muß der Kopiervorgang mit "N" beendet werden.

Wenn der A-Schalter (s. SYS) gesetzt ist, wird die bereits existierende Ausgabedatei automatisch ohne Abfrage gelöscht.

2. Dateiverkettung

Diese Aufrufvariante wird benutzt, um mehrere Dateien in eine Ausgabedatei zu kopieren.

Aufruf: COPY AI=<Dateiliste>,SO=<Ausgabe>$<Infoliste>

Dateiliste ist eine Datei, welche die Namen der Eingabedateien enthält, die hintereinander in die Ausgabedatei kopiert werden sollen. Sie kann mit dem Editor EDIT erstellt werden. In einer Zeile darf jeweils nur eine Datei stehen, z.B.:

```
Datei.1
Datei.2
.
.
Datei.10
```

In der <Infoliste> können wie in der Aufrufvariante 1 die Parameter H, P, L, PL oder PW - getrennt durch Kommas - stehen. Existiert die Ausgabedatei bereits, so wird - wie unter 1 beschrieben - gefragt, ob sie gelöscht werden kann.

3. Kopie mit einem Laufwerk

Diese Aufrufvariante wird benutzt, wenn eine Datei auf eine andere Diskette kopiert werden soll, jedoch nur ein Laufwerk zur Verfügung steht.

Haben Quell- und Ziel-Diskette unterschiedliche Formate, so ist diese Art des Kopierens nur mit der Generiervariante BS1MP/01 bzw. BS1MP/11 (s.6.1) möglich.

Aufruf: COPY SO=<Ausgabe>$<Eingabe>

Beim Aufruf muß die Ziel-Diskette im Laufwerk sein.

Hinweis: Befindet sich das Programm COPY nicht auf der Ziel-Diskette, so kann es zunächst mit

```
COPY/ |CR|
*
```

von einer Programm-Diskette geladen werden. Das System fordert mit '*' neue Eingaben an. Anschließend kann die Ziel-Diskette in das Laufwerk eingelegt und die restlichen Zuweisungen und Parameter eingegeben werden.

Das COPY-Programm gibt während des Kopiervorgangs jedesmal, wenn ein Disketten-Wechsel nötig ist, eine der beiden folgenden Meldungen aus:

```
INSERT INPUT VOLUME ,
```

wenn die Quell-Diskette eingelegt werden muß und

```
INSERT OUTPUT VOLUME ,
```

wenn die Ziel-Diskette verlangt wird.

4. Mehr-Dateien-Kopie

Diese Aufruf-Variante wird benutzt, um mehrere oder alle Dateien einer Diskette auf eine andere Diskette zu kopieren.

Aufruf: COPY$<QUELLE>,<ZIEL>,<Infoliste>

<QUELLE> ist die Quell-, <ZIEL> die Ziel-Diskette. <Infoliste> kann die folgenden Parameter enthalten:

A - Übertragen von Attributen

R - vor jeder Dateikopie wird der Anwender zunächst gefragt, ob er eine Kopie der Datei wünscht. Gibt er "N" ein, wird die Datei übergangen.

S - Auch Dateien mit dem S-Attribut werden kopiert.

DA - Kommen auf Quell- und Ziel-Diskette Dateien mit gleichen Namen vor, so werden die auf der Ziel-Diskette jeweils vor dem Kopieren gelöscht.

DR - Wie DA, jedoch wird vor der Löschung vom Anwender eine Bestätigung erwartet. Kommt sie nicht (Eingabe "N"), so wird diese Datei übergangen.

Die Parameter DA und DR schließen sich gegenseitig aus.
Bei Parameter R wird DR automatisch angenommen. Bereits existierende Dateien werden nicht übertragen, es sei denn, der A-Schalter (s. SYS) ist gesetzt. Dann wird wie bei DA verfahren.

Nicht funktionell mit Copy

Beispiele für die verschiedenen Aufrufvarianten:

1) Man kann sich den Inhalt einer Datei folgendermaßen auf den Bildschirm ausgeben lassen:

```
COPY SI=PROG.SRC
```

Es wird immer ein voller Bildschirmausschnitt der Datei ausgegeben. Nach Eingabe eines beliebigen Zeichens außer "Q" werden die nächsten 24 Zeilen angezeigt. Eingabe von "Q" bricht das Programm ab.
Wird in einem COPY-Aufruf die Angabe von SO weggelassen, so wird SO=:CO: standardmäßig angenommen.

2) Möchte man eine Datei in hexadezimaler Form auf dem Bildschirm ausgeben, so gibt man

```
COPY SI=PROG.HEX$H
```

ein.

Rechts neben der hexadezimalen Darstellung werden die Zeichen in ihrer ASCII-Darstellung angezeigt (bzw. mit einem Punkt, wenn der Hex-Wert kleiner 20H oder größer 7FH ist).

3) Möchte man die Ausgabe auf dem Drucker haben, gibt man ein:

```
COPY SI=PROG.SRC,SO=:LP:
```

Werden keine Angaben zum Format des Ausdrucks gemacht (PW, PL), wird mit folgenden Standardwerten gedruckt:

```
Papierbreite = 110 Zeichen/Zeile
Papierlänge  =  48  Zeilen/Seite
```

4) Möchte man alle Dateien ohne S- Attribut von der INTEL-Diskette im Laufwerk 1 auf die BS1M-Diskette in Laufwerk 0 kopieren, gibt man ein:

```
COPY$:F1:,:D0:
```

5) Man möchte mehrere Dateien von einer Diskette auf eine andere Diskette kopieren. Das Kommando dafür lautet

```
COPY$:F0:,:D1:,R
```

Es kopiert Datei für Datei, fragt vorher jedoch jeweils, ob kopiert werden soll.

Folgender Dialog-Ausschnitt ist dann möglich:

```
.
.
DATA1      COPY ? (Y/N): Y
DATA1      COPIED
DATA2      COPY ? (Y/N): N
DATA3      COPY ? (Y/N): Y
DATA3      COPIED
DATA4      COPY ? (Y/N): Y
DATA4      EXISTS, DELETE ? (Y/N): Y
DATA4      COPIED
.
.
OPERATION COMPLETED. MORE ? (Y/N): N
```

Statt eines 'N' kann auch ein Leerzeichen eingegeben werden.
Der Vorgang kann durch Eingabe von 'Q' abgebrochen werden.

Bei COPY können dieselben Fehlermeldungen auftreten wie
bei RENAME.

2.8 DCOPY

Funktion: DCOPY kopiert eine vollständige Diskette unabhängig von ihrem Inhalt. Die Diskette darf keine CRC-Fehler enthalten.

Aufruf: DCOPY$<Quell-Laufwerk>,<Ziel-Laufwerk>

Ohne Angabe von Quell- und Ziel-Laufwerk wird die Diskette im Laufwerk 0 auf die im Laufwerk 1 kopiert.

Bevor der eigentliche Kopiervorgang nach Aufruf des Programms beginnt, wird noch einmal gefragt:

```
COPY FROM <QUELLE> TO <ZIEL> (Y/N):
```

Wird "Y" eingegeben, beginnt der Programmablauf, Eingabe eines anderen Zeichens beendet das Programm.

Am Ende des Kopiervorgangs erscheint die Meldung

```
EQUAL - more ?
```

Gibt man "Y" an, so kann erneut kopiert werden. Dadurch wird das Kopieren mehrerer Disketten hintereinander erleichtert.

Eingabe eines anderen Zeichens als "Y" bewirkt die Beendigung des Programms.

Folgende Paarungen von Quell- und Ziel-Disketten sind möglich:

```
:Di:  --> :Dj:
:Fi:  --> :Fj:
:Ji:  --> :Jj:
```

Steht nur ein Laufwerk für den Kopiervorgang zur Verfügung, so wird beim Aufruf nur das Quell-Laufwerk angegeben, z.B.

```
DCOPY$:D0:
```

Von dem Programm wird der Anwender mehrfach aufgefordert, die am Kopiervorgang beteiligten Disketten auszutauschen, es befindet sich also abwechselnd die Quell-Diskette zum Lesen und die Ziel-Diskette zum Schreiben im Laufwerk.

Hinweis: Wenn der E-Schalter (s. SYS) gesetzt ist und DCOPY unter EXEC läuft, werden die möglichen Abfragen unterdrückt.

2.9 DELETE

Funktion: DELETE löscht ausgewählte Dateien oder Dateigruppen von einer Diskette bzw. Festplatte. Bei INTEL- oder BS1M-formatierten Disketten und bei Festplatte wird der Platz zur Wiederbelegung sofort wieder freigegeben.

Aufruf: DELETE$:Xn:<Dateinamen>

:Xn: ist die Laufwerksbezeichnung. Sie bleibt solange erhalten, bis bei einem Dateinamen eine andere Laufwerksbezeichnung angegeben wird. :Xn: kann entfallen, wenn das Systemlaufwerk gemeint ist.
<Dateinamen> ist eine durch Kommas getrennte Folge von Dateibezeichnern, wie sie in Dateizuweisungen verwendet werden. Wird anstelle eines alphanumerischen Zeichens im Dateinamen ein '*' angegeben, so wird diese Stelle des Dateinamens ignoriert. Dies bedeutet, daß mit '*' mehrere gleichartige Dateien gelöscht werden können. Das Kommando darf nicht länger als eine Zeile sein.

Beispiele: 1. DELETE$:D1:ABC,DEF,:H0:TEMP

löscht auf :D1: die Dateien ABC und DEF und auf :H0: die Datei TEMP.

2. DELETE$:F1:*.HEX

Auf der INTEL-formatierten Diskette werden alle Dateien, deren Dateinamenserweiterung "HEX" ist, gelöscht.

3. DELETE$:D0:ABC.*

Auf der BS1M-formatierten Diskette werden die Dateien "ABC" mit allen Dateinamenserweiterungen gelöscht. Ist eine dieser Dateien mit W- oder S-Attribut gegen Löschen geschützt, so wird die Datei nicht gelöscht: es erscheint die Meldung "PROTECTED". Wurde vor dem Aufruf der S-Schalter gesetzt (s. SYS), werden auch die Dateien mit S- und/oder W-Attribut gelöscht.

4. DELETE$*

Auf der Diskette im Systemlaufwerk werden a l l e ungeschützten Dateien ohne Warnung gelöscht. Um bei gesetztem S-Schalter das irrtümliche Löschen der gesamten Diskette zu verhindern, muß in diesem Fall vorher eine Sicherheitsabfrage beantwortet werden.

A c h t u n g : Unter EXEC und bei gesetztem E-Schalter (s. SYS) erfolgt keine Warnung.

Wenn der Dateibezeichner auf eine ECMA 54-("IBM"-)formatierte Diskette verweist, dann wird der Kennsatzsektor der Datei mit "DELETED DATA" gekennzeichnet. Eine automatische Freigabe des belegten Platzes erfolgt nicht, da dies beim ECMA 54-Format nicht vorgesehen ist.

Hinweis: Dateien, die durch das W- oder S-Attribut geschützt sind, können gelöscht werden, wenn das Attribut vorher zurückgesetzt oder der Systemschalter S gesetzt wird.

Verzeichnis der Meldungen:

<Dateiname> NOT FOUND:	Der angegebene Dateiname wurde nicht gefunden. Das Programm sucht nach dem nächsten Dateinamen.
<Dateiname> PROTECTED:	Die angegebene Datei ist durch Attribut geschützt und kann nicht gelöscht werden. Die Verarbeitung wird beim nächsten Dateinamen fortgesetzt.
<Dateiname> DELETED:	Löschung wurde durchgeführt; die Verarbeitung wird beim nächsten Dateinamen fortgesetzt.
IMPROPER PARAMETER LIST	Mindestens einer der Dateinamen entsprach nicht den Namenskonventionen des BS1MP.
CAUTION, SYSTEM-SWITCH (S) IS SET ! DO YOU REALLY WANT TO DELETE ALL FILES (Y/N)?	Da der S-Schalter gesetzt ist, werden auch mit W- und S-Attribut geschützte Dateien gelöscht, d. h. die ganze Diskette.

2.10 DIR

Funktion: DIR erzeugt eine Liste der Namen und Charakteristiken aller nicht gelöschten Dateien einer Diskette.

Aufruf: DIR SO=<Ausgabedatei>$<phys.Einheit>I

<phys. Einheit> ist in der bei den Dateizuweisungen beschriebenen Form anzugeben. (:Fi:, :Di:, :Hi: usw.). Bei Fehlen wird das Systemlaufwerk angenommen.

Ist der Parameter 'I' vorhanden, dann erscheinen auch Dateien mit dem "unsichtbar"(invisible) Attribut in der Liste.

Wenn im Aufruf keine Zuweisung zu SO gemacht wird, dann erscheint die Liste auf dem Bildschirm und zwar zunächst nur die ersten 24 Zeilen. Die Ausgabe der nächsten 24 Zeilen kann durch Drücken einer beliebigen Taste (nicht jedoch "Q") veranlaßt werden.

Drücken von "Q" bewirkt den Abbruch des Programmes und den sofortigen Rücksprung ins System.

Eine Liste für INTEL- und BS1M- formatierte Disketten bzw. Festplatte enthält:

o Name der Diskette

o für jede Datei folgende Eintragungen:
  - Dateiname
  - Anzahl Blöcke (=Sektoren) der Datei
  - Dateilänge in Bytes
  - Attribute der Datei (vergl. dazu Beschreibung für ATTRIB)

o Anzahl der belegten Sektoren der Diskette
o Gesamtzahl der im Belegungsverzeichnis registrierten Sektoren.

Hinweise: Sollten die beiden letzten Zahlen zueinander in keinem sinnvollen Zusammenhang stehen, kann ein früherer anormaler Abbruch eines Programmes die Ursache sein. Die Diskrepanz kann durch REMAP in den meisten Fällen beseitigt werden.

Liste für eine ECMA 54-("IBM"-) formatierte Diskette

o Name der Diskette

o für jede Datei die folgenden Einträge:
  - Dateiname
  - jeweils Spur/Sektor für Beginn und Ende des Dateibereichs und der letzten Information in der Datei.
  - Liste der Datei-Flags

Beispiel: DIR$:F1:

erzeugt auf CO das Inhaltsverzeichnis einer INTEL-formatierten Diskette im Laufwerk 1.

```
DIR SO=:LP:$:D1:
```

erzeugt auf dem Drucker das Inhaltsverzeichnis der BS1M-formatierten Diskette im Laufwerk 1.

2.11 DIRPAC

Nach dem Löschen einer Datei mit Hilfe des Programmes DELETE wird der freigewordene Platz im Inhaltsverzeichnis der Diskette bzw. Festplatte nicht wieder benutzt. Das bedeutet nach einer Anzahl von Löschungen, daß das Inhaltsverzeichnis voll belegt sein kann, jedoch zum Teil mit gelöschten Einträgen. Das Inhaltsverzeichnis einer BS1M-formatierten Diskette kann maximal 1024, das einer INTEL-formatierten Diskette maximal 196 Eintragungen enthalten. Das Programm DIRPAC packt das Inhaltsverzeichnis so zusammen, daß die gelöschten Einträge verschwinden und der freigewordene Platz wieder verwendet werden kann.

DIRPAC ist nicht für ECMA 54-("IBM"-) Disketten anwendbar.

Das Programm wird aufgerufen durch

```
DIRPAC$<phys. Einheit>
```

Die <phys. Einheit> wird in der Form :Xn: angegeben.

Wird $ und die <phys.Einheit> nicht angegeben, so wird das Systemlaufwerk angenommen.

Beispiel: DIRPAC$:F1:

packt das Inhaltsverzeichnis der INTEL-Diskette im Laufwerk 1.

2.12 EXEC

Funktion: Aufruf und Abarbeitung einer auf einer Diskette bzw. Festplatte gespeicherten Kommandofolge. Nach Dateiende werden Kommandos wieder über die Tastatur erwartet.

Aufruf: EXEC CI=<Eingabedatei>

Beispiel: EXEC CI=COMFOL.001

Die Eingabedatei (hier: COMFOL.001) kann mit dem Editor EDIT erstellt werden. Sie kann jedes Kommando und jeden Programmaufruf enthalten, der auch direkt unter Kontrolle des BS1MP eingegeben werden kann. Bei Eingaben, die von Dienst- oder Anwenderprogrammen nach ihrem Aufruf verlangt werden, gibt es unterschiedliche Verhaltensweisen. Zum Teil werden die Eingaben aus der Kommandodatei gelesen, gelegentlich werden sie jedoch von der Tastatur erwartet, außer der E-Schalter (s. SYS) ist gesetzt. Tritt bei Ausführung eines Kommandos ein Systemfehler auf, so wird die weitere Verarbeitung der Kommandodatei abgebrochen, es sei denn, der Systemschalter (s. SYS) ist auf C gesetzt. In diesem Fall wird mit dem nächsten Kommando der Folge fortgefahren. Bei Abbruch wird die Meldung

```
JOB ABORTED
```

ausgegeben.

Ein Schrägstrich nach einem Kommando (vor Zuweisungen und Parametern) stoppt die Ausführung, bis ein beliebiges Zeichen über Tastatur eingegeben wird, der Rest der Zeile wird überlesen. Diese Möglichkeit kann z.B. genutzt werden, um den Disketten-Austausch zwischen zwei Programm-Aufrufen durchzuführen.

Beispiel:
```
FORMAT/
$SYSTEM.NEU,L
DIR/
so=:LP:$i
```

Nach FORMAT bzw. DIR wird jeweils der Befehlsfluß unterbrochen.

SUBMIT-Funktion in EXEC zur Benutzung formaler Eingabedateien:

Die SUBMIT-Funktion ersetzt in einem Kommandostring die formalen Parameter durch aktuelle als Parameter spezifizierte Werte. In einer Kommandodatei sind formale Parameter gekennzeichnet durch die beiden Zeichen %n, wobei n eine Ziffer zwischen 0 und 9 ist. Sie können in der Kommandodatei an beliebiger Stelle stehen.

Aufruf: EXEC CI=<Komdat>$<p0>,<p1>,<p2>...,<p9>

<p0> bis <p9> sind die Definitionen für die formalen Parameter in der Kommandodatei.

Beim Aufruf von EXEC wird eine neue Datei EXEC.CSF angelegt. Diese Datei enthält die umgesetzte, absolute Kommandofolge. Die Datei wird nach Abarbeitung der Kommandofolge automatisch gelöscht.

Hinweis: Wenn schon vor Aufruf der SUBMIT-Funktion eine Datei mit Namen EXEC.CSF existiert, wird diese gelöscht.

Beispiel:
Um eine formale Eingabedatei zum Übersetzen und Binden eines Programms zu erstellen, schreibt man mit EDIT eine Kommandofolge wie diese:

```
DELETE$%0.REL
RASM SI=%0.SRC,SO=%0.REL,SL=%1
LINK SO=%0,SL=%1
*C=4000
%0.REL
/
```

Heißt das Programm, das übersetzt und gebunden werden soll, PROG.SRC, lautet der EXEC-Aufruf:

```
EXEC CI=FORM.KOM$PROG,:LP:
```

Nach der Umsetzung und dem Anlegen der temporären Kommandodatei EXEC.CSF, sieht die absolute Kommandodatei wie folgt aus:

```
DELETE$PROG.REL
RASM SI=PROG.SRC,SO=PROG.REL,SL=:LP:
LUNK SO=PROG,SL=:LP:
*C=4000
PROG.REL
/
```

Nach Ausführung der Kommandofolge wird die temporäre Datei EXEC.CSF gelöscht.

2.13 FORMAT

Funktion: Das Programm initialisiert und/oder formatiert Disketten wahlweise im INTEL-, BS1M- oder ECMA 54-Format oder eine Festplatte. Die Benutzung einer nicht initialisierten Diskette bzw. Festplatte ist ohne Behandlung mit FORMAT nicht möglich.

Aufruf:

1. FORMAT$<phys.Einheit><Datenträgername> ,{ S ,I / I ,S / leer leer },n

2. FORMAT$<phys.Einheit><Datenträgername> ,L

<phys.Einheit> hat die Form :Xn: (n = Laufwerksnummer).

<Datenträgername> besteht für :Hn:, :Dn: und :Fn: aus maximal sechs Buchstaben oder Zahlen, wahlweise gefolgt von einem Punkt und bis zu drei weiteren Buchstaben oder Zahlen. Das Format stimmt also mit denen für Dateinamen überein.

Für :Jn: besteht Datenträgername aus maximal 6 Großbuchstaben.

Format 1:

Die Parameter S und I haben die folgende Bedeutung:

I Die Diskette wird initialisiert und formatiert. Wird I weggelassen, so wird die Diskette bzw. Festplatte nur formatiert. Dies ist jedoch nur möglich, wenn sie zuvor bereits einmal initialisiert wurde. Läßt man I weg, so sind nur die physikalischen Einheiten :Hn:, :Fn: und :Dn: zugelassen. Wird <Datenträgername> weggelassen, wird der Datenträger nur initialisiert.

Nach der Formatierung enthält eine INTEL- oder BS1M-formatierte Diskette bzw. die Festplatte lediglich die Systemdateien (Belegungsverzeichnis, Inhaltsverzeichnis, Datei mit Systemfehlermeldungen, Datei mit Namen des Datenträgers), die im Fall des BS1M-Formats bzw. der Festplatte für den Anwender nicht zugänglich sind. Bei INTEL-Disketten beginnen die Namen dieser Dateien mit "ISIS." und besitzen ein F-Attribut.

Zu den einzelnen Disketten-Formaten siehe Anhänge B - D.

S Nach dem Formatieren werden von dem Systemlaufwerk alle Dateien mit einem S-Attribut herüberkopiert. Voraussetzung ist, daß beide Disketten das gleiche Format haben.

n n kann die Werte 1 und 2 annehmen. Wird n weggelassen, wird 1 angenommen. Die Zahl bestimmt die Nummerierungsreihenfolge der Diskette-Sektoren. Bei 1 ist sie einfach aufsteigend, d.h. logisch aufeinanderfolgende Sektoren stehen physikalisch hintereinander. Bei 2 ist die Reihenfolge 1,14,2,15.., d.h. logisch aufeinanderfolgende Sektoren stehen 13 Sektoren - eine halbe Umdrehung - auseinander.

Unterschiedliche Sektorierung führt zu unterschiedlichem Zeitverhalten, Zugriffe auf Direktzugriffsdateien sind bei Sektorsequenz 1 bis zu 30 % langsamer als bei der Sequenz 2. Im Normalfall sollte daher n=2 angegeben werden.

Format 2:

Dieser Aufruf dient nur zur Änderung des Datenträgernamens.

Hinweis: Wenn der E-Schalter (s. SYS) gesetzt ist und Format unter EXEC läuft, werden die entsprechenden Abfragen unterdrückt.

2.14 HELP

Funktion: Das Programm wird benutzt, um Informationen über den Status von Systemschaltern und die Systemkonfiguration zu erhalten.

Aufruf: HELP SO=<Ausgabe>

Wird keine Ausgabedatei (bzw. Drucker) angegeben, wird die Information auf dem Bildschirm ausgegeben.

Es wird die folgende Maske ausgegeben:

BS1MP ANLEITUNG, REV A.xx

1 - Systemkonfiguration

2 - Systemstatusmeldungen

Wählt man Punkt 1 an, so erhält man die folgenden Informationen:

- verfügbare Geräte-Handler
- Hinweis, ob der jeweilige Geräte-Handler ständig speicherresident ist oder jeweils nachgeladen werden muß (transient).
- Hinweis, ob der RFM (Modul für die Unterstützung der Direktzugriffsdateien) geladen ist
- Hinweis, ob die geladene Generierversion die SPOOL-Funktion unterstützt.

- Format des System-Laufwerks (H oder D oder F)

- System-Startadresse

- Anfangsadresse des Anwenderspeichers

Wählt man Punkt 2 an, werden folgende Informationen ausgegeben:

- Status der mit SYS veränderbaren Systemschalter (ausgegebener Text in Großbuchstaben):

A gesetzt:	AUTOMATISCHES LÖSCHEN
S gesetzt:	SCHREIBSCHUTZ AUFGEHOBEN
W gesetzt:	SCHREIBSCHUTZ
C gesetzt:	KEIN ABBRUCH UNTER EXEC
E gesetzt:	KEINE ABFRAGEN UNTER EXEC
Kein Schalter gesetzt:	SYSTEMSCHALTER GESCHLOSSEN

- Abbruchzeichen (mit SYS veränderbar)

- Untergrenze des Pufferbereiches

- Adresse des Systemstacks

- Bildschirmeinstellung (ROLL oder PAGE, Zeilenfortschaltung)

- Generierungs- und Ausgabestand des Betriebssystems

Das HELP-Menü kann mit "Q" wieder verlassen werden.

2.15 KONV

Funktion: Mit KONV kann die im Betriebssystem BS1MP enthaltene Konvertierungstabelle ASCII -- EBCDIC entsprechend geändert werden. Die Systemdiskette muß sich im Laufwerk 0 befinden.

Aufruf: KONV

Zuerst wird die ASCII → EBCDIC Konvertierungstabelle ausgegeben. Durch Eingabe eines ASCII-Wertes wird auf den entsprechenden EBCDIC-Wert positioniert, der dann durch Überschreiben geändert werden kann. Geänderte EBCDIC-Werte werden invertiert dargestellt. Positioniert man aus Versehen falsch, so kann mit CR ins Eingabefeld zurückgekehrt werden, eine Änderung erfolgt dann nicht.

Weitere mögliche Eingaben statt eines ASCII-Wertes:
H=Hardcopy, die entsprechende Tabelle wird ausgedruckt.
R=Rückkehr ins Betriebssystem.
W=Wechseln, d.h. Ausgabe der EBCDIC → ASCII Tabelle, oder wenn diese schon dargestellt ist, nochmalige Ausgabe der ASCII → EBCDIC Tabelle.

Die Bedienung zur Änderung der EBCDIC -> ASCII Tabelle ist sinngemäß die der Änderung der ASCII -> EBCDIC Tabelle.

Kehrt man in das Betriebssystem zurück (Eingabe von R), wird geprüft, ob man eine Änderung vorgenommen hat.
Wenn man keinen Wert geändert hat, wird die Meldung "KEINE ÄNDERUNG IN DER CODE-TABELLE" ausgegeben.
Hat man einen oder mehrere Werte geändert, erhält der ausgewählte "IBM"-Handler ISDHAN.zz die Revisionsnummer REV A.9x.

Fehlermeldungen:

"ECMA 54-HANDLER im LAUFWERK 0 NICHT VORHANDEN" tritt auf, wenn sich das BS1MP nicht im System-Laufwerk befindet.

"KEINE REV BZW. VERS ANGABE IM ERSTEN SEKTOR" tritt auf, wenn die Konvertierungstabelle in Betriebssystemen ohne Versions- oder Revisionsnummer geändert werden soll.

Desweiteren können Betriebssystemfehlermeldungen auftreten.

2.16 RELEAS

Funktion: RELEAS schließt Dateien, die durch ASSIGN (s.2.3) offengehalten wurden.

Aufruf: RELEAS$<Folge logischer Einheiten>

Die logischen Einheiten der Folge müssen durch Kommas getrennt werden.

Beispiel: RELEAS$SO,SL

Hinweis: RELEAS setzt CI und CO nicht zurück.

2.17 REMAP

Funktion: REMAP versucht, das zerstörte Belegungsverzeichnis einer INTEL- oder BS1M-formatierten Diskette oder einer Festplatte anhand ihres Inhaltsverzeichnisses zu rekonstruieren.

Aufruf: REMAP$<phys. Einheit>

oder REMAP$

Im zweiten Aufruf wird vom Programm das System-Laufwerk angenommen.

Der Erfolg der Rekonstruktion hängt vom Zustand des Inhaltsverzeichnisses ab. Wenn auch nur eine falsche Spur- oder Sektornummer gefunden wird, bricht das Programm ab.

2.18 RENAME

Funktion: RENAME ändert den Namen der im Aufruf angegebenen Datei.

Aufruf: RENAME$<phys. Einheit><Dateiname>,<neuer Dateiname>

<Dateiname>: Bezeichnung der umzubenennenden Datei

<neuer Dateiname>: Angabe im üblichen Format, aber ohne phys. Einheit

Beispiel: RENAME$:F1:ORANGE,APFEL

Die Datei ORANGE auf :F1: wird in APFEL umbenannt.

Verzeichnis der Meldungen

<Dateiname> NOT FOUND	Die angegebene Datei wurde nicht gefunden.
<neuer Dateiname> EXISTS, DELETE (Y/N)	Eine Datei mit dem neuen Dateinamen existiert bereits. Durch Eingabe von "Y" kann diese Datei gelöscht werden. Anschließend wird die Umbenennung durchgeführt. Man kann das Programm RENAME jedoch mit "N" auch vorzeitig beenden. Diese Meldung wird unterdrückt, wenn der A-Schalter (s. SYS) gesetzt ist.
<Dateiname> RENAMED	Die Umbenennung wurde fehlerlos durchgeführt.

<Dateiname> PROTECTED	Die Datei ist durch ein W- oder S-Attribut geschützt und kann nicht umbenannt werden.
IMPROPER UNIT SPECIFICATION	Der Bezeichner für die phys. Einheit ist falsch. Die Verarbeitung wurde abgebrochen.
IMPROPER PARAMETER LIST	Die Parameter-Liste ist fehlerhaft. Die Verarbeitung wird abgebrochen.

2.19 RESCUE

Funktion: Wenn eine Datei unabsichtlich oder fälschlicherweise gelöscht wurde, kann sie mit RESCUE unter bestimmten Voraussetzungen gerettet werden.

Aufruf: RESCUE$<phys.Einheit><Dateiname>

Voraussetzungen:

Bei einer INTEL- oder BS1M-formatierten Diskette:

- Seit der Löschung darf keine neue Datei erzeugt worden sein.

Bei einer ECMA 54-("IBM"-) formatierten Diskette:

- Seit der Löschung darf kein neuer Dateikennsatz vergeben worden sein
- Der durch die gelöschte Datei belegte Platz darf nicht infolge Dateiüberlappung beschrieben worden sein.

Beispiel: DELETE$:D1:XYZ

RESCUE$:D1:XYZ

Verzeichnis der Meldungen:

<Dateiname> RECLAIMED	Die Datei wurde gerettet. Alle von RESCUE durchgeführten Prüfungen verliefen fehlerfrei.
<Dateiname> NOT FOUND	Es konnte keine gelöschte Datei mit dem angegebenen Namen gefunden werden.
<Dateiname> LOST	Die Datei konnte nicht mehr gerettet werden (weil eine der obenstehenden Bedingungen nicht gegeben war).

2.20 REV

Allgemeine Hinweise

- REV gehört der Software-Klasse B an.
  REV ist nur unter BS1M und BS1MP ablauffähig.

- REV ist ein Wartungsprogramm und dient zur Ausgabe des Ausgabestandes (REVISION) der Systemprogramme.

- Folgende Aufrufe sind möglich:

```
REV  SO=<datei>$I,<:Xn:>                    (1)
     SO=:LP:

REV  SO=<datei>,SI=<:Xn:dateiname>$R         (2)
     SO=:LP:                          A
```

Zu (1):
- Beispiele:
```
REV SO=:LP:$I,:D1:
REV$:D1:
```

- Zusätzlich zu den DIR-Angaben werden die Ausgabestände (REV) auf SO ausgegeben.
  Wird keine Angabe gefunden, wird NONE ausgegeben.

  Das Programm sucht im 1. Sektor einer jeden Datei nach REV. Beginnt die Erweiterung der Datei mit R (r), S (s) oder D (d), werden 10 Sektoren auf diese Angabe abgesucht.
  RFM-Dateien werden übersprungen.

- Wird SO nicht explizit zugewiesen, wird :CO: für SO angenommen.

- Wird der Parameter I angegeben, werden auch die mit dem Attribut I versehenen Dateien auf SO angezeigt.

- Bei fehlender Laufwerksangabe :Xn: wird das System-Laufwerk angenommen.

Zu (2):
- Beispiel 1: `REV SO=:LP:,SI=:D1:DAT$`
  auf SO:
```
REVISION-/VERSION-Number of :D1:DAT
          No.:    REV A.01
```

Beispiel 2: `REV SI=:D1:AWMSV1$R`
auf CO:
```
REVISION-/VERSION-Number of :D1:AWMSV1
          No.:    REV 3.01R
```

Beispiel 3: `REV SI=AWPRGR$A`
auf CO:
```
REVISION-/VERSION-Number of :H0:AWPRGR
          No.:    REV 3.00
                  REV 3.01R
                  ---NONE---
```

- Die unter SI zugewiesene Datei wird untersucht.

- Ist kein Parameter angegeben, wird der erste Sektor auf den Ausgabestand abgesucht und auf SO ausgegeben.
  Wird SO explizit nicht ausgegeben, wird :CO: für SO angenommen.

- Bei Angabe des Parameters R wird die SI-Datei auf das erste Systemmodul untersucht, das zum Anwenderprogramm gebunden wurde (z.B. MSV1-Prozedur). Wird so ein Modul gefunden, wird der Ausgabestand unter der Form: X.YYR ausgegeben. R zeigt an, daß es sich um ein Systemmodul handelt, welches zum Anwenderprogramm gebunden wurde.

- Wird der Parameter A angegeben, wird die SI-Datei auf alle Ausgabestände untersucht.
  Alle gefundenen Ausgabestände werden auf SO ausgegeben und mit NONE abgeschlossen.

2.21 SORT

2.21.1 Leistungsbeschreibung

Das Programm SORT liest eine angegebene Eingabedatei und erzeugt daraus eine Ausgabedatei, in der die Sätze in der Reihenfoge des angegebenen Sortierbegriffes geordnet sind.

Der ein- oder mehrteilige Sortierbegriff muß Teil des Datensatzes sein.

SORT ist für sequentielle Dateien im INTEL-, im BS1M- und im ECMA 54- ("IBM"-)Format sowie für Festplatte verwendbar. IBM-Dateien werden umcodiert und stehen während des Sortiervorganges als 7-Bit-ASCII-Zeichen zur Verfügung.

Die maximale Länge des Sortierbegriffes beträgt 255 Bytes. Er kann aus bis zu zehn verschiedenen Teilen bestehen.
Jeder Teil des Sortierbegriffes kann aufsteigend oder absteigend sein.

Der SORT benötigt für temporäre Arbeitsdateien verfügbaren Raum auf einer Diskette im INTEL- oder BS1M-Format oder auf Festplatte.

Die maximale Größe der zu sortierenden Eingabedatei wird durch diesen verfügbaren Raum bestimmt.

2.21.2 Aufruf des SORT

Der SORT kann entweder direkt aufgerufen werden, wobei ihm alle Parameter mit dem Aufruf übergeben werden müssen, oder er kann nach Interpretierung der Parameter selbständig eine gebundene Version erzeugen. Diese wird unter ihrem eigenen Namen aufgerufen, wobei keine Parameter angegeben werden müssen.

Aufruf für Laden und Sortieren:

SORT$<Parameter>

Der Anwender kann sich EXEC-Dateien (s. 2.12) erzeugen, um sich mehrmaliges Eingeben des Aufrufs zu ersparen.

Aufruf für Laden und Binden:

entweder:

SORT SO=<Programmname>$<Parameter>

oder:

SORT$SAVE <Programmname> <Parameter>

Wird der Aufruf: 'SORT so=<Programmname>$<Parameter>' verwendet, ist es sinnvoll, für die Eingabedatei nicht 'SAVE' anzugeben, da sonst die Fehlermeldung:

ERROR when 'SAVE' found

ausgegeben wird, um eine falsche Interpretation der Parameter zu vermeiden.

2.21.3 Parameter des SORT

Die Parameter müssen in der angegebenen Reihenfolge eingegeben und durch mindestens einen Zwischenraum voneinander getrennt werden.

Falls nötig, können die Parameter in mehreren Zeilen eingegeben werden, die durch CR zu beenden sind. Weitere Trennzeichen sind überflüssig.

In der folgenden Parameterliste sind die Schlüsselwörter in Großbuchstaben geschrieben, aber nur, um sie hervorzuheben. Bei der Eingabe werden sie auch in Kleinbuchstaben akzeptiert.

Die in Klammern gesetzten Ausdrücke sind durch die entsprechenden aktuellen Dateinamen, Prozedurnamen usw. zu ersetzen.

Parameter:

I		Eingabedatei
II	TO	Ausgabedatei
III	USING	:Xn:
IV	PREPROC	Eingabeprozedur
V	POSTPROC	Ausgabeprozdur
VI		DISC
	SAME	I O W
		DISK
VII		Position, Länge A
	KEY	
		Position, Länge D
VIII		BIN Satzlänge
	REC	
		ASCII Satzlänge

Siehe hierzu die Beispiele ab Seite 13.

Beschreibung der Parameter

I Eingabedatei

Name der Datei, die sortiert werden soll. Der Dateiname muß nach den im BS1MP gültigen Regeln gebildet sein. Jede vom BS1MP unterstützte sequentielle Datei kann sortiert werden.

Liegt die Eingabedatei auf einer "IBM"-formatierten Diskette, muß die Zuweisung :Jx: (EBCDIC) lauten; als Satzbeschreibung muß ASCII angegeben werden.

II Ausgabedatei

Name der Datei, in welche die sortierten Datensätze ausgegeben werden. Jede vom BS1MP unterstützte Dateiform kann angegeben werden. Das Schlüsselwort TO ist obligatorisch.

Der Parameter kann auch weggelassen werden. In diesem Fall wird keine Ausgabedatei erzeugt, sondern der Benutzer muß sich in einer Ausgabeprozedur selbst darum kümmern.

III Zuweisung der Arbeitsdateien

Mit diesem Parameter wird der Datenträger für die temporären Arbeitsdaten angegeben.

Sie müssen auf einer vom BS1MP unterstützten ,formatierten Diskette liegen, wobei "n" die Nummer des Disketten-Laufwerks ist. Das Schlüsselwort USING ist wahlweise.

Die maximale Größe der zu sortierenden Datei hängt ab von dem auf der Arbeits-Diskette verfügbaren freien Raum. Vor jedem Datensatz in einer Arbeitsdatei wird ein Längenfeld von zwei Bytes eingefügt. Andererseits werden beim Sortieren von ASCII-Dateien Ketten von gleichen Zeichen gepackt. Es kann daher schwierig sein, abzuschätzen, wie groß der benötigte Arbeitsbereich sein muß. Meist wird er etwas größer sein als die Eingabedatei.

Die Arbeitsdateien haben alle den Namen SW.x, wobei "x" durch einen Buchstaben ersetzt wird, beginnend mit "A" für die erste Arbeitsdatei usw. in alphabetischer Reihenfolge.

Achtung: Alle auf der Arbeits-Diskette bereits existierenden Dateien mit solchen Namen gehen verloren!

Nach erfolgreichem Abschluß des Sortierens werden alle Arbeitsdateien gelöscht.

IV Eingabeprozedur

Die Eingabeprozedur ist ein Anwendermodul, das jeweils angesprungen wird, wenn ein Datensatz eingelesen wurde und zur Bearbeitung bereitsteht. Das Modul muß im RELOCATABLE-Format sein, wie es durch RASM erzeugt wird. Es darf nur aus einem Code-Segment bestehen, d.h. im Source-Code dürfen die Pseudoanweisungen ORG und DSEG nicht vorkommen.

Die Eingabeprozedur wird vor Sortierbeginn geladen. Für jeden zu sortierenden Satz wird sie einmal durchlaufen. Das erste Byte der Prozedur sollte auch ihr Einsprungpunkt sein.

Wenn die Routine angesprungen wird, enthält Doppelregister HL die Adresse des ersten Bytes des Datensatzes.

In der Eingabeprozedur kann mit dem Datensatz jede beliebige Verarbeitung vorgenommen werden, jedoch muß der Satz beim Aussprung aus der Routine am selben Speicherplatz stehen wie beim Einsprung. Unter bestimmten Einschränkungen kann auch die Satzlänge verändert werden (Näheres s. unter Abschn. VIII).

Nach dem Rücksprung dürfen die Inhalte sämtlicher Register verändert sein. Nur der Inhalt des A-Registers wird geprüft:

A = 0: Der Satz wird übersprungen.

A = 1: Der Satz wird sortiert und der nächste Satz wird eingelesen.

A > 1: Der Satz wird sortiert und die Eingabeprozedur wird wieder angesprungen.

Wenn die Eingabeprozedur nach dem Einlesen eines Satzes angesprungen wird, ist A = 0. Wird sie aber ohne Nachlesen sofort wieder angesprungen (d.h. A war vorher > 1, s.o.), dann bleibt der Wert in A unverändert.

V Ausgabeprozedur

Für sie gelten dieselben Auflagen wie für die Eingabeprozedur. Die Routine wird jedesmal angesprungen, wenn ein Datensatz zur endgültigen Ausgabe bereitsteht. HL enthält die Adresse des ersten Bytes des Satzes.

Nach dem Aussprung wird der Inhalt von A geprüft:

A = 0: Der Satz wird übersprungen (nicht ausgegeben).

A > 0: Der Satz wird ausgegeben.

Wurde keine Ausgabedatei zugewiesen (Parameter II, s.o.), dann ist der Inhalt von A gleichgültig.

VI SAME (gleiche physikalische Einheiten)

Damit wird angegeben, welche Dateien sich auf derselben Diskette befinden sollen.

Falls in den Parametern I, II und III andere phys. Einheiten zugewiesen wurden, aber SAVE angegeben wurde, dann wird angenommen, daß die SAME-Angabe falsch sei.

Die Schlüsselwörter DISC bzw. DISK sind wahlweise.
Bedeutung der Schlüsselbuchstaben:

I: Eingabe (INPUT)

O: Ausgabe (OUTPUT)

W: Arbeitsgerät (WORK UNIT)

Fehlt die SAME-Angabe und wurde für die Parameter I, II und III die gleiche Einheit zugewiesen, dann wird von SORT rechtzeitig ein Disketten-Wechsel angefordert, weil angenommen wird, daß die Dateien sich nicht auf der selben Diskette befinden.

VII Sortierbegriff

Der Sortierbegriff legt fest, in welcher Reihenfolge die Datensätze in der Ausgabedatei stehen sollen.

Der Sortierbegriff kann aus 1 bis 10 nicht unterbrochen Teilbereichen des Datensatzes ("Sortierfeldern") bestehen. Die Sortierfelder müssen in jedem Satz dieselbe Position (gerechnet vom Satzanfang) haben.

Jedes Sortierfeld wird durch drei Angaben festgelegt:

- die Position seines ersten Bytes, gerechnet vom Satzanfang (erstes Byte des Satzes = 1),
- seine Länge (Anzahl Bytes), und
- die Sortierfolge für dieses Feld.

Die Sortierfolge wird durch die Buchstaben "A" (aufsteigend) und "D" (absteigend) bestimmt. Fehlt die Angabe, dann wird "A" angenommen.

Die Gesamtlänge des Sortierbegriffes (Summe aller Sortierfelder) darf nicht größer als 255 Bytes sein.

VIII Satzbeschreibung

Die Eingabesätze können als ASCII oder BIN (aer) definiert werden, gefolgt von der Satzlänge in Bytes. Das Schlüsselwort REC ist wahlweise.

Bei IBM-Dateien muß ASCII angegeben werden, auch wenn sie auf der Diskette in EBCDIC stehen.

Als BIN definierte Sätze müssen fixe Länge haben. Der Satzinhalt ist beliebig.

Wurde ASCII angegeben, dann dürfen die Sätze von variabler Länge sein, müssen aber mit CR (mit oder ohne LF) abgeschlossen sein. In der Ausgabedatei wird hinter jedes CR ein LF geschrieben. Die angegebene Satzlänge sollte gleich der größten vorkommenden Satzlänge + 1 (für das CR) sein. Werden längere Sätze gefunden, dann werden diese auf die angegebene Länge - 1 gekürzt. Sätze, die zu kurz sind, um den Sortierbegriff zu enthalten, werden intern mit Null erweitert und vor der endgültigen Ausgabe wieder auf die ursprüngliche Länge gekürzt.
Die Sätze sollten 7-Bit-ASCII-Zeichen enthalten, da das achte Bit beim Einlesen abgeschnitten und intern (zum Packen) verwendet wird.

In ASCII-Dateien darf die Satzlänge in der Ein- oder Ausgabeprozedur durch Verschieben des CR verändert werden, jedoch nur innerhalb der definierten Satzlänge. Falls in der Eingabeprozedur die Satzlänge auf die geschilderte Weise verkürzt wird, dann wird empfohlen, den Rest des Eingabepuffers auf Null zu setzen. Dies hat keinen Einfluß auf das Sortierergebnis, verkürzt jedoch die Sortierzeit und spart etwas Platz in den Arbeitsdateien.

2.21.4 Beispiele

I Eine Datei namens AFILE mit Binärsätzen von 16 Bytes Länge soll sortiert und in die Datei BFILE ausgegeben werden. Der Schlüssel ist in den ersten 5 Bytes jedes Satzes. Alle Dateien befinden sich auf der Diskette im Laufwerk :D0:.

```
SORT$AFILE TO BFILE USING :D0: SAME I O W KEY 1 5 BIN
16
```

AFILE:

```
01 04 7F 00 00 00 04 05 01 00 00 22 05 06 01 01
00 01 2C 00 04 04 01 03 00 11 05 06 01 04 00 01
2E 00 04 00 01 06 00 06 0G 1A BE C2 05 06 01 0B
00 01 1A 00 04 0F 01 0D 00 FE 0D 3E 00 C8 23 13
AF 05 C8 C3 05 06 01 18 00 01 08 00 04 04 01 1A
00 2A 05 06 01 1B 00 01 2C 00 04 04 01 1D 00 11
05 06 01 1E 00 01 2E 00 04 0A 01 20 00 7E 12 13
23 FE 0D C2 05 06 01 27 00 01 20 00 04 00 01 29
00 3E 01 C9 00 00 FF 00
```

BFILE:

```
00 01 1A 00 04 0E 01 0D 00 FE 0D 3E 00 C8 23 13
00 01 2C 00 04 04 01 03 00 11 05 06 01 04 00 01
00 2A 05 06 01 1B 00 01 2C 00 04 04 01 1D 00 11
00 3E 01 C9 00 00 FF 00 00 00 00 00 00 00 00 00
01 04 7F 00 00 00 04 04 01 00 00 22 05 06 01 01
05 06 01 1E 00 01 2E 00 04 0A 01 20 00 7E 12 13
23 FE 0D C2 05 06 01 27 00 01 20 00 04 08 01 29
2E 00 04 08 01 06 00 06 0F 1A BE C2 05 06 01 0B
AF 05 C8 C3 06 06 01 18 00 01 08 00 04 04 01 1A
```

II Eine ASCII-Datei CFILE mit variabler Satzlänge (max. 80 Bytes) auf der Diskette :D0: soll sortiert und auf Drucker ausgegeben werden. Die Arbeitsdateien sollen auf :H0: liegen. Sortierbegriff sind die Kontonummer (Bytes 16-21), aufsteigend, und für gleiche Kontonummern nach Datum, absteigend (Bytes 28-33, Form: TTMMJJ).

```
SORT$:D0:CFILE TO :LP: USING :H0:
KEY 16 6 32 2 D 30 2 D 28 2 D ASCII 81
```

CFILE:

```
A.MÜLLER        200432    020179
D.BAUER         700211    110578
J.KRAUS         800199    110578
F.HUBER         510455    070378
R.MAYER         908867    121177
L.SCHMIDT       800623    041078
E.KARL          112788    131277
D.BAUER         700211    070677
R.MAYER         908867    240878
F.HUBER         510455    030378
D.BAUER         700211    160579
J.KRAUS         800199    261177
D.BAUER         700211    300377
```

Druckausgabe:

```
E.KARL          112788    131277
A.MÜLLER        200432    020179
F.HUBER         510455    070378
F.HUBER         510455    030378
D.BAUER         700211    160579
D.BAUER         700211    110578
D.BAUER         700211    070677
D.BAUER         700211    300377
J.KRAUS         800199    110578
J.KRAUS         800199    261177
L.SCHMIDT       800623    041078
R.MAYER         908867    240878
R.MAYER         908867    121177
```

III Eine ASCII-Datei DFILE mit variabler Satzlänge (max. 80 Bytes) auf :D0: soll sortiert und in EFILE (ebenfalls auf :D0:, jedoch auf einer anderen Diskette) ausgegeben werden. Die Arbeitsdateien sollen auf :D1: sein. In der Ausgabedatei sollen alle Wörter der Eingabedatei in alphabetischer Reihenfolge vorhanden sein, jedoch jedes Wort nur einmal. Ein Wort sei definiert als ununterbrochene Zeichenkette von Buchstaben oder Ziffern, max. 15 Zeichen lang.

Das Problem wird gelöst mit Hilfe einer Eingabeprozedur, welche aus jedem Wort in einem Eingabesatz einen Satz macht und dieser an den SORT übergibt, und einer Ausgabeprozedur, welche die mehrfach vorkommenden Wörter eliminiert. Die Assemblerlistings dieser beiden Prozeduren befinden sich auf Seite ...ff (unter den Namen PRE und POST).

```
SORT$DFILE TO EFILE :D1: PREPROC PRE POSTPROC POST
KEY 1 15 ASCII 81
```

DFILE:

EINE ASCII-DATEI DFILE MIT VARIABLER SATZLÄNGE (MAX: 80 BYTES) AUF :D0: SOLL SORTIERT UND IN EFILE (EBENFALLS AUF :D0:) AUSGEGEBEN WERDEN. DIE ARBEITSDATEIEN SOLLEN AUF :D1: SEIN. IN DER AUSGABEDATEI SOLLEN ALLE WÖRTER DER EINGABEDATEI IN ALPHABETISCHER REIHENFOLGE VORHANDEN SEIN, JEDOCH JEDES WORT NUR EINMAL. EIN WORT SEI DEFINIERT ALS UNUNTERBROCHENE ZEICHENKETTE VON BUCHSTABEN ODER ZIFFERN, MAX. 15 ZEICHEN LANG.

EFILE:

15
80
ALLE
ALPHABETISCHER
ALS
ARBEITSDATEIEN
ASCII
AUF
AUSGABEDATEI
AUSGEGEBEN
BUCHSTABEN
BYTES
DATEI
DEFINIERT
DER
DFILE
DIE
EBENFALLS
EFILE
EIN
EINE
EINGABEDATEI
EINMAL
F0
F1
IN
JEDES
JEDOCH
LANG
MAX
MIT
NUR
ODER
REIHENFOLGE
SATZLÄNGE
SEI
SEIN
SOLL
SOLLEN
SORTIERT
UND
UNUNTERBROCHENE
VARIABLER
VON
VORHANDEN
WERDEN

```
    WORT
    WÖRTER
    ZEICHEN
    ZEICHENKETTE
    ZIFFERN


$TITLE P R E


:
: Presort routine
: Take each word as a separate entry
:
cr              equ     0dh
:
pre:            shld    recad
                ora     a
                jnz     pre2
                lxi     d,reci      ; record first time

prei:           mov     a,m
                stax    d
                inx     d
                inx     h
                cpi     cr
                jnz     prei

pre2:           lxi     h,reci      ; search leading edge
pre2a:          mov     a,m
                cpi     cr
                jz      pre10
                call    char
                jnc     pre3
                inx     h
                jmp     pre2a

pre3:           xchg                ; leading edge found,
                                    ; transfer back
                lhld    recad

pre4:           ldax    d
                call    char
                jc      pre5
                mov     m,a
                inx     d
                inx     h
                jmp     pre4
```

```
pre5:       push    d
            mvi     m,cr
            inx     h
            push    h
            lhld    recad
            lxi     d,81
            dad     d
            xchg
            pop     h

pre6:       mov     a,h
            cmp     d
            jnz     pre7
            mov     a,l
            cmp     e
            jz      pre8

pre7:       mvi     m,0
            inx     h
            jmp     pre6

pre8:       pop     h
            xchg                ; move down in internal
                                ; buffer
            lxi     h,reci

pre9:       ldax    d
            mov     m,a
            inx     h
            inx     d
            cpi     cr
            jnz     pre9
            mvi     a,2
            ret

pre10:      xra     a
            ret

char:       cpi     30h
            rc
            cpi     3ah
            jc      char1
            cpi     40h
            rc
            cpi     60h
            cmc
            rnc
            sui     20h

char1:      ora     a
            ret

;
recad:      dw      0
reci        ds      81
            end
```

```
$TITLE   P O S T
;
; suppress equal words
;
cr          equ         0dh
;
post:       shld        recad
            lxi         d,intrec
            mvi         b,15
post1:      ldax        d
            cmp         m
            jnz         post2
            cpi         cr
            mvi         a,0
            rz
            inx         h
            inx         d
            xra         a
            dcr         b
            rz
            jmp         post1
post2:      lhld        recad
            lxi         d,intrec
post3:      mov         a,m
            stax        d
            inx         d
            inx         h
            cpi         cr
            jnz         post3
            mvi         a,1
            ret
;
recad:      dw          0
intrec:     ds          81
;
            end
```

2.21.5 Zeitbedarf

Der Zeitbedarf für das Sortieren wird sehr stark durch die notwendigen E/A-Operationen bestimmt. Beim Sortieren kleiner Dateien (unter 30 KB) werden die Sätze nur einmal gelesen bzw. geschrieben, bei größeren jedoch zweimal.

2.21.6 Fehlermeldungen

Error when <Parameter> found (Parameter-Fehler)

BS-Error xxx when write <Dateiname> *)

BS-Error xxx when read <Dateiname> *)

Output file exists. Change medium or type another name (Ausgabedatei existiert bereits. Diskette wechseln oder anderen Namen eingeben)

Bad filename typed. Type again. (Falscher Dateiname wurde eingegeben. Gib neuen Dateinamen ein)

Bad Data in <Dateiname> (In der Datei wurde ein Fehler festgestellt)

*) xxx ist eine Systemfehlermeldung des BS1MP

22 SPOOL

Funktion: Drucken von Listdateien als Hintergrundprozeß quasiparallel zu einem anderen Programmablauf.

Aufruf: SPOOL <Zuweisung>$<Kommando>,<Parameter>

<Zuweisung>: SI= Eingabedatei für SPOOL-Warteschlange
SO= SPOOL-Ausgabegerät (Standard=:LP:)

<Kommando>: eines der nachfolgend beschreibenen SPOOL-Kommandos

<Parameter>: zum Kommando gehörige Parameter

Achtung: Das Dienstprogramm SPOOL ist nur ablauffähig unter einer BS1MP-Konfiguration (s. 6.1), die diese Möglichkeit enthält.

Der SPOOL muß vor der ersten Benutzung aktiviert und nach Beendigung deaktiviert werden. Der Bediener sollte folgendermaßen vorgehen:

1. SPOOL aktivieren
2. SPOOL-Kommandos geben oder dem SPOOL-Ausgabegerät Datei zur Verfügung stellen
3. SPOOL deaktivieren

Folgende Kommandos stehen zur Verfügung:

B SPOOL-Beginn. Aktivierung des SPOOL.

Bsp.: SPOOL$B

Das Kommando kann eine Zuweisung enthalten, wenn bereits eine SPOOL-Warteschlangen-Datei vorhanden ist. Das ist jedoch nicht der Normalfall.

Bsp.: SPOOL SI=QUEUE$B

Das Standardausgabegerät des SPOOL ist :LP:. Wenn :XM: gewünscht wird, ist eine Zuweisung für SO notwendig.

Bsp.: SPOOL SO=:XM:$B

Die Anzahl der Einträge in der Druckerwarteschlange ist standardmäßig 16. Dieser Wert kann vermindert (bis auf min. 1) und vergrößert (bis auf max. 64) werden.

Bsp.: SPOOL$B,30

aktiviert den SPOOL und legt 30 Puffer für Einträge in der SPOOL-Warteschlange an.

Nach Aktivierung des SPOOL-Systems, erfolgt die Ausgabe auf das SPOOL-Ausgabegerät über das System-Laufwerk.

Bsp.: TXT SI=TEXT,SO=:LP:

druckt die Datei TEXT nicht direkt aus, sondern über eine Warteschlangen-Datei auf dem System-Laufwerk, die von SPOOL erzeugt wird.

S SPOOL stoppen. Die Ausgabe über SPOOL wird verhindert

Bsp.: SPOOL$S

Der SPOOL kann mit dem Kommando R wieder fortgesetzt werden.

R SPOOL fortsetzen nach dem S-Kommando. Es gibt drei verschiedene Möglichkeiten zum Wiederstarten des SPOOL:

- C Wiederstart an der aktuellen Position der aktuellen Datei der Warteschlange.

  Bsp.: SPOOL$R,C

- B Wiederstart am Beginn der aktuellen Datei der Warteschlange.

  Bsp.: SPOOL$R,B

- N Wiederstart bei der nächstfolgenden Datei der Warteschlange.

  Bsp.: SPOOL$R,N

Wenn kein zweiter Parameter angegeben wird, wird C angenommen.

E SPOOL Ende. Deaktivierung des SPOOL.
Dies kann nur erreicht werden, wenn keine Einträge in der Warteschlange mehr vorhanden sind.

Bsp.: SPOOL$E

Das SPOOL-System muß wieder aktiviert werden mit dem Kommando B, bevor ein neues SPOOL-Kommando gegeben werden kann.

Q SPOOL abbrechen.
Das Kommando Q hat dieselbe Wirkung wie das Kommando E, außer daß es sofort abbricht.

I Einfügen von Dateien vor dem ersten Eintrag in der Warteschlange.

Bsp.: SPOOL$I,LIST1,LIST2,LIST3

fügt die Dateien LIST1, LIST2, LIST3 in die Druckerwarteschlange ein. Danach lautet die Reihenfolge der ersten drei Einträge: LIST3, LIST2, LIST1.

A Anfügen von Dateien an die SPOOL-Warteschlange als letzten Eintrag.

Bsp.: SPOOL$A,DAT1,DAT2,DAT3

fügt die Dateien DAT1, DAT2, DAT3 am Ende der Warteschlange hinzu. Danach lautet die Reihenfolge der letzten drei Einträge: DAT1, DAT2, DAT3.

D Löschen von Dateien in der Warteschlange.

Bsp.: SPOOL$D,DRUCK1,DRUCK2,DRUCK3

löscht die Dateien DRUCK1, DRUCK2, DRUCK3 aus der Drucker-Warteschlange. Wenn die Dateien durch SPOOL erzeugt wurden, werden sie in der Warteschlange und auf dem System-Laufwerk gelöscht. Handelt es sich um vom Benutzer mit dem Kommando A oder I in die Warteschlange eingefügte Dateien, wird nur der Eintrag in der Warteschlange gelöscht, aber nicht die Datei selbst.

L Liste des Inhalts der Drucker-Warteschlange und SPOOL-Status ausgeben.

Bsp.: SPOOL$L

bringt Informationen wie diese auf den Bildschirm:

```
NO  P     TYPE          NAME  EXT      LENGTH

01  *     AUTO FILE     SPOOL 007      22134
02        AUTO FILE     SPOOL 008      15799
03        USER FILE     LIST1           2143
04        USER FILE     LIST2           2143
05        USER FILE     LIST3           2143

11 Buffers in free queue
SPOOL STATE : RUNNING
```

Die Parameter bedeuten:

NO Position der Datei innerhalb der Druck-Warteschlange

P Ein '*' in dieser Spalte zeigt an, daß diese Datei gerade ausgedruckt wird.

TYPE Zeigt den Dateityp.

AUTO FILE ist eine von SPOOL erzeugte Datei. Dies geschieht bei der Zuweisung auf das SPOOL-Ausgabegerät. Eine AUTO FILE wird nach Druckende automatisch gelöscht. Sie kann auch mit dem Kommando D gelöscht werden.

USER FILE ist eine vom Benutzer erzeugte Datei, die mit Hilfe des Kommandos A oder I in die Druck-Warteschlange eingefügt wurde. Eine USER FILE wird nach Druckende nicht gelöscht, sondern nur aus der Warteschlange entfernt. Das gilt auch für das D-Kommando.

NAME ist der Name der Datei, die mit SPOOL gedruckt werden soll.

EXT ist die Erweiterung des Namens.

LENGTH zeigt die Dateilänge in BYTES an.

2.23 SYS

Funktion: Mit dem Programm können mehrere Systemschalter gesetzt und gelöscht werden, die im Zusammenhang mit verschiedenen Dienstprogrammen unterschiedliche Funktionen haben.

Aufruf: SYS$<Systemschalterliste> (zum Setzen)

oder SYS (zum Rücksetzen aller Systemschalter mit Ausnahme des Abbruchzeichens (Quit-Zeichen))

Es gibt die folgenden Systemschalter:

S Ist dieser Schalter gesetzt, so können mit S- oder W-Attribut versehene Dateien gelöscht oder umbenannt werden, ohne daß das Attribut rückgesetzt werden muß.

Die Systemschalter W (s.u.) und S können nicht gleichzeitig gesetzt werden, da ein Schalter den anderen überschreibt.

C (Continue-Schalter). Tritt in der Abarbeitung einer EXEC-Kommandofolge ein Betriebssystemfehler auf, so wird die Folge mit dem nächsten Kommando fortgesetzt, nicht - wie sonst - abgebrochen.

W Dateien können weder umbenannt noch gelöscht werden, solange dieser Schalter gesetzt ist. W überschreibt den S-Schalter.

A Bereits existierende Dateien werden automatisch gelöscht, wenn eine neue Datei mit dem gleichen Namen angelegt wird.

E Der EXEC-Parameter-Modus wird eingeschaltet, um Abfragen zu unterdrücken wie z. B.

OPERATION COMPLETED; MORE? (Y/N):

Der Schalter ist nur unter EXEC aktiv.

Q Mit diesem Schalter kann ein beliebiges Abbruchzeichen definiert oder rückgesetzt werden.

Ein Abbruchzeichen bewirkt, daß bei seiner Eingabe über die Tastatur das laufende Programm abgebrochen wird und das System in den Monitor verzweigt. Mit G CR kann das Programm fortgesetzt, mit GFD00 kann in das BS1MP verzweigt werden.

Schalter werden im Aufruf mit vorgesetztem Plus oder einfach dem kennzeichnenden Buchstaben gesetzt und durch eine vorgesetztes Minuszeichen gelöscht.

Der aktuelle Status der Systemschalter und des Abbruch-Zeichens kann mit dem Dienstprogramm HELP überprüft werden.

Beispiele:

1) SYS$+W,-C,+Q=01

- o Dateilöschungen und umbenennungen sind nicht möglich (+W)
- o Continue-Schalter ist ausgeschaltet (-C)
- o das Zeichen Control A (01H) ist als Abbruchzeichen definiert.

2) SYS$S,C,-Q

- o Löschschutz für S- und W-Dateien ist aufgehoben (S)
- o Continue-Schalter ist gesetzt (C)
- o das Abbruchzeichen ist zurückgesetzt, d. h. Programme können nicht mehr durch Eingabe eines Zeichens abgebrochen werden (-Q).

2.24 T X T

Funktion:

Mit dem Programm TXT ist es möglich, Textdateien, die mit dem Editor EDIT erstellt worden sind, formatiert auszudrucken. Die Format-Steuerzeichen, mit denen der Ausdruck kontrolliert wird, steuern unter anderem die Seitennummerierung, Tabulatorpositionen und Silbentrennung.

Die Steuerzeichen werden während der Erfassung des Textes in Form bestimmter Abkürzungen und Sonderzeichen eingegeben und bleiben im Text stehen.

Aufruf:

```
TXT SI=<Textdatei>,SL=<LISTDATEI>
```

oder

```
TXT SI=<Textdatei>,SO=<LISTDATEI>
```

oder

```
TXT AI=<Jobdatei>,SO=<LISTDATEI>
```

Im letzten Fall können in der Jobdatei mehrere Namen von Textdateien gespeichert werden, die eine nach der anderen verarbeitet wird.

Steuerzeichen:

Steuerzeichen sind Bestandteile des Textes. Es gibt davon drei Arten.

A Kontrollzeichen des Typs 1

Kontrollzeichen des Typs 1 gelten entweder für die ganze Datei oder solange, wie die Parameter nicht neu versorgt werden. Sie werden in getrennten Zeilen in den Text eingefügt. Jede derartige Zeile muß mit ":" beginnen. Die eigentliche Kontrollinformation besteht aus Schlüsselworten von zwei Zeichen Länge, gefolgt von einem oder mehreren Parametern.

Die Parameter (Zahlen) müssen von den Schlüsselworten durch Komma oder Zwischenraum getrennt werden.

Folgende Schlüsselwort/Parameter-Kombinationen sind möglich:

PL Papierlänge

Ein Parameter, nämlich die Anzahl der Zeilen, die auf das vorhandene Papier gedruckt werden kann. Pro Zoll sind sechs Zeilen möglich.

LM Linker Rand

Ein Parameter, der die erste Druckposition links in einer Zeile bestimmt.

RM Rechter Rand

Ein Parameter, der die letzte Druckposition rechts innerhalb einer Zeile bestimmt.

TS Tabulator-Stopp

Die Anzahl der Tabulator-Stopps, deren Position angegeben wird, ist nur durch die Länge der Eingabezeile beschränkt.

RA Rechtsbündige Anpassung

Keine Parameter. Alle Zeilen werden rechtsbündig geschrieben.

NA Aufhebung der rechtsbündigen Anpassung

Keine Parameter

TP Tabulator-Stopp für Absatz

Zwei Parameter, gültig für die nächsten Absätze im Text. Die erste Zahl gibt an, an welcher Schreibstelle die erste Zeile eines neuen Absatzes beginnen soll, die zweite bezeichnet den Beginn aller folgenden Zeilen des Absatzes.

SL Leerzeilen

Ein Parameter, der angibt, wieviele Leerzeilen vor jedem Absatz eingefügt werden sollen.

FL Erste Zeile

Ein Parameter, der die erste Textzeile auf einer neuen Seite bestimmt. Standardzuweisung: Zeile 5.

LL Letzte Zeile

Ein Parameter, der die letzte Druckzeile auf einer Seite bestimmt. Standardzuweisung: Zeile 68.

PN Seitennummerierung

Drei oder vier Parameter:
1.) Nummer der Zeile
2.) Position innerhalb der Zeile
3.) Format:
    1: Ohne Nullenunterdrückung
    2: Seitennummer innerhalb von Strichen
       (z.B. - 5 -)
4.) Beginn der Seitennummerierung (wahlweise).
    Fehlt die Angabe, dann wird auf den internen
    Seitenzähler zugegriffen. Die Seitennummerierung beginnt, sobald es nach der Eingabe des
    Kommandos sinnvoll ist.

HT Seitenüberschrift

Drei Parameter:
1.) Zeile
2.) Position innerhalb der Zeile
3.) Gewünschter Text. Da der Rest der Zeile als Text
    interpretiert wird, sollte der dritte Parameter
    entweder am Ende der Kommandozeile oder allein
    in der nächsten Zeile stehen.

B Kontrollzeichen des Typs 2

Kontrollzeichen des Typs 2 beziehen sich nur auf den folgenden Paragraphen. Paragraphen beginnen mit einer Zeile, die an der ersten Stelle einen Punkt und sonst keine Zeichen enthält.

RM RA NA TS TP SL wie bei Typ 1

UF Unformatiert

Der folgende Paragraph wird so gedruckt, wie er abgespeichert ist. Relevante Kontrollzeichen sind nur PL und LM.

NP Neue Seite

Die nächste Druckzeile ist die erste auf einer neuen Seite.

ST Sprung auf eine bestimmte Zeile

Ein Parameter, nämlich die Nummer der Zeile, in welcher der Druck fortgesetzt werden soll. Seitenlänge beachten.

C Kontrollzeichen des Typs 3

Kontrollzeichen des Typs 3 werden direkt in den Text eingefügt:

! Ausrufungszeichen: Neue Zeile

Das nächste zu druckende Zeichen erscheint in einer neuen Zeile.

> Größer-Zeichen: Tabulator-Sprung

Das nächste Zeichen wird beim nächsten Tabulator-Stopp gedruckt.

- Bindestrich: Trennungszeichen

Erscheint der Bindestrich innerhalb eines Wortes, so bezeichnet er eine Stelle, wo das Wort getrennt werden kann. Passt das Wort noch in die Zeile, dann wird der Strich nicht mitgedruckt.

Sollen die drei beschriebenen Zeichen keine Kontrollfunktion haben, so muß ihnen im Text ein Schrägstrich folgen.

3 Sprachsystem Assembler

Beschreibung der Assemblersprache

Eine Beschreibung liegt in der Siemens-Druckschrift "Programmiersprache SAB 8080/8085" vor (Bestell-Nr. B 2236), jedoch müssen folgende Einschränkungen beachtet werden:

Die Verwendung und Programmierung von Makros ist nicht möglich. Die Pseudobefehle MACRO und ENDMACRO sind verboten.

Als Pseudoinstruktionen sind ORG, CSEG, DSEG, EXTRN, PUBLIC, EQU und END erlaubt.

Zusätzlich sind folgende Instruktionen verwendbar:

$TITLE (Überschriften im Protokoll)

$E(ject) (Blattvorschub)

$PL xx (Seitenlänge; xx = Anzahl Druckzeilen/Seite)

$PW xx (Papierbreite; xx = Anz. Druckzeichen/Zeile)

Arithmetische und logische Ausdrücke im Operandenfeld sind nicht erlaubt (Ausnahme: + und -)

Zu dem Sprachsystem gehören Programme zum Erfassen, Korrigieren, Übersetzen, Binden und Testen von Assembler-Programmen.

Alle diese Programme werden auf der BS1MP-System-Diskette mitgeliefert.

Im einzelnen gibt es folgende Hilfsmittel zur Programmierung:

RASM	Verschiebbarer Assembler (3.4)
LINK	Programm zum Binden verschiebbarer Assembler-Module (3.2)
MEMDMP	Ausgabe eines bestimmten Speicherbereiches als ablauffähiges Programm (3.3)
DEBUG	Testen eines Programmes (3.1)
XREF	Ausgabe einer Querverweisliste eines Assemblerprogrammes (3.5)
EDIT	Editor zur Erfassung und Änderung von Programmen. (4.)

3.1 DEBUG

Funktion: DEBUG lädt ein Programm und macht die erforderlichen Dateizuweisungen, startet jedoch nicht den Ablauf. Stattdessen geht die Kontrolle an den Monitor über.

Nun können die Kommandos des Monitors (5.1) angewendet werden, um das Programm unter dessen Kontrolle ablaufen zu lassen.

Aufruf: DEBUG$<Programmname> <Zuweisungen>

Beispiel: DEBUG$:D1:NEWPROG SI=DAT,SO=:CO:

lädt NEWPROG von :D1:, weist die Dateien zu und springt in den Monitor. Die Zuweisungen sind nur erforderlich, wenn sie nicht vom Anwenderprogramm selbst vorgenommen werden.

3.2 LINK

Funktion: LINK bindet durch RASM (s. 3.4) erzeugte Module. Die Ausgabe kann, gesteuert durch Parameter, sowohl absolut (ladbar) als auch relativierbar (relocatable) erfolgen.
Das relativierbar ausgegebene Modul kann wiederum mit anderen Moduln gebunden werden. Die Ausgabe kann dabei sowohl ladbar als auch relativierbar erfolgen.

Für die Testphase der Programmierung kann über einen weiteren Parameter auch dann ein ladefähiges Programm erzeugt werden, wenn beim Binden noch unbefriedigte EXTRN-Adressen auftreten. In diesem Fall wird als Wert 0000 eingesetzt (im Regelfall darf eine solche Programmstelle nicht angesprochen werden).

LINK unterstützt die Benutzung der Overlay-Technik, siehe hierzu Parameter X und Beispiel 1 in Anhang F.

Aufruf: LINK SO=<Ausgabe-Datei>,SL=<Auflistung>,
AI=<Kommando-Datei>$<Binde-Parameter>

<Ausgabe-Datei>	Binderausgabe (relativ oder absolut)
<Auflistung>	Datei oder Gerät für das Binderprotokoll
<Kommando-Datei>	Vorbereitete Datei mit LINK-Kommandos

<Binde-Parameter>	Durch Kommas getrennte LINK-Kommandos. LINK-Kommandos werden zunächst der Kommando-Datei entnommen. Ist die Kommandoeingabe nach Erreichen des Dateiendes noch nicht durch '/' abgeschlossen, werden die Kommandos hinter dem $-Zeichen ausgewertet. Ist die Eingabe auch dann noch nicht mit '/' abgeschlossen, werden weitere Kommandos über die Tastatur erwartet.

Es gibt folgende LINK-Kommandos:

*A	Abbruch des Bindevorganges bei Auftreten von unbefriedigten EXTRN-Adressen
*I	Ausgabe in ladbarer Form trotz unbefriedigter EXTRN-Adressen
*E	Ein- und Ausschaltung eines Schalters für die Ausgabeunterdrückung von PUBLICs bei relocativen Binden
*N	Ausgabe von maximal drei Moduln im LINK-Listing
*R	Ausgabe in relativierbarer Form Wird *R fortgelassen, wird der Code in ladbarem Format ausgegeben.
*X	Ein- und Ausschalten eines Schalters für die Ausgabeunterdrückung von Moduln. Module, die zwischen zwei X-Parametern gebunden werden, werden zwar bei der Auflösung von PUBLIC- und EXTRNAL-Bezügen berücksichtigt, sie werden jedoch nicht ausgegeben. Siehe hierzu das Beispiel in Kapitel 9.2 und Anhang F.
*C= nnnn	Ladeadresse für das Codesegment
*D= nnnn	Ladeadresse für das Datensegment Falls das Datensegment dem Codesegment unmittelbar folgen soll, dann kann entweder *D weggelassen oder *D = *C angegeben werden.

*S= nnnn	Startadresse des Programmes Zur Beachtung: Wenn die zu bindenden Module Namen im END-Statement haben, dann wird das erste gefundene als Startadresse benutzt. Eine explizit angegebene Startadresse (*S = ...) hat jedoch Vorrang.
{:Hn: / :Dn: / :Fn:} <Dateiname>	Namen der zu bindenden Module
/	bezeichnet das Ende der Parametereingabe
*Q	Abbruch des Bindevorganges und Rücksprung ins BS1MP (Ausgabedatei wird geschlossen)
*P= nnn	(dezimale) Anzahl der Zeilen pro Seite bei der Auflistung (mindestens 24).
nnnn	bei C/D/S : 4-stellige hexadezimale Adressen. " H" (wie im Assembler notwendig) darf nicht verwendet werden.
*Z	Datum und Uhrzeit werden aus dem Monitorbereich ins LINK-Listing übernommen, wenn SL zugewiesen wurde
*Z= Datum Zeit	Die angegebene Zeichenfolge wird in das LINK-Listing übernommen, wenn SL zugewiesen wurde

Hinweise:

01 Bei Angabe von Dateien gelten für die Namensbildung die üblichen Vorschriften des BS1MP. Dateien auf ECMA 54-Disketten sind nicht zugelassen.

02 Wird kein Laufwerk angegeben, wird das Systemlaufwerk angenommen.

03 LINK prüft nicht auf Doppelbelegung im Speicher.

04 Die Protokollierung von unbefriedigten oder doppelt definierten Extern-Adressen am Bildschirm wird nach jeweils einem Bildschirminhalt angehalten. Durch Betätigung einer beliebigen Taste wird die Protokollierung fortgesetzt. Dies gilt nicht für den Aufruf des Binders über EXEC.

05 Wurde LINK über eine EXEC-Datei aufgerufen und treten unbefriedigte Extern-Adressen auf, so wird, wenn weder *A noch *I angegeben wurde, die Datei in verschieblichem Format (Parameter *R) geschrieben.

06 Die Parameter *C, *D und *S schliessen den Parameter *R aus, bei gleichzeitiger Angabe erfolgt eine Fehlermeldung. Ausnahme: Wenn nach '/' noch unbefriedigte Bezugsadressen existieren, kann in diesem Fall mit *R vorgebunden werden.

07 Der Parameter *I schliesst den Parameter *R aus.

08 Der Parameter *A schliesst den Parameter *I aus.

09 Pro Zeile dürfen mehrere Parameter, durch Kommas getrennt, angegeben werden. Die Reihenfolge ist von den Benutzerwünschen abhängig.
Zwischen Modulnamen können neue Ladeadressen spezifiziert werden.

Fehlermeldungen:

Bad assignment on SO	Nicht zulässiges Ausgabegerät.
IGNORED...	Der entsprechende Bindeparameter ist logisch oder syntaktisch falsch.
*R not allowed with *C/*D/*S	Ist *R angegeben, dürfen weder *C, *D noch *S angegeben werden.
*I not allowed with *R	*I und *R dürfen nicht gemeinsam angegeben werden.
*A not allowed with *I	*I und *A dürfen nicht gemeinsam angegeben werden.
NOT FOUND...	Die angegebene Datei wurde nicht gefunden.
NOT RELOC.:...	Die angegebene Datei enthält keinen verschieblichen Code.
No more memory-space available	Kein Speicherplatz mehr vorhanden. Die Anzahl der Externadressen muß reduziert werden.
Double defined Symbols:...	Die ausgegebenen Externadressen sind in wenigstens 2 Moduln definiert worden.
Undefined Externals:...	Die ausgegebenen Externadressen wurden nirgends definiert.
Bad reloc. input encountered	Innerhalb eines Moduls befinden sich nicht interpretierbare Daten (Keine bindbare Datei).
Symbol not found in PASS2. Linker Error.	Die LINK-interne Symboltabelle ist unvollständig aufgebaut worden (eventuell zuviele Symbole).

3.3 MEMDMP

Funktion: MEMDMP gibt einen RAM-Bereich als Binärdatei aus. Diese kann anschließend als Rrogramm geladen werden.

Aufruf: MEMDMP SO=<Ausgabedatei>$<Parameter>

<Ausgabedatei>: Name der Datei, in die der Speicherinhalt gebracht werden soll.

<Parameter>: Folge von drei hexadezimalen Zahlen (ohne "H"-Kennzeichnung!), getrennt durch Kommas. Die erste und zweite Zahl bezeichnen Beginn und Ende des auszugebenden Speicherbereichs, die dritte ist die spätere Startadresse des Programmes.

Wenn beim Aufruf von MEMDMP keine Parameter angegeben werden, meldet sich das Programm mit dem Eingabezeichen "?".

Der Bediener kann dann durch Kommas getrennte Adreßpaare eingeben (eines pro Zeile). Jedes Adreßpaar bedeutet einen auszugebenden Speicherbereich. Sobald eine Folge von drei Adressen in einer Zeile eingegeben wird, beginnt die Ausgabe der durch die früheren Adreßpaare spezifizierten Bereiche, und die zuletzt angegebene Adresse wird als Startadresse übernommen. Danach springt das Programm zurück in das BS1MP. Auf diese Weise können mehrere getrennte Speicherbereiche in eine Datei auf die Diskette geschrieben werden.

Bei der Startadresse wird keine Adreßrelativierung vorgenommen.

3.4 RASM

Funktion: Verschiebbarer Assembler. Das Programm übersetzt Quellcode, der in 8080-Assembler geschrieben ist, in einen nicht festadressierten Code. Der von RASM produzierte Code muß mit dem Programm LINK auf feste Adressen des Hauptspeichers "gebunden" werden, bevor er als ablauffähiges Programm geladen werden kann. Mit LINK können überdies mehrere mit RASM erzeugte Dateien zu einem Programm zusammengebunden werden.

Aufruf: RASM SI=<Eingabe>,SO=<Ausgabe>,
SL=<Auflistung>$<INFO>

oder: RASM SI=<Eingabe>,SO=<Ausgabe>,
AL=<Fehlerliste>$<INFO>

<Eingabe>	Dateiname des Quellprogramms (Source-Code)
<Ausgabe>	Dateiname des durch den Assembler erzeugten Moduls.
<Auflistung>	Name der Datei oder des Gerätes, in die bzw. auf das das Übersetzungsprotokoll ausgegeben wird.
<Fehlerliste>	Name der Datei oder des Gerätes, in die bzw. auf das die Fehlermeldungen ausgegeben werden. Wurde SL definiert, so hat diese Zuweisung keine Wirkung.
<INFO>	Angabe der Zeilenbreite (PWnnn, nnn = Anzahl Zeichen pro Zeile) und der Seitenlänge (PLnnn, nnn = Anzahl Zeilen pro Seite), getrennt durch Komma. Diese Angaben müssen nicht gemacht werden. INFO überschreibt die Steuerbefehle $PW und $PL im Programm (s.u.).

Wird SO nicht angegeben, dann wird kein Modul erzeugt. Sind weder SL noch AL zugewiesen, dann erscheinen die Fehlermeldungen auf CO.

RASM kennt folgende Steuerbefehle für die Listausgabe:

$TITLE	Alle die diesem Pseudobefehl in der selben Zeile folgenden Zeichen werden jeweils als erste Zeile gedruckt, wenn ein Seitenwechsel stattfindet. $TITLE kann in einem Programm mehrfach angegeben werden und überschreibt jeweils die vorhergehenden Spezifikationen.
$E	veranlaßt einen Seitenvorschub.
$PLnnn	nnn gibt die Anzahl der Zeilen pro Seite (12 - 255) an. Standardwert ist 48.
$PWnnn	nnn gibt die Anzahl der Zeichen pro Zeile an. Standardwert ist 110. Darüber hinausgehender Text wird abgeschnitten, die der Assemblerliste folgende Symboltabelle wird der Papierbreite angepasst.

Neben den Steuerbefehlen für die Listausgabe gibt es weitere Pseudobefehle, die den Übersetzerlauf von RASM beeinflussen, jedoch nicht in Maschinenbefehle umgesetzt werden. Drei Befehle steuern die drei Adreßzähler des Programms RASM, die zu den drei möglichen Segmentarten (Code-, Daten- und absolut adressiertes Segment) gehören. Code- und Datensegment können mit LINK beim Binden auf unterschiedliche Adressen gebunden werden, unterscheiden sich jedoch nicht in ihrer Funktion.

Mit dem Befehl ORG kann bereits beim Übersetzen eine absolute Adressierung vorgenommen werden.

CSEG — Bis zum nächsten DSEG oder ORG werden die Assemblerbefehle zum Codesegment hinzugefügt.

DSEG — Bis zum nächsten CSEG oder ORG werden die Assemblerbefehle zum Datensegment hinzugefügt.

ORG Adr. — Beginn eines absolut adressierten Segments. Die absolute Adresse folgt dem ORG-Befehl.

Wird weder CSEG, DSEG noch ORG angegeben, so wird CSEG angenommen.

PUBLIC — Symbol 1 , Symbol 2 ,...
definiert Einsprungpunkte in einem Modul, die von anderen Moduln verwendet werden können.

EXTRN — Symbol 1 , Symbol 2 ,...
definiert Bezüge zu anderen Modulen.

EQU — definiert einen symbolischen Namen, beispielsweise:

```
CLOSE  EQU   0FD27H
```

erlaubt es, die Adresse FD27H im Programm mit CLOSE zu bezeichnen.

END — ist der letzte Befehl eines Programmes. Eventuell folgende Befehle werden von RASM ignoriert.

Folgt dem Befehl END eine Adresse, z.B.

```
END  START
```

so wird sie als Startadresse des Moduls benutzt.

Beim Binden mehrerer Module mit LINK wird entweder die explizite Startadresse (*S=...) oder die erste in einem END-Befehl eines Moduls gefundene Adresse benutzt.
Die explizite Adresse hat Vorrang.

DS<Ausdruck> — Der <Ausdruck> kann eine Variable, eine Zahl oder ein mit + oder - verbundener Ausdruck aus beidem sein.

DS veranlasst RASM, den gerade gültigen Adreßzähler um den durch Ausdruck gegebenen Wert zu erhöhen. Der dadurch freigelassene Speicherraum bleibt undefiniert.

Module, die durch RASM erzeugt wurden, werden mit dem Programm LINK zusammengebunden und absolutiert (s. 3.2).

Assembler-Listing

Die Zeilen des Assembler-Listenausdrucks sind in vier Felder unterteilt:

Position	Inhalt
1 - 6	Adreßzähler, gefolgt von C für Code- oder D für Datensegment. Bei absoluten Adressen ist die Stelle 6 leer.
9 - 19	Assemblercode. Enthält er eine Adresse, so folgt ihm ein C, D oder X, je nachdem, ob es sich um eine Adresse im Code- oder Datenbereich oder in einem anderen Modul handelt.
21 - 25	Nummer der Quellcode-Zeile.
ab 26	Quellcode-Zeile. Tabulatorcodes im Quellcode generieren Blankzeichen bis auf Position 9, 16 oder 26. Quellcode-Zeichen werden in Großbuchstaben ausgedruckt mit Ausnahme von Zeichenketten, die in Hochkommas stehen (Strings).

Nach dem Assembler-Listing wird eine Symbol-Tabelle ausgedruckt. Es gibt die folgenden Symbol-Typen:

C	Das Symbol gehört zum Code-Segment. Die Adresse ist relativ zum Anfang des Segments.
D	Das Symbol gehört zum Daten-Segment. Die Adresse ist relativ zum Anfang des Segments.
X	Das Symbol ist extern definiert. Es wird keine Adresse angegeben.
*	Das Symbol ist mehrfach definiert (Fehler).
U	Das Symbol ist undefiniert (Fehler).

Steht keines der Zeichen neben dem Symbol, so besitzt es einen absoluten Wert. Ist ein Symbol PUBLIC deklariert, so steht ein * vor dem Symbolnamen.

RASM kann die folgenden Fehlermeldungen ausgeben:

01	Syntax-Fehler im Quellcode (nicht zulässiges Zeichen, Feld zu lang)
02	Unbekannter Befehls-Code
03	Zu viele Einträge in der Zeile
04	Symbol-Tabelle ist voll
05	Unzulässiger Ausdruck beim ersten Übersetzerdurchgang. Symbole in ORG- Ausdrücken müssen definiert sein, bevor sie verwendet werden.
06	EQU-Anweisung mit unzulässigem Operanden
07	Unzulässiger Ein-Byte-Ausdruck
08	Fehler bei Sprungmarke (s.Fehler 05)
09	Unzulässiger Ausdruck
10	Fehlender Ausdruck
11	Undefiniertes Symbol
12	Symbol mehrfach definiert
13	Falsche Register-Angabe
14	Zuviele Trennzeichen

3.5 XREF

Funktion: Erzeugung einer Querverweisliste für eine Quellcode-Datei.

Aufruf: XREF SI=<Eingabedatei>,SL=<Ausgabedatei>$nn

nn ist eine zweistellige Zahl und gibt die Anzahl der Zeilen pro Seite an.

Die Eingabedatei muß das Format der INTEL-8080-Assemblersprache haben.

In der Querverweisliste werden alle symbolischen Namen erfaßt, die nicht für die Assemblersprache reserviert sind.

Die Ausgabedatei enthält die symbolischen Namen und die Nummer der Zeile, in welcher sie angesprochen werden. Die Definitionszeile wird durch ein nachfolgendes "#" (23H) gekennzeichnet.

4. E D I T

4.1 EINFÜHRUNG

Der Editor EDIT dient der Erfassung und Veränderung von Programmen und Texten, allgemein von alphanumerischen Dateien.

Er ist zeilenorientiert, wobei eine Zeile bis zu 128 Zeichen lang sein kann, jedoch davon nur die ersten 80 Zeichen bearbeitet werden können.

Der Editor hält einen gewissen Textpuffer im Speicher, von dem ein Fenster von 23 Zeilen auf dem Bildschirm gezeigt wird.

Die Textzeilen können, wenn sie am Bildschirm gezeigt werden, direkt oder durch zahlreiche Textbearbeitungsfunktionen geändert werden.

Der Benutzer hat eine Reihe von Möglichkeiten, bestimmte Zeilen des Textes aufzufinden, etwa dadurch, daß er bestimmte Zeichenketten sucht oder den Text auf dem Bildschirm auf- und abwärts rollen läßt.

In diesem Handbuch wird unterschieden zwischen Funktionen und Befehlen, beide Begriffe werden unter der Bezeichnung Kommando zusammengefasst. Auch die Eingabe von Kontrollfunktonszeichen wird als Kommando bezeichnet.

Ein wesentlicher sachlicher Unterschied zwischen Befehlen und Funktionen liegt darin, daß Befehle im allgemeinen für vorbereitende und abschließende Aufgaben, wie z.B. zum Öffnen und Schließen von Dateien und zum Setzen der Tabulator-Stopps, benutzt werden, während Funktionen beim eigentlichen Editieren, z.B. beim Suchen von Zeichenketten, benutzt werden.

Der gesamte augenblicklich im Textpuffer vorhandene Text kann direkt angesprochen werden. Deshalb ist es vorteilhaft, einen möglichst großen Pufferbereich zu haben. Die Größe des Pufferbereichs hängt von der Generiervariante des Betriebssystems (s. 6.1) ab und evtl. paralell ablaufender Programme. Je kürzer die Zeilen sind, desto mehr Zeilen können im Hauptspeicher gehalten werden.

In diesem Kapitel werden bei der Beschreibung der einzelnen Kommandos die folgenden Zeichen verwendet:

|

senkrechte Striche trennen mehrere Alternativen, z. B. bedeutet

Parameter := Dezimahlzahl | " Zeichenfolge "

daß der Parameter entweder eine Dezimahlzahl oder eine Zeichenfolge ist.

Allgemeine Syntax der Kommandos:

Adreßbezug Funktionsname Funktionsparameter

Wird eine Zeilenadresse ohne einen Funktionsnamen eingegeben, so wird der Pufferzeiger an den Anfang dieser Zeile gestellt.

4.2 BEDIENUNG DES EDITORS

4.2.1 Aufruf des Editors

Der Editor wird durch folgenden Befehl aufgerufen:

```
EDIT  si=<Eingabedatei>,so=<Ausgabedatei>
      sl=<Listendatei>,ai=<Zusatzeingabedatei>
oder

EDIT  si=<Eingabedatei>,ai=<Zusatzeingabedatei>
      sl=<Listendatei>$<Erweiterung>
```

Der Befehl wird vom Benutzer über die Tastatur eingegeben. Der Editor kopiert Daten von der Eingabedatei in die Ausgabedatei unter Kontrolle der Befehle, die der Bediener über die Tastatur eingibt.

Die Listendatei ist inhaltlich mit der Ausgabedatei identisch, mit dem Unterschied, daß vor jeder Zeile die Zeilennummer ausgegeben wird.

Wenn keine Eingabedatei bearbeitet werden soll - z.B. beim erstmaligen Erstellen einer Datei - kann die Zuweisung einer Standard-Eingabedatei (SI) unterbleiben. Ebenso sind auch die anderen Dateizuweisungen nicht obligatorisch.

Die zweite Aufrufvariante dient zum Editieren von vorhandenen Dateien. Wird nach dem $ keine Erweiterung angegeben, wird die vorhandene Eingabedatei z. B. TEXT umbenannt in TEXT.BAK (Back-up-Datei) und eine neue Datei mit Namen TEXT erstellt. Bei Angabe einer Erweiterung wird die alte Datei unter dem Namen mit dieser Erweiterung gesichert. Wenn der A-Schalter (s. SYS) gesetzt ist und bereits eine Back-up-Datei mit gleicher Erweiterung existiert, wird diese automatisch gelöscht.

4.2.2 Bildschirmaufteilung

Der Bildschirm ist in zwei Bereiche unterteilt:

a) Zeile 1 - Kommando- und Mitteilungsfeld
b) Zeile 2-25 - Datenteil des Bildschirms

a) Kommando- und Mitteilungszeile:

Spalte 1	enthält einen Stern, der die Bereitschaft des Editors zur Annahme eines neuen Be fehls oder einer neuen Funktion anzeigt.
Spalte 2-72	Befehls-/Funktionsfeld
Spalte 20-72	enthält Fehlermeldungen des Editors
Spalte 73-77	zeigt die aktuelle Zeilennummer oder "NEW" an.
Spalte 78	enthält "/"
Spalte 79-80	zeigt die aktuelle Spaltennummer (1-80) an

Das Mitteilungsfeld - in dem sich noch das letzte Kommando und eventuell eine Meldung befinden - wird gelöscht, sobald ein neuer Befehl bzw. eine neue Funktion eingegeben wird.

Nach Abschluß eines Befehls/einer Funktion wird der Cursor entweder auf dasselbe Zeichen positioniert, auf das der Puffer-Zeiger deutet, oder er verbleibt im Kommandofeld. Dies hängt von dem Befehl/der Funktion ab.

Wurde ein Fehler entdeckt, so wird eine entsprechende Meldung ausgegeben. Danach sollten ein neuer Befehl oder eine neue Funktion eingegeben werden.

b) Datenteil des Bildschirms

Die übrigen Zeilen des Bildschirms stellen ein Fenster dar, das 24 Zeilen (23 im automatischen Roll-Modus) des Textpuffers zeigt.

Bei den Kommandos steht nach der Ausführung die aktuelle Zeile - d.h. diejenige, auf die der Puffer-Zeiger weist - in der Zeile 13 des Bildschirms, es sei denn, die aktuelle Zeile wurde durch das Kommando nicht verändert.

Man kann jedoch den Cursor im Text beliebig auf andere Zeilen positionieren und dort mit Hilfe der Kontrollfunktionszeichen den Text auch ändern. Möchte man neue Kommandos eingeben, so muß man den Cursor zunächst durch Drücken der HOME-Taste in die Kommando-Zeile setzen. Der Puffer-Zeiger bleibt dort stehen, wo sich der Cursor zuletzt befand. Ist das erste eingegebene Zeichen des Kommandos alphanumerisch, so wird die Kommando-Zeile gelöscht.

4.2.3 Einführung in die Kommandos

Der Texteditor zeigt seine Bereitschaft, Kommandos anzunehmen durch einen Stern (*) an, der an der ersten Stelle der ersten Zeile steht.

Zur Bedienung des Editors können drei Arten von Kommandos gegeben werden:

- allgemeine Befehle
- Textbearbeitungsfunktionen
- Kontrollfunktionstasten

Wird ein Kommando eingegeben, so zeigt ein Unterstrich die nächste freie Stelle im Kommandofeld an.

Befehle oder Funktionen können in Groß- oder Kleinbuchstaben eingegeben werden.

Es ist möglich, für die Befehls-/Funktionsnamen Abkürzungen zu verwenden. In dieser Beschreibung wird der Teil des Kommandos, der nicht unbedingt eingegeben werden muß, in Klammern gesetzt.

Kommandos können einzeln oder in Kommandofolgen eingegeben werden.

Jedes Kommando in einer Kommandofolge muß durch ein ";" abgetrennt werden.

Die Kommandos in einer Kommandofolge werden in der angegebenen Reihenfolge ausgeführt.

Konnte ein Kommando in einer Kommandofolge nicht ausgeführt werden oder lieferte sie ein fehlerhaftes Ergebnis, so wird der Rest der Folge nicht mehr ausgeführt.

Nach Eingabe der Wagenrücklauf-Taste (CR) wird der eingegebene Befehl, die eingegebene Funktion oder Funktionsfolge ausgeführt.

Wenn ein Befehl/ eine Funktion beendet ist, wird der Cursor entweder auf Zeile 1 oder auf Zeile 13 des Bildschirms positioniert.

Dies hängt vom Ergebnis des zuletzt ausgeführten Kommandos ab.

Im zweiten Fall kann jetzt die Position des Cursors durch Betätigen der Kontrollfunktionstasten verändert werden (siehe Kapitel 4.4). Ein neuer Befehl oder eine neue Funktion kann erst nach Drücken der HOME-Taste eingegeben werden. Gibt man CR ein, so wird das letzte Kommando wiederholt.

Der Editor läßt für die Standard-Eingabedatei (SI) eine Satzlänge von mehr als 80 Zeichen zu. Es werden jedoch höchstens 80 Zeichen auf dem Bildschirm angezeigt und bearbeitet. Die maximale Satzlänge beträgt 128 Zeichen.

4.2.4 Löschen von Eingabefehlern

Jeder bei der Eingabe eines Kommandos unterlaufene Fehler kann durch Drücken der CURSOR-LINKS-Taste (-) behoben werden. Dabei ist die Taste für jedes Zeichen, das gelöscht werden soll, einmal zu betätigen. (Steht der Cursor am linken Rand der Zeile, so bleibt die CURSOR- LINKS-Taste ohne Wirkung).

Durch Betätigen der DEL-Taste werden die Zeichen im Kommandofeld gelöscht. Der Editor meldet sich mit einem Stern.

4.2.5 Kommandowiederholungen

Kommandos oder Kommandofolgen können beliebig oft wiederholt werden, wenn die Zeichenfolge in spitze Klammern gesetzt und dieser eine Dezimalzahl vorangestellt wird. Die Zahl gibt an, wie oft die Wiederholung ausgeführt werden soll.

Syntax des Wiederholungskommandos:

```
n   'Funktion oder Funktionsfolge'
```

n muß zwischen 1 und 255 liegen.

Beispiel:

```
3 R;W
```

liest drei Pufferinhalte aus der Eingabedatei und schreibt sie in die Ausgabedatei. Wurde dabei das Ende der Eingabedatei erreicht, wird die Befehlsfolge mit einer Fehlermeldung abgebrochen.

4.3 KOMMANDOS DES EDITORS

4.3.1 Tabulator-Benutzung

Wenn der Editor Dateien einliest, die Tabulatorzeichen (09 und 18 hexadezimal) enthalten, expandiert er die Zeilen entsprechend den gerade gesetzten Tabulator-Stopps. Beim Hinausschreiben werden die Zeilen, sofern der TP-Befehl gegeben wurde, gemäß den aktuellen Tabulator-Stopps gepackt.

Achtung:
Der Benutzer muß darauf achten, daß die Tabulator-Stopps beim Einlesen mit denen beim vorhergehenden Hinausschreiben der Datei übereinstimmen.

4.3.1.1 Setzen von Tabulator-Stopps

Der Befehl zum Setzen von Tabulator-Stopps ist in vier Unterbefehle untergliedert:

Befehl: T(n1,n2,n3,...) (ni = Dezimalzahl)

TC COBOL Tab-Positionen
(1,8,12,16...)

TA ASSEMBLER Tab-Positionen
(1,9,16,26)

TF FORTRAN Tab-Positionen
(1,6,7,10,13,16,..)

Der erste Befehl wird zum Setzen beliebiger Tab-Stopps benutzt. Die Tab-Positionen werden durch Dezimalzahlen, die durch Kommas voneinander getrennt sind, angegeben.

Wird keine Zahl angegeben, so werden die Standardwerte genommen.

Standardwerte sind: 9, 17, 25, ...,

Es können maximal 15 Tab-Positionen gesetzt werden.

Früher festgelegte Tab-Positionen werden automatisch gelöscht, wenn dieser Befehl gegeben wird.

Der Wert des Pufferzeigers wird nicht verändert. Der Cursor steht anschließend an der ersten Stelle des Kommandofeldes.

4.3.1.2 Anzeige der aktuellen Tabulator-Stopps

Befehl: TD

Dieser Befehl bewirkt, daß die aktuellen Tabulatorpositionen in der Kommandozeile angezeigt werden.

Der Wert des Pufferzeigers wird nicht verändert.

4.3.2 Ein-/Ausgabebefehle

Ehe eine Ein-/Ausgabeoperation ausgeführt werden kann, muß eine Eingabe- bzw. Ausgabedatei festgelegt sein.

Wenn die Standard-Listendatei (SL) zugewiesen wurde, werden alle Ausgaben, die in die Standard-Ausgabedatei (SO) geschrieben werden, auch in die SL-Datei übertragen. Zusätzlich wird die entsprechende Zeilennummer vor jede Zeile gesetzt.

4.3.2.1 Lesen von Zeilen der Eingabedatei in den Textpuffer

Befehl: R(EAD) (n) (n=Dezimalzahl)

Dieser Befehl veranlaßt den Editor, n Zeilen von der Standard-Eingabedatei in den Textpuffer zu lesen. Wird n weggelassen, liest der Editor so lange, bis der Textpuffer zu dreiviertel gefüllt ist.

Wenn der Lesevorgang beendet ist, erscheint die letzte eingelesene Zeile in Zeile 13 des Bildschirms.

Pufferzeiger und Cursor stehen am Anfang der 13. Zeile.

Wenn ein Tabulator-Zeichen erkannt wird, wird die Zeile entsprechend den aktuellen Tabulator-Stopps expandiert. Wenn dadurch die Länge der Zeile 128 Bytes überschreitet, wird eine Fehlermeldung ausgegeben.

Beispiel:

```
R
```

Durch den Befehl wird der Textpuffer zu 3/4 mit Zeilen aus der Eingabedatei gefüllt. Die Größe des Textpuffers ist abhängig von der Betriessystemvariante.

4.3.2.2 Schreiben des Textpuffers in eine Ausgabedatei

Befehl: W(RITE) (*) (n)

(n = Dezimalzahl)

Dieser Befehl veranlaßt den Editor, n Zeilen in die angegebene Ausgabedatei zu schreiben. Wird n weggelassen, so wird der ganze Textpuffer übertragen.

Die kopierten Zeilen werden aus dem Textpuffer gelöscht, außer wenn dem Kommando WRITE ein Stern (*) folgt.

Der Pufferzeiger wird an den Anfang des Textpuffers gestellt.

Pufferzeiger und Cursor werden auf die erste Stelle der Zeile 13 des Bildschirms positioniert.

Beispiel:

W

Der Pufferinhalt wird in die Ausgabedatei geschrieben und anschließend gelöscht. Mit R kann jetzt der nächste Pufferinhalt eingelesen werden.

4.3.2.3 Schließen der Ausgabedatei

Befehl: OC

Der Befehl bewirkt, daß die aktuelle Ausgabedatei (SO/SL) geschlossen wird.

Ist keine Ausgabedatei definiert worden, wird ein Fehler gemeldet.

Der Puffer-Zeiger wird nicht verändert; der Editor bleibt im Kommando-Modus.

4.3.2.4 Öffnen einer Ausgabedatei

Befehl: OA "<Dateiname>"

Mit dem Kommando kann eine neue Ausgabedatei für SO zugewiesen werden. Der <Dateiname> muß den Konventionen des Betriebssystems BS1MP entsprechen. Zuweisungen von Mehrfachdateien sind nicht möglich.

War SO bereits zugewiesen oder existierte der angegebene Dateiname bereits, so wird eine Fehlermeldung ausgegeben.

Der Puffer-Zeiger wird nicht verändert; der Editor bleibt im Kommando-Modus.

Beispiel:

Wenn während des Editierens die Diskette "überläuft", kann man in folgender Weise eine Folgedisketten-Behandlung vornehmen:

Mit OC

wird die Ausgabedatei auf der vollen Diskette geschlossen, anschließend wird mit

OA":D1:Fortse.tz"

auf Laufwerk 1 eine Fortsetzdatei eröffnet. Mit

W

wird der Rest des Textpuffers in die neue Datei geschrieben.

4.3.2.5 Kopieren bis zu einer angegebenen Adresse

Befehl: CU Zeilennummer
CU "<Zeichenfolge>"

Der Befehl bewirkt, daß die Zeilen der Eingabedatei in die Ausgabe- und/oder Listdatei kopiert werden, bis die angegebene Zeilennummer bzw. die angegebene Zeichenfolge gefunden wurde.

Die angegebene Zeile wird an den Beginn des Puffers gespeichert, anschliessend liest der Editor bis zum Ende der Datei ein bzw. solange, bis der Puffer zu 3/4 voll ist.

Ist der Wert der angegebenen Zeilennummer kleiner als der der ersten Zeile im Puffer oder sind SI oder SO nicht zugewiesen, wird ein Fehler angezeigt.

Der Puffer-Zeiger wird nicht verändert; der Editor bleibt im Kommando-Modus.

4.3.2.6 Einlesen ab einer angegebenen Adresse

Befehl: SU Zeilennummer
SU "<Zeichenfolge>"

Das Kommando bewirkt, daß die Eingabedatei SI durchgelesen wird, bis die eingegebene Zeilennummer bzw. die <Zeichenfolge> gefunden wird. Diese Zeile wird an den Anfang des Puffers gespeichert, der Editor liest anschliessend bis an das Ende der Datei ein bzw. solange, bis der Puffer zu 3/4 gefüllt ist.

Ist die angegebene Zeilennummer kleiner als die erste Zeilennummer im Puffer oder wurde keine Eingabedatei zugewiesen, erfolgt eine Fehlermeldung.

Wird die angegebene Zeichenfolge nicht gefunden, liest der Editor bis an das Ende der Datei.

Der Puffer-Zeiger wird an den Anfang des Puffers positioniert.

4.3.2.7 Normaler Abschluß

Befehl: E(XIT)

Durch diesen Befehl kopiert der Editor den Textpufferinhalt und falls vorhanden den Rest der Eingabedatei direkt in die Ausgabedatei(en).

Die Kontrolle geht anschließend wieder an das Betriebssystem zurück, das die zugewiesenen Dateien schließt.

Wenn keine Ausgabedatei zugewiesen wurde, erfolgt eine Fehlermeldung. Mit dem Kommando OA kann eine Ausgabedatei zugewiesen werden und dann das EXIT-Kommando wiederholt werden.

4.3.3 Zusätzliche Eingabedatei

Der Editor bietet die Möglichkeit, zwei Eingabedateien zu mischen.

Folgende Befehle können dazu verwendet werden:

4.3.3.1 Zuweisung der zusätzlichen Eingabe:

Befehl: AA "<Dateiname>"

Der <Dateiname> muß den Konventionen des Betriebssystems BS1MP entsprechen.

Dieser Befehl ist einzugeben, wenn eine zweite Eingabedatei gewünscht wird.

Der Wert des Puffer-Zeigers wird nicht verändert.

Der Cursor steht an der ersten Stelle des Kommandofeldes.

4.3.3.2 Der Befehl AI

Befehl: AI(n)(* L)
AI "<Zeichenfolge>"(* L)

(n = Dezimalzahl, L = Zeilenadresse)

Der Befehl bewirkt, daß der Editor n Zeilen aus der zusätzlichen Eingabedatei liest - oder bis zum ersten Auftreten von <Zeichenfolge> - und diese Zeilen hinter der angegebenen Zeilenadresse in den Puffer einfügt.

Wird "* L" weggelassen, so wird hinter die aktuelle Zeile eingefügt.

Die Datei wird automatisch geschlossen, wenn das Dateiende erreicht wird.

Der Puffer-Zeiger wird auf die letzte eingelesene Zeile gesetzt.

4.3.3.3 Der Befehl AS

Befehl: AS (n)
AS "<Zeichenfolge>"

(n = Dezimalzahl)

Der Befehl bewirkt, daß n Zeilen - oder alle Zeilen bis zum Auftreten von <Zeichenfolge> - in der zusätzlichen Eingabedatei überlesen werden.

In der zweiten Form des Befehls wird auch die Zeile mit der Übereinstimmung überlesen.

Wird n ausgelassen oder <Zeichenfolge> im Rest der Datei nicht gefunden, überliest der Editor den Rest der Datei.

Bei Erreichen von Dateiende wird die Datei automatisch geschlossen.

Der Puffer-Zeiger wird nicht verändert, der Editor bleibt im Kommando-Modus.

Beispiel:

In ein Assemblerprogramm, das gerade editiert wird, möchte man einen Abschnitt aus einem anderen Programm ("ABC.SRC") kopieren. Der Abschnitt beginnt mit der Sprungmarke "ANFANG:" und reicht bis zur Sprungmarke "ENDE:". Der Abschnitt soll hinter die aktuelle Zeile in den Textpuffer kopiert werden. Anschließend soll die zusätzliche Datei geschlossen werden.

```
AA"ABC.SRC"

AS"ANFANG:"

AI"ENDE:"

AS"XXXXXXXXX"
```

Da XXXXXXXXX in ABC.SRC nicht vorkommt, wird die Datei bis zum Ende durchsucht und dann geschlossen.

4.3.4. Zeilenadressierung

Der Inhalt des Textpuffers läßt sich in zwei Kategorien unterteilen:

- Textzeilen, die von der Standard-Eingabedatei gelesen wurden

- neue bzw. geänderte Textzeilen.

Für die zweite Kategorie sind die Adressierungsmöglichkeiten stärker eingeschränkt als für die erste.

Es gibt fünf verschiedene Möglichkeiten, um eine Zeile anzusprechen:

1 - Zeilennummer (Dezimalzahl)
2 - Zeichenfolge
3 - Puffer-Zeiger ($)
4 - erste Zeile (:)
5 - letzte Zeile (.)

Zu jeder Zeilenadresse kann eine Dezimalkonstante addiert oder subtrahiert werden: z.B.: $+5 oder: .-10.

Zeilennummer

Jede Zeile, die von der Standard-Eingabedatei gelesen wurde, kann direkt durch ihre Zeilennummer angesprochen werden. Der Benutzer braucht hierfür nur die Dezimalzahl einzugeben.

Neue oder geänderte Zeilen können nicht auf diese Weise adressiert werden.

Beispiel:

```
210      (in Kommandozeile)
```

Der Pufferinhalt wird so gezeigt, daß die Zeile 210 in der Mitte des Bildschirms (Zeile 13) steht.

Zeichenfolge

Eine Zeile kann direkt durch eine Zeichenfolge, die in dieser Zeile steht, angesprochen werden. Die Zeichenfolge muß in Anführungsstrichen eingegeben werden.

Eine Zeichenfolge kann ein oder mehrere Zeichen "senkrechter Strich" enthalten. Stellen, die in dieser Weise markiert sind, werden nicht untersucht.

Das ASCII-Zeichen für "senkrechter Strich" ist 7EH. Auf ECMA-Tastaturen wird dieses Zeichen als "ß" dargestellt.

Beispiel: siehe "Textintervall".

Puffer-Zeiger

Der Editor besitzt zwei veränderliche Zeiger:

- einen Zeilenzeiger
- einen Zeichenpositionszeiger

Die Werte dieser beiden Zeiger hängen von der ausgeführten Editierfunktion ab.

Diese beiden Zeiger werden fortan als Puffer-Zeiger bezeichnet.

Der Zeilenzeiger kann durch Betätigen der "$"Taste adressiert werden.

Der Spaltenzeiger wird entweder direkt durch den Benutzer verändert, indem dieser den Cursor bewegt, oder indirekt durch verschiedene Befehle und Funktionen.

Erste Zeile

Die erste Zeile des Textpuffers kann durch Eingabe des Zeichens ":" angesprochen werden.

Letzte Zeile

Die letzte Zeile des Textpuffers kann durch Eingabe des Zeichens "." angesprochen werden.

Textintervall

Ein Textintervall ist eine Folge von einer oder mehreren aufeinanderfolgenden Zeilen. Die erste und die letzte Zeile müssen - getrennt durch ein Komma - angegeben werden. Enthält das Intervall nur eine Zeile, reicht es aus, nur sie zu nennen.

In diesem Handbuch werden Textintervalle durch I gekennzeichnet.

Ein Intervall kann also definiert sein durch:

I = <Zeile>

oder

I = <Zeile 1> , <Zeile 2>

(ist <Zeile 1> _ <Zeile 2> gilt <Zeile 1> = <Zeile 2>)

Ist <Zeile 1> leer, so wird für <Zeile 1> die erste Zeile angenommen.

Ist <Zeile 2> leer, so wird für <Zeile 2> die letzte Zeile angenommen.

Das zuletzt definierte Textintervall kann mit "&" angesprochen werden.

Beispiele: (Textintervalle sind unterstrichen)

:,.f"abcd"

Mit dem Befehl wird im Intervall ":,.", d.h. dem ganzen Textpuffer, nach der Zeichenkette "abcd" gesucht.

"Anf",$s"abcd""efgh"

Im Textintervall, das vom ersten Auftreten der Zeichenkette "Anf" bis zur aktuellen Zeile ($) reicht, wird "abcd" durch "efgh" ersetzt.

100,200dl

Das Textintervall von Zeile 100 bis Zeile 200 wird gelöscht.

4.3.5 Zeichenweise Positionieren

Verschieben des Puffer-Zeigers

Syntax:	Adreßbezug	:: =	leer
	Funktionsname	:: =	P(N)
	Funktionsparameter	:: =	Vorzeichen
			| Dezimalkonstante

Der Puffer-Zeiger wird um die durch die Dezimalkonstante angegebene Anzahl von Stellen entweder nach rechts (Vorzeichen = leer oder +) oder nach links (Vorzeichen = -) verschoben.

Der Puffer-Zeiger wird nicht über die aktuelle Zeile hinausbewegt.

Der Cursor wird auf das Zeichen gesetzt, das dem neuen Wert des Pufferzeigers entspricht.

Beispiele:

PN3 - bewirkt, daß der Cursor um 3 Stellen nach rechts bewegt wird.

100,200F"xx";pn+5
- wird im Zeilenintervall zwischen Zeile 100 und 200 die Zeichenfolge "xx" gefunden, so wird der Cursor von dieser Zeichenfolge aus 5 Stellen weiter nach rechts bewegt

:,"xx"f"yy";pn-3
- wenn die Zeichenfolge gefunden wurde, wird der Cursor 3 Stellen links davon positioniert.

Zeilenanfang

Syntax:	Adreßbezug	:: =	leer
	Funktionsname	:: =	PS
	Funktionsparameter	:: =	leer

Die Funktion bewirkt, daß der Puffer-Zeiger auf den Anfang der aktuellen Zeile gestellt wird.

Zeilenende

Syntax	Adreßbezug	:: =	leer
	Funktionsname	:: =	PL
	Funktionsparameter	:: =	leer

Die Funktion bewirkt, daß der Puffer-Zeiger auf das letzte Zeichen der Zeile gestellt wird, das nicht Blank ist.

4.3.6 Löschfunktionen

Zeilen löschen

Syntax:	Adreßbezug	::=	Textintervall
	Funktionsname	::=	DL
	Funktionsparameter	::=	leer

Die durch das Textintervall angegebenen Zeilen werden gelöscht.

Der Puffer-Zeiger wird auf die erste Stelle, die der letzten gelöschten Zeile folgt, positioniert. Wurde die letzte Zeile des Textpuffers gelöscht, wird der Puffer-Zeiger an den Anfang der neuen letzten Zeile gestellt.

Die aktuelle Zeile wird in der 13. Zeile des Bildschirms angezeigt.

Beispiele:

100dl	- Die Zeile 100 wird aus dem Textpuffer entfernt
200,250DL	- die Zeilen 200 bis einschl. 250 werden gelöscht
:,"xxx"d	- die erste Zeile und alle folgenden einschließlich der ersten Zeile, die die Zeichenfolge "xxx" enthält, werden gelöscht.

Zeichen löschen

Syntax:

Adreßbezug	::=	leer
Funktionsname	::=	D(C)
Funktionsparameter	::=	leer
		Dezimalkonst. |
		"<Zeichenfolge>"

Ist der Funktionsparameter leer oder eine Dezimalzahl, wird die angegebene Anzahl von Zeichen, beginnend mit dem durch den Puffer-Zeiger markierten Zeichen, gelöscht.

Alle folgenden Zeichen in derselben Zeile werden um die angegebene Anzahl von Stellen nach links geschoben.

Wenn kein Wert angegeben wurde, wird der Wert 1 angenommen.

Der Puffer-Zeiger und der Cursor weisen auf das dem gelöschten folgende Zeichen.

Wird eine<Zeichenfolge>angegeben, werden alle Zeichen, beginnend bei der Position des Puffer-Zeigers, bis zur angegebenen <Zeichenfolge> - ausschliesslich der Zeichenfolge selbst - gelöscht.

Wird in der aktuellen Zeile keine Übereinstimmung mit der angegebenen Zeichenfolge gefunden, werden keine Zeichen gelöscht.

Beispiele:

d	- Löschen eines Zeichens
dc5	- Löschen von 5 Zeichen
d"t1"	- wird "t1" in der aktuellen Zeile gefunden, werden alle Zeichen von der Puffer-Position bis "t1" gelöscht.

4.3.7 Modus: Zeilen einfügen

Syntax:

Adreßbezug	::=	leer | Zeilenadresse
Funktionsname	::=	I(L)|A(L)
Funktionsparameter	::=	leer | Dezimalzahl

Um Zeilen vor der ersten Zeile des Textpuffers einzufügen bzw. an die letzte Zeile anzuhängen, bietet der Zeileneinfügungsmodus zwei Funktionen:

- Einfügen vor eine angegebene Zeile
- Anfügen nach einer angegebenen Zeile

Wenn kein Adreßbezug angegeben wird, wird die Zeile, die der Pufferzeiger markiert, angenommen.

Funktionsparameter :: = leer

Zeile 13 des Bildschirms wird gelöscht und der Cursor auf die erste Stelle gesetzt.

Die verbleibenden Zeilen des Bildschirms - nicht der des Puffers - werden, soweit möglich, mit Text gefüllt.

Im Fall von "IL" findet sich die angegebene Zeile anschließend auf Zeile 14 des Bildschirms, im Falle A auf Zeile 12.

Diese beiden Funktionen verändern die Wirkung der Wagenrücklauf (CR)-Taste (siehe Kap. 4.4).

Der Modus: ´Zeilen Einfügen´ bleibt wirksam, bis der Cursor durch Betätigen einer Kontrollfunktionstaste aus der aktuellen Zeile entfernt wird (Ausnahme: Wagenrücklauf=CR, CURSOR RECHTS, CURSOR LINKS, TAB VORWÄRTS; oder wenn bei Eingabe alphanumerischer Zeichen Spalte 80 überschritten wird).

Die Funktionen I und A werden nur wirksam, wenn sie an der letzten Stelle innerhalb einer Funktionsfolge stehen.

Funktionsparameter :: = Dezimalkonstante

Die angegebene Zahl von Leerzeilen wird vor bzw. hinter die angegebene Zeilenadresse in den Puffer eingefügt, abhängig davon, ob I oder A verwendet wurde.

4.3.8 Zeichenfolgen einfügen

Syntax:

Adreßbezug	::= leer
Funktionsname	::=IT|AT
Funktionsparameter	::=" Zeichenfolge "

Wird der IT-Funktionsname benutzt, so wird die angegebene Zeichenfolge vor dem durch den Puffer-Zeiger markierten Zeichen in den Textpuffer eingefügt.Bei AT wird die Zeichenfolge dahintergesetzt.

Würde durch die Funktion die Zeile länger als 80 Zeichen, wird keine Einfügung vorgenommen und eine Fehlermeldung ausgegeben.

Der Puffer-Zeiger und der Cursor werden auf das erste Zeichen nach der Einfügung gestellt.

Beispiele:

IT"xx"	- Die Zeichenfolge "xx" wird vor dem Zeichen eingefügt, auf das der Cursor positioniert ist.
10,20f"xx";it"y"	- wenn zwischen Zeile 10 und 20 die Zeichenfolge "xx" das erste Mal auftaucht, wird davor die Zeichenfolge "y" eingefügt. Ist die Zeichenfolge "xx" im Intervall nicht vorhanden, wird keine Einfügung vorgenommen.

4.3.9 Zeilen kopieren

Syntax:

Adreßbezug	::= leer | Zeilenadresse
Funktionsname	::= C(L)
Funktionsparameter	::= leer | Textintervall

Die Funktion bewirkt, daß das angegebene Textintervall vor die angegebene Zeile kopiert wird.

Ist der Adreßbezug leer, wird die aktuelle Zeile angenommen. Wird kein Funktionsparameter angegeben, wird die aktuelle Zeile kopiert.

4.3.10 Textzeilen verschieben

```
Syntax:   Adreßbezug          :: leer | Zeilenadresse
          Funktionsname       ::=M(L)
          Funktionsparameter  ::= leer|
                                  Textintervall
```

Diese Funktion bewirkt, daß das angegebene Textintervall vor den angegebenen Adreßbezug geschoben wird.

Wenn kein Adreßbezug angegeben wurde, wird die durch den Puffer-Zeiger markierte Zeile angenommen.

Das Textintervall wird aus seiner früheren Position entfernt.

Beispiele:

```
100m20,30        - Die Zeilen 20-30 werden entfernt und
                   vor Zeile 100 gestellt.
```

4.3.11 Austauschfunktionen

Zeichenfolge ersetzen

```
Syntax:   Adreßbezug          ::= leer | Textintervall
          Funktionsname       ::=S(T)
          Funktionsparameter  ::="Zeichenfolge1""Zeichenfolge2"
```

Durch diese Funktion kann eine Zeichenfolge, Folge 1, durch eine andere, Folge 2, im angegebenen Textintervall ersetzt werden.

Wenn kein Adreßbezug angegeben wird, wird die aktuelle Zeile angenommen.

Die letzte Zeile, in der eine Ersetzung durchgeführt wurde, wird auf Zeile 13 des Bildschirms angezeigt. Die verbleibenden Zeilen des Bildschirms werden, soweit möglich, mit Text gefüllt.

Puffer-Zeiger und Cursor werden auf das Zeichen gesetzt, das der letzten Ersetzung folgt.

Die Anzahl der Ersetzungen wird im Mitteilungsfeld in Zeile 1 angezeigt, wenn die Funktion die letzte in einer Funktionsfolge war.

Beispiele:

100s"xx""yy"	- Die Zeichenfolgen "xx" in Zeile 100 werden, sofern vorhanden, durch "yy" ersetzt
:,"xx"ST"y""zz"	- in dem angegebenen Intervall d.h. zwischen dem Anfang des Textpuffers und der ersten Zeile, die die Zeichenfolge "xx" enthält, wird "y" jedes Mal durch die Zeichenfolge "zz" ersetzt.

Wort ersetzen

Syntax:

Adreßbezug	::= leer | Textintervall
Funktionsname	::=SW
Funktionsparameter	::="Zeichenfolge1""Zeichenfolge2"

Durch diese Funktion wird im eingegebenen Textintervall eine Zeichenfolge, die ein Wort darstellt, Folge 1, durch eine andere Folge, Folge 2, ersetzt.

Wenn kein Adreßbezug angegeben wird, wird die aktuelle Zeile angenommen.

Die letzte Zeile, in der eine Ersetzung durchgeführt wurde, wird auf dem Bildschirm in Zeile 13 angezeigt. Die verbleibenden Zeilen des Bildschirms werden, soweit möglich, mit Text gefüllt.

Puffer-Zeiger und Cursor stehen auf dem Zeichen, das dem letzten ersetzten folgt.

Die Anzahl der Ersetzungen wird im Mitteilungsfeld in Zeile 1 angezeigt, wenn die Funktion als letzte in einer Funktionsfolge stand.

Beispiele:

100sw"xx""yy" :,"zz"sw"ABC""abc"	- gleiche Wirkung wie bei der Funktion "Zeichenfolge ersetzen", außer wenn die Zeichenfolgen "xx" oder "ABC" Teile einer längeren Zeichenfolge sind. In diesem Fall wird keine Ersetzung durchgeführt.

4.3.12 Suchfunktionen

Zeichenfolge vorwärts suchen

```
Syntax:   Adreßbezug          ::= leer | Textintervall
          Funktionsname       ::=F(F)
          Funktionsparameter  ::="Zeichenfolge"
```

Wird kein Adreßbezug angegeben, so wird beginnend mit dem Zeichen, auf das der Puffer-Zeiger zeigt, der Textpuffer bis zum Ende nach der angegebenen Zeichenfolge durchsucht.

Die Zeile, in der die Zeichenfolge auftaucht, wird in Zeile 13 des Bildschirms angezeigt und der Rest des Bildschirms wird, soweit möglich, mit Text gefüllt.

Wenn ein Textintervall angegeben wurde, so wird es von Anfang bis Ende durchsucht.

Wenn keine Übereinstimmung gefunden wurde, wird eine Meldung ausgegeben.

In diesem Fall wird der Wert des Puffer-Zeigers nicht verändert.

Zeichenfolge rückwärts suchen

```
Syntax:   Adreßbezug          ::= leer
          Funktionsname       ::=FB
          Funktionsparameter  ::="Zeichenfolge"
```

Die Funktion bewirkt, daß beginnend von dem Zeichen, auf das der Puffer-Zeiger weist, der Textpuffer rückwärts bis zur ersten Übereinstimmung auf die Zeichenfolge durchsucht wird.

Die Zeile, die die Übereinstimmung enthält, wird am Bildschirm in Zeile 13 angezeigt und der Rest des Bildschirms wird mit Text angefüllt, wenn dies möglich ist.

Falls keine Übereinstimmung gefunden wurde, wird eine Meldung ausgegeben.

Der Wert des Pufferzeigers wird nicht verändert.

4.3.13 Funktionen zur Zeilenmanipulation

Zeilen trennen

Syntax:

Adreßbezug	::=	leer
Funktionsname	::=	LS|LSL
Funktionsparameter	::=	leer

Der Befehl bewirkt, daß die aktuelle Zeile in zwei Zeilen zerlegt wird.

Der Text wird, beginnend mit der Puffer-Zeiger-Position, in die neue Zeile verschoben.

Soll eine 128 Zeichen lange Zeile getrennt werden, muß die zweite Form des Befehls benutzt werden.

Wird die zweite Form des Befehls verwendet, wird der Text außerdem links ausgerichtet.

Der Puffer-Zeiger wird nicht verändert.

Zeilen verketten

Syntax:

Adreßbezug	::=	leer
Funktionsname	::=	L(C)
Funktionsparameter	::=	leer | Dezimalkonstante

Die Funktion bewirkt die Verkettung der aktuellen mit der darauffolgenden Zeile. Führende Leerzeichen in der zweiten Zeile werden ausgelassen.

Wird die resultierende Zeile nicht länger als 80 Zeichen, so wird keine Dezimalkonstante angegeben. Wird eine Konstante angegeben, so kann die Zeile bis 128 (Dezimalkonstante) lang werden.

Der Puffer-Zeiger wird nicht verändert.

4.3.14 Zusätzliche Befehle

Die in diesem Abschnitt beschriebenen Befehle werden beim Editieren normalerweise nicht verwendet. Sie können jedoch für den Bediener manchmal nützlich sein.

Der Wert des Puffer-Zeigers wird nicht verändert.

Der Cursor steht an der ersten Stelle des Kommandofeldes.

Löschen des Textpuffers:

Befehl: K(ILL)

Der Befehl bewirkt, daß der gesamte Textpuffer gelöscht wird.

Rückkehr ins Betriebssystem

Befehl: Q(UIT)

Dieser Befehl ermöglicht eine sofortige Rückkehr ins Betriebssystem.

Die zugewiesenen Dateien werden geschlossen.

Ausgabedatei retten

Sollte während der Arbeit mit dem Editor das System aus irgendeinem Grund "abstürzen", so kann man den Inhalt des Textpuffers dennoch in die Ausgabedatei wegschreiben. Bis auf die letzte Änderung vor dem Absturz werden die Daten gerettet.

Folgendes Monitor-Kommando muß gegeben werden:

.G2803 CR-Taste

Dadurch wird eine Routine angesprungen, die den Puffer in die Ausgabedatei schreibt und die Datei abschließt.

4.4 KONTROLLFUNKTIONSTASTEN

Die folgenden Cursorpositionierungstasten stehen zur Verfügung:

- der Cursor wird auf das vorhergehende Zeichen in der aktuellen Zeile bewegt

- a) im Modus "Zeilen Einfügen": der Cursor wird auf die nächste Tabulatorposition in der aktuellen Zeile gestellt

  b) sonst wird der Cursor auf das nächste Zeichen in der aktuellen Zeile gestellt

  der Cursor wird auf dieselbe Stelle in der vorhergehenden Zeile positioniert

  der Cursor wird auf dieselbe Stelle in der folgenden Zeile gestellt

der Cursor wird an den Anfang der ersten Zeile des Bildschirms gestellt (Kommando-Modus)

CR/WAGENRÜCK-LAUFTASTE

a) im Modus "Zeilen Einfügen": die Zeilen vor der gerade eingefügten Zeile und die Zeile selbst werden um eine Zeile nach oben gerollt; die Eingabezeile wird gelöscht und der Cursor auf die erste Position gestellt

b) sonst wird der Cursor auf das erste Zeichen der folgenden Zeile gestellt

der Inhalt des Bildschirms wird um eine Zeile nach oben gerollt, wenn mehr Zeilen im Textpuffer vorhanden sind; sonst hat die Taste keine Wirkung; die Cursorposition wird nicht verändert.

der Inhalt des Bildschirms wird eine Zeile nach unten gerollt, wenn nicht der Anfang des Textpuffers angezeigt wurde; sonst hat die Taste keine Wirkung; die Stellung des Cursors wird nicht verändert

(CONTROL + I) der Cursor wird auf die nächste Tab-Stelle in der aktuellen Zeile gestellt. Wenn keine weiteren Tabs vorhanden sind, wirkt diese Taste wie die CURSOR RECHTS-Taste

(CONTROL + G) der Cursor wird auf die vorhergehende Tab-Position in der aktuellen Zeile gesetzt

Folgende Tasten für Editierfunktionen stehen zur Verfügung:

CHAR (CONTROL + P) ZEICHEN EINFÜGEN (nicht in der Kommando-Zeile). Durch die Funktionstaste wird ein Leerzeichen in der Position eingefügt, auf die der Puffer-Zeiger weist. Der Rest der Zeile wird um eine Position nach rechts verschoben. Überschreitet die Zeilenlänge 80 Zeichen, geht das letzte Zeichen verloren.

CHAR (CONTROL + Q) ZEICHEN LÖSCHEN (nicht in der Kommado-Zeile). Das Zeichen, auf das der Puffer-Zeiger weist, wird gelöscht, der Rest der Zeile wird um eine Stelle nach links verschoben. Die letzte Stelle wird mit Leerzeichen ausgefüllt.

DEL löscht den Inhalt einer Zeile

4.5 Zusammenfassung der Editor Kommandos

Ein-/Ausgabefunktionen

AA	- Zuweisen einer zusätzlichen Eingabedatei
AI	- Einfügen von zusätzlichen Eingabedaten
AS	- Auslassen von zusätzlichen Eingabedaten
CU	- Kopieren bis zur vorgegebenen Adresse
E(XIT)	- Normale Rückkehr ins Betriebssystem
K(ILL)	- Löschen des Textpuffers
O(A)	- Zuweisung einer Ausgabedatei
OC	- Schließen der Ausgabedatei
Q(UIT)	- Sprung ins Betriebssystem
R(EAD)	- Lesen von der Standard-Eingabedatei (SI)
SU	- Überlese bis zur vorgegebenen Adresse
T(n1,..)	- Setzen Tab-Stopps
TA	- Setzen von Assembler-Tab-Positionen
TC	- Setzen von COBOL-Tab-Positionen
TD	- Anzeige der aktuellen Tab-Positionen
TF	- Setzen von FORTRAN-Tab-Positionen
W(RITE)	- Schreiben des Textpufferinhalts in Ausgabedatei (SO/SL)

Textbearbeitungsfunktionen

A(L)	- Zeile(n) anfügen (Zeileneinfügungsmodus)
AT	- Text anfügen
C(L)	- Zeile kopieren
D(C)	- Zeichen löschen
DL	- Zeile(n) löschen
FB	- Zeichenfolge rückwärts suchen
F(F)	- Zeichenfolge vorwärts suchen
I(L)	- Zeile(n) Einfügen (Zeileneinfügungsmodus)
IT	- Zeichenfolge einfügen
L(C)	- Zeilen verketten
LS	- Zeile zerlegen
LSL	- Zeile zerlegen mit linksbündiger Ausrichtung
M(L)	- Textzeile(n) verschieben
PL	- Puffer-Zeiger auf letztes Zeichen positionieren
P(N)	- Puffer-Zeiger um die angegebene Anzahl von Zeichen verschieben
PS	- Puffer-Zeiger auf Zeilenanfang positionieren
S(T)	- Zeichenfolge ersetzen
SW	- Wort Ersetzen

4.6 Fehlermeldungen des Editors

E00: Interner Fehler
E01: Textpuffer-Überlauf
E02: Textpuffer-Bereich zu klein
E03: Zuviele Funktionswiederholungen
E04: Unerlaubtes oder falsch geschriebenes Kommando
E05: Unzulässiger Wert eines Adreßbezugs
E06: Fehlerhafte Syntax eines Adreßbezugs
E07: Fehlende Zahl nach + oder - Zeichen in einem Adreßbezug
E08: Unerlaubter Anfang einer Zeichenfolge
E09: Zeichenfolge wurde nicht mit " abgeschlossen
E10: Angegebene Zeichenfolge wurde nicht gefunden
E11: Angegebene Zeilennummer wurde nicht gefunden
E12: Keine Ausgabedatei zugewiesen
E13: Betrirbssystemfehler Code xxH
E14: Keine Eingabedatei zugewiesen
E15: Unzulässiger Tab-Wert
E16: Angabe zuvieler Tab-Stopps
E17: Keine kopierbare Zeile vorhanden
E18: Keine Einfügung durchgeführt
E19: Unzulässiger Wert der Dezimalkonstante
E20: Unzulässige Syntax des Befehls/ der Funktion
E21: Keine Verkettung durchgeführt
E22: Kommando durch Bediener unterbrochen
E23: Steuerzeichen verursachen Eingabezeile größer 128 Zeichen
E24: Textpuffer leer
E25: Kein löschbares Zeichen vorhanden
E26: Versuch zu lesen nach Dateiende
E27: Senkrechter Strich in Zeichenfolge 2
E28: Kommandos müssen durch ; getrennt werden
E29: Kein Platz mehr auf der Diskette
E30: Ausgabedatei existiert bereits auf Diskette
E31: unerlaubter Dateiname
E32: Ausgabedatei wurde schon zugewiesen
E33: Eingabedatei wurde schon zugewiesen
E34: Datei kann wegen Mangel an Hauptspeicherplatz nicht eröffnet werden. Fehler kann durch Schließen überflüssiger Dateien behoben werden.
E35: Zu viele Dateien eröffnet. Bei Aufruf des Editors dürfen nur SI, SO, SL und AI zugewiesen werden.
E36: Unerlaubtes Ausgabegerät
:CO: darf nicht als Ausgabegerät zugewiesen werden
E37: Leere Zeichenfolge
E38: Ersetzen von Zeichen bewirkt Verlängerung der Zeile auf mehr als 80 Zeichen
E39: Angegebene Zeilenlänge ist zu klein.

5 Der Monitor

Der Monitor ist ein Teil des Betriebssystems BS1MP. Er befindet sich als Datei mit dem Namen ZZMON auf der Diskette oder Festplatte und wird vom Urlader während des Einschaltens in den Hauptspeicher geladen. Im Menü wird angegeben, von wo der Monitor geladen wird.
Der Zweck des Monitors ist es, alle geräteabhängigen Ein-/Ausgaben zu unterstützen. Folgende Geräte werden mit Treiberroutinen unterstützt:

- SD-Diskette
- DD-Diskette
- Festplatte
- Tastatur
- Bildschirm
- Drucker
- serielle, asynchrone Ein-/Ausgabe

Der Monitor enthält auch die notwendigen Dateieingaberoutinen zum Laden von Programmen von Festplatte oder Diskette in den Hauptspeicher, z. B. des eigentlichen BS1MP. Außerdem stellt er ein interaktives Testwerkzeug zur Verfügung, das es dem Benutzer ermöglicht, Haltepunkte im Programm zu setzen und es schrittweise auszuführen. Es gibt auch Kommandos, um Registerinhalte oder Hauptspeicherzellen anzuzeigen und zu ersetzen.

Der XMON/23 ist kompatibel zum XMON/F Level 3 der 6.610.

5.1 Kommandos des Monitors

Der Monitor wird normalerweise angesprungen,

- wenn die Netzspannung eingeschaltet wird und das Betriebssystem nicht geladen werden kann, z.B. weil keine Datei namens SYSTEM auf der Diskette oder Festplatte existiert. Zuvor muß eine (beliebige) Taste gedrückt werden, da der Monitor zunächst auf das Einlegen einer System-Diskette wartet. Nach Einschaltung der Netzspannung wird der (RAM-) Speicher von 2000H bis FFFFH auf Null gesetzt, außerdem wird eine Checksummen-Prüfung des Monitors durchgeführt. Gegebenenfalls wird Fehlermeldung 3C ausgegeben.

- wenn während des Programmablaufes ein Haltepunkt entdeckt wird (Programm läuft im DEBUG-Modus). Der Programmzeiger zeigt die Adresse des Haltepunktes an, ebenso werden die Inhalte der Register angezeigt.

- wenn der RESET-Knopf gedrückt wird. Anschließend werden die Inhalte der Register angezeigt, der Programmzeiger steht auf Null.

- wenn ein Sprung auf Adresse Null ausgeführt wird.

- wenn das Quitt-Zeichen verwendet wird (s. SYS)

Nach Ansprung des Monitors meldet er sich mit ".".

Folgende Kommandos stehen für den Test von Assemblerprogrammen im DEBUG-Modus zur Verfügung:

S Lesen oder Ändern eines Speicherplatzes

X Lesen oder Ändern eines Registers

G Starten eines Programmes (mit oder ohne Haltepunkte)
Tastaturpuffer gelöscht ab 6 FD00

D Ausgabe eines Speicherbereiches auf den Bildschirm

F Füllen eines Speicherbereiches mit einem Wert

L Laden eines Programmes von Diskette bzw. Festplatte

R Laden des BS1MP von Diskette bzw. Festplatte

T Funktion als Teletype

H Ausgabe des Monitor-Gerätestandes

B Setzen von Basisregistern

Wenn bei einem Kommando die Eingabe einer hexadezimalen Adresse oder eines ebensolchen Wertes angefordert wird, kann eine beliebige Anzahl von hexadezimalen Ziffern (0 - F) eingegeben werden. Werden bei der Angabe einer Zahl mehr Zeichen als benötigt angegeben, so werden nur die zuletzt eingegebenen gewertet.

Um ein Kommando rückgängig zu machen, gibt man " -" ein. Die Zeile wird gelöscht, danach ist der Monitor wieder für neue Eingaben bereit. Abweichend hiervon kann man im L-Kommando mit der Taste " -" den Cursor positionieren, ohne die Zeile zu löschen.

Wenn der Monitor einen falschen Kommandobuchstaben empfängt, antwortet er zunächst mit "*", dann mit dem Eingabezeichen ".".

In den Monitorkommandos können Adressen in unterschiedlicher Weise angegeben werden:

- als vierstellige hexadezimale Zahl, z.B. 1234 oder FFFF
- als vierstellige hexadezimale Zahl plus einem Basisregister, z.B. 1234.B0. Die tatsächliche Adresse ist die Summe 1234+(B0), (B0) bedeutet hier: Inhalt von Basisregister B0.
- als Basisregister plus Offset, z.B.: .B1+1234. Wiederum ist die Adresse 1234+(B1)
- als Summe zweier Basisregister, z.B.: .B1.B2. Die tatsächliche Adresse ist dann (B1)+(B2)

In allen Fällen kann das B weggelassen werden, statt 1234.B0 kann auch 1234.0 angegeben werden.

Ist ein Teilkommando (mit CR oder SP) abgeschlossen, wird die effektive Adresse ausgegeben.

Es enthalte B1 den Wert 1000H, dann wird

G.1+1230 durch Eingabe von CR zu G.1+1230=2230

Die Basisregister B0 bis B7 werden mit dem Kommando B (s.u.) gesetzt. Sie können nur im Zusammenhang mit den Monitorkommandos benutzt werden.

5.1.1 S-Kommando

Ausgeben und/oder Ändern eines Speicherplatzes

Syntax: S<Adresse> SP

Der Inhalt der adressierten Speicherstelle wird ausgegeben. Zur Ausgabe der nächsten Speicherstelle(n) wird SP eingeben.

Zur Änderung der angezeigten und anschließenden Ausgabe der nächsten Speicherstelle:

<Wert> SP

Rückkehr zum Eingabezeichen ".":

CR

Zur Änderung der angezeigten Speicherstelle und Rückkehr zum Eingabezeichen:

<Wert> CR

<Wert> ist danach der neue Inhalt der Speicherstelle.

5.1.2 X-Kommando

Ausgeben und/oder Ändern eines Registers

Syntax: X <Register-Identifikation>

Die Register haben folgende Identifikationen:

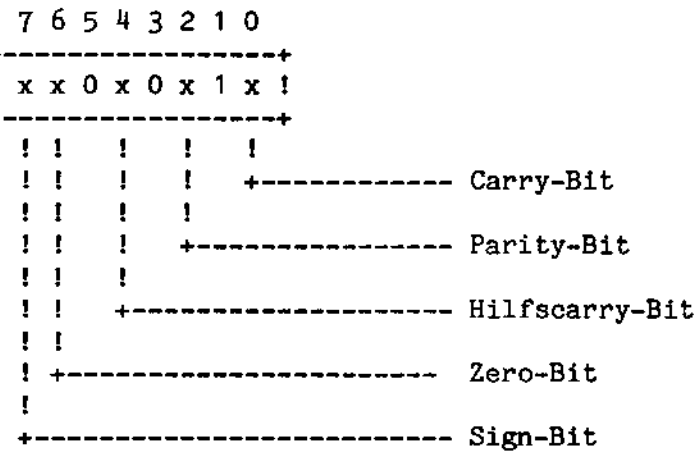
A, B, C, D, E, H, L	: Einzelregister
F	: Statusregister
P	: Programmzähler
S	: Kellerzeiger (Stackpointer)
X	: Alle Register

Der Inhalt des ausgewählten Registers wird angezeigt. Rückkehr zum Eingabezeichen: CR

Ändern eines Register-Inhalts: <Hex.Wert> CR

Wird statt CR SP eingegeben, so wird der nächste Registerinhalt angezeigt.

Stackzeiger (S) und Programmzähler (P) werden als Zwei-Byte-Kombination gesetzt. Das F-Register (Status-Flip-Flops) enthält die Zustandsbits in folgender Ordnung:



Mit XF werden die einzelnen Flags folgendermaßen dargestellt:

S = Vorzeichen (Sign) 0 = positiv

Z = Null (Zero) 1 = Null

A = Hilfscarry 1 = gesetzt

P = Parität 1 = gerade Parität

C = Carry 1 = gesetzt

F = Flagregister gesamt

5.1.3 G-Kommando

Starten eines Programms

Syntax:
```
G [CR]
G <Adresse 1> [CR]
G [SP] <Adresse 2> [CR]
G <Adresse 1> [SP] <Adresse 2> [CR]
G [SP] <Adresse 2> [SP] <Adresse 3> [CR]
G <Adresse 1> [SP] <Adresse 2> [SP] <Adresse 3> [CR]
```

<Adresse 1> ist die gewünschte Startadresse. Wenn sie nicht angegeben wird (wie in der ersten, dritten und fünften Form des Befehls), wird die aktuelle Adresse - auf die der Programmzähler zeigt - genommen, z.B. beim Wiederstart nach einem Haltepunkt.

<Adresse 2> und <Adresse 3> sind Haltepunkte. Sie werden auf die eingegebenen Adressen gesetzt.

Wenn ein Haltepunkt erreicht wird, werden beide zurückgesetzt. Die Haltepunkte werden ebenfalls zurückgesetzt, wenn das Programm durch den RESET-Knopf unterbrochen wird.

Zum Setzen eines Haltepunktes ersetzt der Monitor den Befehl im Anwenderprogramm durch einen RST 0. Wurde dieser Befehl bereits vorher im Anwenderprogramm benutzt, können Verwicklungen entstehen.

5.1.4 D-Kommando

Ausgabe eines Speicherbereiches

Syntax: D<Adresse> , <Länge> [CR]

Die Ausgabe des Speicherbereiches hat das folgende Aussehen:

```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
2000 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46  0123456789ABCDEF
2010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
2020 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46  0123456789ABCDEF
  .
  .
```

Links stehen die Speicheradressen (jeweils in 16-er Abstufung), rechts daneben 16 hexadezimale Werte und in der rechten Kolonne ihre ASCII-Codierung bzw. ein Punkt, wenn es keinen entsprechenden abbildbaren ASCII-Code (Code kleiner 20H oder größer 7FH) gibt.

Der angegebene Speicherbereich wird bildschirmweise ausgegeben. Bei Eingabe eines beliebigen Zeichens außer "Q" wird der nächste Teil des Speicherauszugs ausgegeben, bei "Q" wird die Ausgabe abgebrochen.

5.1.5 F-Kommando

Füllen eines Speicherbereiches

Syntax: F<untere Adresse>,<obere Adresse>,<Wert>CR

Alle Speicherstellen, beginnend mit der unteren und endend mit der oberen Adresse, werden mit dem eingegebenen Wert belegt.

Ist <untere Adresse> größer als <obere Adresse> , so wird der Bereich oberhalb von <untere Adresse> und unterhalb von <obere Adresse> mit <Wert>gefüllt.

Beispiel: .F4000,6000,00 CR

füllt den Bereich von 4000 bis 6000 mit 00.

Achtung: Der Bereich von 0000 bis 1FFF ist schreibgeschützt und kann mit dem F-Kommando nicht gefüllt werden. Der Bereich von 2000 bis 27FF darf nicht gefüllt werden, weil sonst der Datenbereich des Monitors zerstört wird.

5.1.6 L-Kommando

Laden eines binären Programmes

Syntax: L{:Hn: / :Dn: / :Fn:} <Dateiname> CR

n ist die Laufwerknummer (0 - 3 bei Disketten, 0 und 1 bei Festplatte).

Der Dateiname kann gebildet werden, wie es in Kapitel 1.2.1 für INTEL- und BS1MP-Dateien beschrieben wird.

Nach Durchführung des Aufrufs ist das unter <Dateiname> gespeicherte Programm geladen, mit dem Kommando .G CR kann es gestartet werden.

5.1.7 R-Kommando

Laden des BS1MP

Syntax: R {:Hn: / :Dn: / :Fn:}

Dieses Kommando ist eine Abkürzung für

L {:Hn: / :Dn: / :Fn:} SYSTEM CR

n ist dabei die Laufwerknummer.

SYSTEM ist dabei der Name des Programmes, welches das BS1MP nachlädt.

5.1.8 T-Kommando

Teletype-Funktion

Syntax: T CR

Funktion off-line, d.h. ohne ein an die Asynchronschnittstelle angeschlossenes Gerät:

Jedes über die Tastatur eingegebene Zeichen wird auf dem Bildschirm abgebildet.

Den Teletype-Betrieb kann man durch Betätigen des Resetschalters wieder verlassen.

Funktion on-line:

Über die Tastatur eingegebene Zeichen werden asynchron zum angekoppelten Gerät übertragen. Ankommende Zeichen werden auf dem Bildschirm ausgegeben.

Die Übertragung findet ohne ein höheres Protokoll statt.

5.1.9 H-Kommando

Ausgabe des Gerätestandes:

Syntax: H [CR]

Ausgegeben wird der Monitor-Gerätestand in der Form

*** XMON/23 Resident Monitor XXXXXX.Y ***

Die eigentliche Gerätestandsnummer ist Y.

Bei Aufruf des Kommandos wird außerdem eine Checksummenprüfung des Monitors durchgeführt. Bei Auftreten eines Checksum-Fehlers wird Meldung 3C ausgegeben.

5.1.10 B-Kommando

Setzen eines Basisregisters

Syntax: .B <Basisregister-Nummer> [SP] <alte Adresse> (wird ausgegeben) <neue Adresse> [CR]

oder .B <Basisregister-Nummer> [SP] <alte Adresse> (wird ausgegeben) [CR]

Durch das Kommando können die acht Basisregister B0 bis B7 mit Adreß-Werten versorgt werden. Sie können dann in anderen Monitorkommandos bei der Adreßbezeichnung benutzt werden.

Die zweite Form des Kommandos dient zur Ausgabe des aktuellen Basisregister-Inhaltes.

5.2 Vom Anwender aufrufbare Routinen

Der Monitor enthält einige dem Benutzer zugängliche Routinen.

Nach dem Rücksprung aus den unten ausgeführten Routinen ist das Carry-Bit gesetzt, wenn ein Fehler aufgetreten ist. In diesem Fall steht der Fehlercode im Akkumulator A. Die Bedeutungen der Fehlercodes sind im Anhang "Systemfehlermeldungen" aufgelistet.

Registerinhalte werden durch die Monitorroutinen nicht verändert, sofern in ihnen nicht Ergebnisse übergeben werden.

Folgende Routinen stehen zur Verfügung:

TTI (40H)
Lesen eines Zeichens von der Tastatur. Die Routine bleibt im Wartezustand, bis ein Zeichen eingegeben wird.

Das Zeichen wird im A-Register abgeliefert.

Hinweis: Dieser Routine ist die Routine INCHAR (FD09H) des Betriebssystems vorzuziehen. Bei Verwendung von INCHAR ist die Umstellung der Eingabe von der Tastatur auf eine Eingabedatei problemlos möglich.

TTINW (DCH)
Wie TTI, jedoch kehrt die Routine sofort zu dem aufgerufenen Programm zurück, selbst wenn kein Zeichen über Tastatur eingegeben wurde. In diesem Fall wird das Carry-Bit gesetzt.

TTO (43H)
Schreiben eines Zeichens auf den Bildschirm an der momentanen Cursor-Position. Das zu schreibende Zeichen muß im A-Register stehen. Der Cursor wandert eine Position weiter.

Codes kleiner oder gleich 31 (1FH) werden als Kontrollzeichen interpretiert, geben also kein abbildbares Zeichen auf dem Bildschirm aus.

TTONC (46H)
Wie TTO, jedoch werden alle Zeichen (auch Kontrollzeichen) als Datenzeichen interpretiert. Die semigraphischen Zeichen (01-1FH und 80-FFH) im Bedienzustand ATTR23 können nur mit TTONC ausgegeben werden.

Ist der Bedienzustand ATTR21 eingestellt, werden Codes kleiner oder gleich 1FH und größer 7FH als Leerzeichen dargestellt.

CURSOR (49H)
Positionierung des Cursors auf eine gewünschte Stelle

H - gewünschte Zeile (0 - 24)
L - gewünschte Position innerhalb der Zeile (0 - 79)

Alle außerhalb dieser Bereiche liegenden Parameter setzen den Cursor auf Zeile 0/ Spalte 0 (HOME-Position).

PRINT (52H)
Ausgeben eines (im A-Register stehenden) Zeichens auf den Drucker über Kanal B.

PRNST (D9H)
Test des Druckerzustands. Ist beim Aussprung Carry gesetzt, ist der Drucker aktiv. Wenn kein Carry gesetzt ist, ist er bereit zum Drucken. Der Anwender kann einen eigenen Treiber schreiben. Die Adresse dieses Treibers muß dann in STADR (2776H) stehen. Beim Aussprung muß Carry wie oben definiert sein.

INDISP (6AH)
Lesen des Zeichens von der Bildschirmposition, auf der der Cursor steht. Der Cursor wandert ein Feld weiter.

Das Zeichen wird in A übergeben.

RCVI (82H)
Empfang eines Zeichens über die Asynchronschnittstelle.

Die Routine wartet, bis ein Zeichen über Kanal A ankommt und liefert dieses im A-Register ab. Ankommende Zeichen werden in einem 31 Byte langen Puffer gespeichert.

Das höchstsignifikante Bit (80H) des Zeichens wird auf 1 gesetzt (Übertragung im 7-Bit-Rahmen), wenn während der Übertragung ein Fehler festgestellt wurde (Parity, Überlauf, Zeichenrahmenfehler).

Werden Zeichen im 8-Bit-Rahmen übertragen, (durch Mikroschalter einzustellen), so wird im Fehlerfall der Empfangspuffer rückgesetzt, das Carry-Bit gesetzt und im Akkumulator A Fehlercode 3DH übergeben.

RCVI8 (C4H)
Wie RCVI, jedoch werden alle 8 Bits eines Zeichens eingelesen.

RCVINW (DFH)
Wie RCVI, jedoch wartet die Routine nicht, wenn noch kein Zeichen empfangen worden ist. In diesem Fall wird das Carry-Bit gesetzt.

XMIO (CAH)
Senden eines Zeichens (8-Bit) über die Asynchronschnittstelle (Kanal A).
A: zu sendendes Zeichen.
Ist der Sendepuffer noch voll, wartet die Routine, bis das Zeichen in dem Sendepuffer abgelegt werden kann.

XMIONW (E2H)
Wie XMIO, jedoch kehrt die Routine mit gesetztem Carry zurück, wenn der Sendepuffer voll ist.

EXFUNC (D3H)
Einsprung für erweiterte Sprungtabelle für Funktionen, die später implementiert werden.
C: Funktions-Code
Die anderen Register beinhalten die nötigen Parameter. Funktions-Code 01H ist der Timer2 der 6.611. Näheres hierzu s. unter Kapitel 9.3.

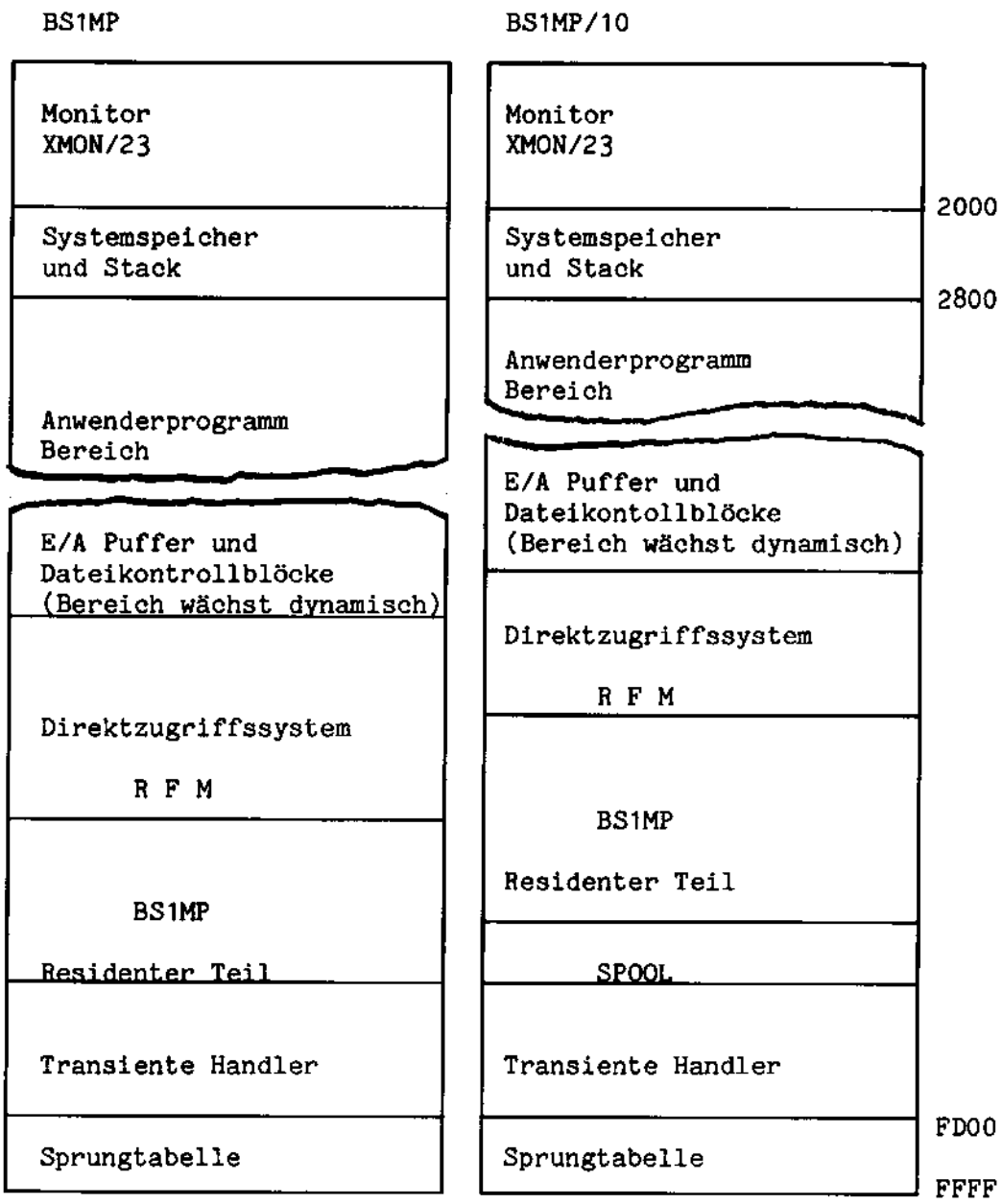
DCWRIT (106H)
Direkter Zugriff auf die Schreibfunktion des Display-Controllers.
C: Funktion-Code zwischen 00 und 7FH (s. Anhang G).
A: Schreibdaten falls nötig
Bei Rücksprung mit Carry steht in A der Fehler-Code. Fehler 42 tritt bei ungültigem Funktions-Code auf. Mit dieser Routine können Funktionen wie Software-Schalter (s. 9.6), Push-Keys und die neuen Bildschirmattribute angesprochen werden.

DCREAD (109H)
Direkter Zugriff auf die Lesefunktionen des Display-Controllers.
C: Funktions-Code zwischen 80H und FFH (s. Anhang G)
A: Daten falls nötig
Nach dem Rücksprung steht in A das gelesene Zeichen oder bei gesetztem Carry der Fehler-Code. Fehler 42 tritt bei ungültigem Funktions-Code auf. Mit dieser Routine können der aktuelle Stand der Software-Schalter (s. 9.6), der Push-Keys und der neuen Bildschirmattribute gelesen werden.

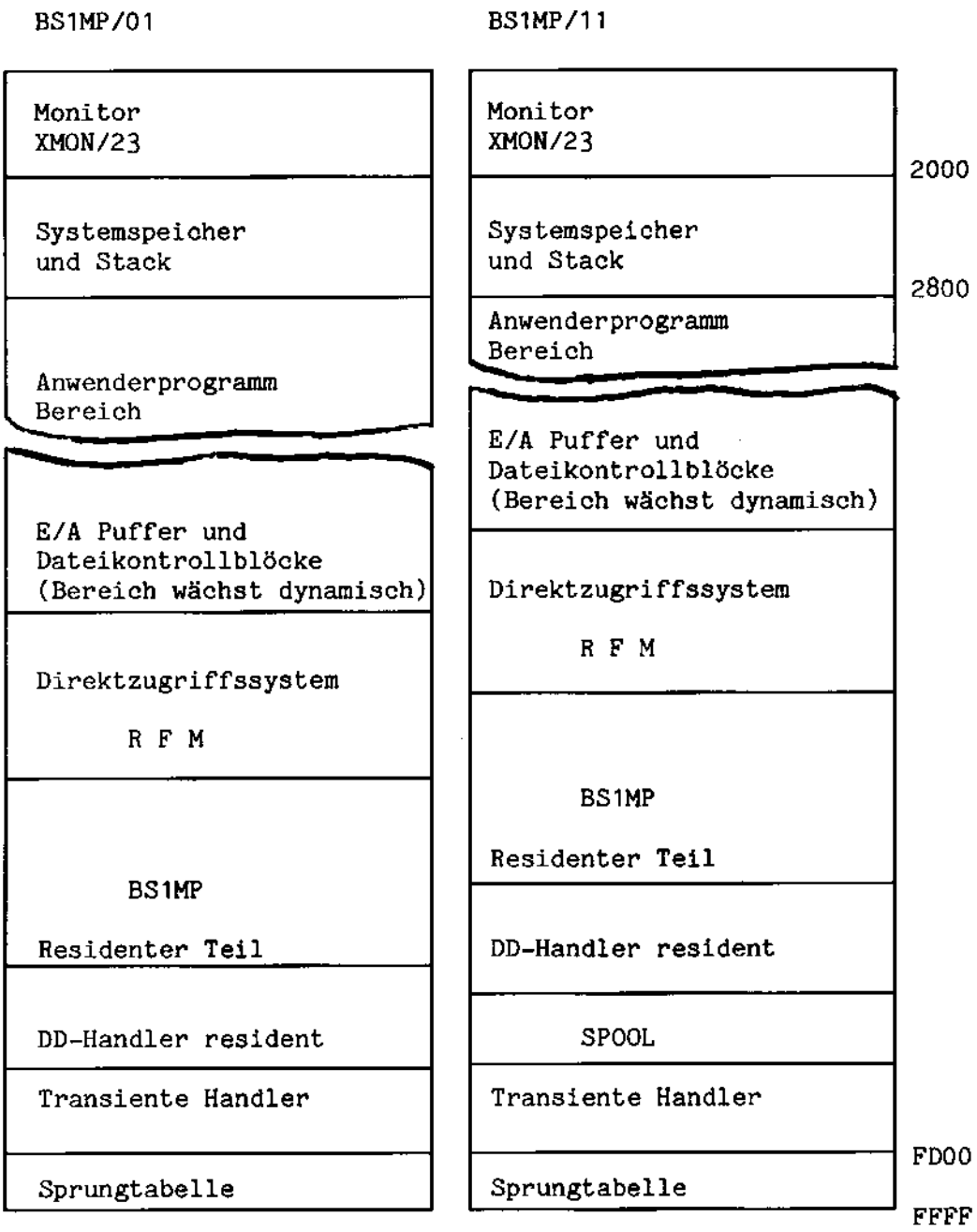
6 Programmierung mit dem BS1MP

6.1 Speicherbelegung

Es gibt vier Generiervarianten des BS1MP, die im folgenden mit BS1MP, BS1MP/01, BS1MP/10 und BS1MP/11 bezeichnet werden. Sie unterscheiden sich in der Aufteilung in residenten und transienten (überlagerbaren) Code. Figur 1 und 2 zeigen Größe, Startadresse und
Speicherbelegung der vier Varianten.



Figur 1: Speicherbelegung BS1MP und BS1MP/10



Figur 2: Speicherbelegung BS1MP/01 und BS1MP/11

BS1MP:

Im BS1MP gibt es einen Overlay-(Überlagerungs-)Bereich, in den abhängig vom Diskettenformat der jeweils benötigte Disketten- oder Festplatten-Handler (Treiberbaustein) geladen wird.

Benutzt man in einer Anwendung mehrere Formate parallel, so muß der passende Handler stets nachgeladen werden, bevor der Zugriff erfolgen kann. Hierzu muß das nachzuladende Modul auf dem System-Laufwerk bereitstehen.

Das Nachladen wird vom Betriebssystem durchgeführt, ohne daß der Anwender sich darum kümmern muß.

Die Generiervariante BS1MP wird man daher normalerweise benutzen, wenn man nur mit einem einzigen Format arbeitet bzw. eine 6.611 mit Festplatte zur Verfügung steht, wo die Nachladezeiten keine Rolle spielen.

Hinweis: Beim ersten Laden des Systems wird der Handler des System-Laufwerks geladen.

Das BS1MP mit RFM belegt ca. 17 KByte, ohne den RFM belegt es ca. 11 KByte. Die jeweilige Startadresse kann mit HELP (s. 2.14) überprüft werden.

Auf dem System-Laufwerk müssen neben dem Betriebssystem (Datei BS611) und dem Ladeprogramm (Datei: SYSTEM) während des Programmablaufs gegebenenfalls auch die Handler für die verschiedenen Formate vorhanden sein:

1 Mbyte:	TDDHAN	BS1M-Format
	THDHAN	Festplatten-Format (bei 6.611-C)
256 KByte:	TSDHAN	INTEL-Format
	ISDHAN	ECMA-54-("IBM")-Format

BS1MP/01:

Im BS1MP/01 ist der Handler (Treiberbaustein) für die BS1M-formatierte Diskette ständig resident, lediglich die Handler für die anderen Formate müssen nach Bedarf von dem System-Laufwerk nachgeladen werden.

Da das BS1MP/01 zwei Handler gleichzeitig im Hauptspeicher halten kann, ist die gleichzeitige Bedienung zweier unterschiedlicher Formate - von denen eines das BS1M-Format ist - ohne ständiges Nachladen von Systemteilen möglich. Diese Generiervariante wird man daher benutzen, wenn man z.B. ständig mit ECMA-54-Disketten (als Datenaustauschmedium) arbeitet.

Das BS1MP/01 belegt mit RFM rund 20 KByte Hauptspeicher, ohne RFM belegt es ca. 14 KByte. Die jeweilige Startadresse kann mit HELP (s. 2.14) überprüft werden.

Neben den Dateien BS611.01 (Betriebssystem) und SYSTEM (Ladeprogramm) müssen auf dem System-Laufwerk noch die beiden folgenden Handler bereitstehen:

THDHAN.01	Festplatten-Format (bei 6.611-C)
TSDHAN.01	INTEL-Format
ISDHAN.01	ECMA-54-("IBM")-Format.

BS1MP/10:

Diese Generiervariante enthält zusätzlich zum BS1MP noch die Spool-Funktion. Die Spool-Funktion belegt ca. 2K Bytes d. h. das BS1MP/10 ist jeweils 2K Bytes größer als das BS1MP. Da der SPOOL einen transienten Teil hat, müssen auf dem System-Laufwerk die Spool-Handler vorhanden sein:

THDSPL.10 (bei 6.611-C)
TDDSPL.10
TSDSPL.10

und wie oben:

BS611.10
THDHAN.10 (bei 6.611-C)
TDDHAN.10
TSDHAN.10
ISDHAN.10

BS1MP/11:

Diese Generiervariante ist eine Kombination von BS1MP/01 und BS1MP/10. Sie enthält also den Handler für das BS1M-Format resident und unterstützt die Spool-Funktion. Folgende Dateien müssen auf dem System-Laufwerk vorhanden sein:

BS611.11

THDSPL.11 (bei 6.611-C)
TDDSPL.11
TSDSPL.11

THDHAN.11 (bei 6.611-C)
TSDHAN.11
ISDHAN.11

Übersicht der Generiervarianten mit RFM:

Name	Größe in K Bytes	Kennzeichen
BS1MP	17	alle Handler transient
BS1MP/01	20	DD-Handler resident
BS1MP/10	19	transient + SPOOL
BS1MP/11	22	resident + SPOOL

6.2 Benutzung der BS1MP-Funktionen

Zwischen den Adressen FD00H und FE8AH liegt die Sprungtabelle für Systemfunktionen, die dem Anwender zugänglich sind:

FD00	BS1MP	FD03	ERROR	FD06	LOAD
FD09	INCHAR	FD0C	OUTCHAR	FD0F	GET
FD12	PUT	FD15	PUTSTR	FD18	GETBIN
FD1B	PUTBIN	FD1E	OPEN	FD21	READ
FD24	WRITE	FD27	CLOSE	FD2A	EDBH
FD2D	EDWH	FD30	EDHB	FD33	EDHW
FD36	BUFALLOC	FD39	BUFDEALLOC	FD3C	STIMER
FDDF	PHYSINT	FE2A	DELETE	FE2D	RENAME
FE54	UPCHAR	FE57	UPCASE	FE5A	UPLINE
FE5D	LDIG	FE60	NIBBLE	FE63	GETBYT
FE66	GETADR	FE69	PUTBYT	FE6C	PUTADR
FE6F	INCBYT	FE72	INCADR	FE75	DECBYT
FE78	DECADR	FE7B	TSTBYT	FE7E	TSTADR
FE81	COMPSTR	FE84	FILLSTR	FE87	COPYSTR
FE8A	IDENT				

Eine BS1MP-Funktion wird aufgerufen durch eine CALL-Anweisung mit der zugehörigen Sprungtabellen-Adresse. Vor dem CALL müssen die zugehörigen Parameter in die entsprechenden Register gebracht werden. Nach dem Rücksprung zeigt das Carrybit an, ob die Routine normal (Carry = 0) oder fehlerhaft (Carry gesetzt) abgelaufen ist. Im letzteren Fall enthält das A-Register den Fehlercode. Carry gesetzt und A = 0 bedeutet bei Eingabe Dateiende, bei Ausgabe Überlauf.

Routinen, die E/A-Operationen auf sequentiellen Dateien enthalten, verlangen in HL als Parameter die Adresse eines Dateikontrollblockes (FCB). E/A-Operationen auf Direktzugriffsdateien werden im Kapitel 7 behandelt.

Falls die normalen Dateizuweisungen (SI, SO usw.) verwendet werden, braucht sich der Anwender mit den Formaten der FCBs nicht zu beschäftigen.

Möchte man in einem Assemblerprogramm selbst Dateien eröffnen, also nicht über logische Zuweisungen im Programmaufruf gehen, so kann man sich die notwendigen FCB's mit der Routine PHYSINT (FDDFH) aufbauen. Die genauere Kenntnis der FCB- Struktur ist dazu nicht nötig (s.6.3).

Die höheren Programmiersprachen (COBOL, BASIC) wickeln den Verkehr mit sequentiellen Dateien mit ihren eigenen Sprachmitteln ab.

Im folgenden wird eine kurze Beschreibung der verfügbaren Systemfunktionen gegeben. Unter der Rubrik "Einsprung" sind jeweils die vor dem Einsprung bereitzustellenden Parameter beschrieben, unter "Aussprung" die Resultate nach Ablauf der Funktion. Wenn nicht anders angegeben, werden alle Register gerettet und wiederhergestellt mit Ausnahme derer, die anschließend ein Ergebnis enthalten.

Adresse	Bezeichnung	Funktion und Parameterversorgung
FD00	BS1MP	Funktion: Rücksprung in den Kommandomodus des BS1M. Einsprung: Mit JMP oder CALL. Normalerweise kehrt man mit RET in das Betriebssystem zurück. Ist in diesem Fall Carry gesetzt, muß A einen Fehlercode enthalten, der dann vom Betriebssystem angezeigt wird, andernfalls ist kein Parameter erforderlich. Aussprung: Keine Rückkehr zum Anwender. Über logische Einheiten eröffnete Dateien werden automatisch geschlossen, ausgenommen solche, die mit ASSIGN eröffnet wurden.
FD03	ERROR	Funktion: Gibt eine Fehlermeldung ERROR XX und - falls in der Systemfehlerdatei vorhanden - einen Fehlertext auf CO aus. Rücksprung ins Anwenderprogramm, falls das Carrybit beim Ansprung nicht gesetzt war, sonst Sprung in das Betriebssystem. Einsprung: A: Fehlercode
FD06	LOAD	Funktion: Lädt ein Programm von einer INTEL- oder BS1M-formatierten Diskette bzw. von Festplatte. Einsprung: A = Laufwerk-Nummer, HL = Adresse eines neun Zeichen langen Namenfeldes. Der Name muß mit Binär-Null auf sechs Zeichen aufgefüllt werden, wenn er kürzer ist; der Punkt einer Namenserweiterung muß weggelassen, die Namenserweiterung mit Binär-Null auf drei Zeichen aufgefüllt werden.

Adresse	Bezeichnung	Funktion und Parameterversorgung

Beispiel: Name = "ART.1"

Darstellung im Assemblerprogramm:

```
db  'ART',0,0,0,'1',0,0
```

Bei Rückkehr aus der Routine steht in HL die Startadresse des Programms.

Die folgenden E/A-Routinen für sequentielle Dateien (bis einschließlich OPEN) verlangen die Adresse eines Dateikontrollblocks (FCB), der durch die Zuweisung logischer Einheiten im Programmaufruf angelegt wird aber auch vom Anwender selbst eingerichtet werden kann.

Folgende Abkürzungen werden benutzt:

EOE = Ausgabedateiende. (Ende des vorgesehenen Dateibereichs erreicht).
EOF = Ende der Eingabedatei
EOT = Textende

Adresse	Bezeichnung	Funktion und Parameterversorgung

FD09 INCHAR

Funktion:
Einlesen eines Zeichens aus einer Datei bzw. von einem Gerät (abhängig von Dateikontrollblock)

Einsprung:
HL = Adresse des Dateikontrollblocks (FCB).

Aussprung:
Das eingelesene Zeichen ist in A. Falls nach Rücksprung Carry gesetzt, ist in A der Fehlercode oder es ist EOF aufgetreten (A=0). Einzelheiten der Arbeitsweise von INCHAR (abhängig von zwei Bit-Schaltern im FCB) befinden sich im Anhang E.

FD0C OUTCHAR

Funktion:
Schreiben eines Zeichens in eine Datei bzw. auf ein Gerät (abhängig vom Dateikontrollblock).

Einsprung:
HL = Adresse des Dateikontrollblocks (FCB).
A = zu schreibendes Zeichen.

Adresse	Bezeich-nung	Funktion und Parameterversorgung
		Aussprung: Falls Carry nach Rücksprung gesetzt ist, ist in A der Fehlercode bzw. es ist EOE aufgetreten (A=0). Einzelheiten der Arbeitsweise von OUTCHAR (abhängig von zwei Bit-Schaltern im FCB) befinden sich im Anhang E.
FDOF	GET	Funktion: Einlesen von Zeichen bis zum ersten Auftreten von CR (0DH) Einsprung: HL = Adresse des Dateikontrollblocks (FCB). DE = Adresse eines Benutzerpuffers, in den die eingelesenen Zeichen abgelegt werden. Aussprung: Ist nach Rücksprung Carry gesetzt, so ist in A der Fehlercode bzw. es ist EOT aufgetreten (A=0). DE ist nach dem Rücksprung verändert.
FD12	PUT	Funktion: PUT schreibt einen Satz bis zum ersten Auftreten von CR einschließlich und hängt LF (0AH) an. Einsprung: HL = Adresse des Dateikontrollblocks (FCB). DE = Adresse eines Benutzerpuffers, aus dem die Zeichen ausgelesen und weggeschrieben werden. Aussprung: Ist nach Rücksprung Carry gesetzt, so ist der Fehlercode in A bzw. es ist EOE (A=0) aufgetreten. DE ist nach Rücksprung verändert.
FD15	PUTSTR	Wie PUT, jedoch wird weder CR noch LF am Ende angehängt.
FD18	GETBIN	Funktion: Lesen einer bestimmten Zahl von Zeichen. Einsprung: HL = Adresse des Dateikontrollblocks (FCB). DE = Adresse des Satzpuffers. B = Zeichenzähler. In B wird angegeben, wieviel Zeichen in den Puffer eingelesen werden sollen.

Adresse	Bezeichnung	Funktion und Parameterversorgung

Aussprung:
Bei gesetztem Carry nach dem Rücksprung steht in A der Fehlercode, bzw. es ist EOF eingetreten (A=0). DE steht auf der Adresse des letzten eingelesenen Zeichens + 1, B ist null, außer, vor Einlesen der angegebenen Anzahl von Zeichen trat EOF auf. In diesem Fall enthält B die Anzahl der nicht mehr gelesenen Zeichen.

FD1B PUTBIN

Funktion:
Schreiben einer bestimmten Zahl von Zeichen.

Einsprung:
Wie GETBIN.

Aussprung:
Bei gesetztem Carry nach dem Rücksprung steht in A der Fehlercode bzw. ist EOE aufgetreten (A=0). DE steht auf der Adresse des letzten geschriebenen Zeichens + 1, B ist 0, außer es trat vor Herausschreiben aller Zeichen EOE auf. In diesem Fall enthält B die Anzahl der noch zu schreibenden Zeichen.

FD1E OPEN

Funktion:
Öffnen einer sequentiellen Datei.

Einsprung:
HL = Adresse des Dateikontrollblocks (FCB).
A = Open-Code

Open-Code = 1 Öffnen für Eingabe
Open-Code = 2 Öffnen für Ausgabe

Aussprung:
Bei gesetztem Carry steht in A der Fehlercode.

FD21 READ

Funktion:
Einlesen eines Sektors *) in den FCB-Puffer.

Einsprung:
HL = Adresse des Dateikontrollblocks (FCB).

Aussprung:
Bei gesetztem Carry steht in A der Fehlercode oder es wurde EOF erreicht (A=0).

*) 128 Byte bei 256 KByte-, 256 Byte bei 1 MByte-Formaten.

Adresse	Bezeichnung	Funktion und Parameterversorgung

FD24 WRITE

Funktion:
Es wird ein Sektor *) aus dem FCB-Puffer geschrieben.

Einsprung:
HL = Adresse des Dateikontrollblocks (FCB).

Aussprung:
Bei gesetztem Carry steht in A der Fehlercode bzw. wurde EOE erreicht (A=0).

FD27 CLOSE

Funktion:
Durch CLOSE werden bestimmte Informationen ins Inhaltsverzeichnis der Diskette geschrieben. Schließt man eine nicht logischen Einheiten zugewiesene Ausgabedatei bei Pogrammende nicht mit CLOSE, gehen sämtliche Daten verloren. Dateien, die über die Zuweisung zu logischen Einheiten eröffnet wurden, dürfen nicht mit CLOSE geschlossen werden, da dies automatisch beim Rücksprung in Betriebssystem geschieht.

Einsprung:
HL: Adresse des Dateikontrollblocks (FCB).

Aussprung:
Bei gesetztem Carry Fehlercode in A.

FDDF PHYSINT

Funktion:
Die Routine legt einen FCB an und reserviert die notwendigen Pufferbereiche.

Einsprung:
HL = Adresse einer Zeichenfolge die den Dateinamen enthält, abgeschlossen mit einem CR.
A = 0 - Eingabedatei, 1 - Ausgabedatei.

Beispiele für solche Zeichenfolgen:

':F1:DAT' ':HO:DAT.ABC' 'KUNDEN'

Beispiel:

```
PHYSINT   equ       OFDDFH
          .
          lxi       h,dat
          mvi       a,1       ;Ausgabedatei
          call      PHYSINT
          .
dat:      db        ':HO:DAT.ABC',ODH
```

*) 128 Byte bei 256 KByte-, 256 Byte bei 1 MByte-Formaten

Adresse	Bezeichnung	Funktion und Parameterversorgung
		Aussprung: Ist Carry nach Rückkehr aus der Routine gesetzt, steht in A ein Fehlercode. Ist Carry nicht gesetzt, enthält HL die Adresse des FCB's, der von PHYSINT angelegt wurde.
FD2A	EDBH	Funktion: Aufbereitung eines 1-Byte-Hex.-Wertes in ASCII. Einsprung: A = umzuwandelndes Byte HL = Adresse eines 2-Byte-Empfangfeldes Aussprung: ASCII-Zeichen im Empfangsfeld. HL ist um zwei erhöht. Beispiel: Das Byte 4AH wird in 34H 41H umgewandelt, kann jetzt also auf dem Bildschirm dargestellt werden.
FD2D	EDWH	Funktion: Aufbereitung eines 2-Byte-Hex.Wertes in ASCII. Einsprung: DE = umzuwandelndes Wort HL = Adresse eines 4-Byte-Empfangfeldes Aussprung: ASCII-Zeichen im Empfangsfeld. HL ist um 4 erhöht.
FD30	EDHB	Funktion: Umkehrung von EDBH. Einsprung: HL = Adresse eines 2-Byte-Bereichs Aussprung: Hex. Wert in A. HL ist um 2 erhöht. Ist Carry gesetzt, enthält A Fehlercode 15H (andere als die Zeichen 0-9, A-F können nicht umgesetzt werden, kleine Buchstaben werden nicht akzeptiert).

Adresse	Bezeichnung	Funktion und Parameterversorgung
FD33	EDHW	Funktion: Umkehrung von EDWH. Einsprung: HL = Adresse eines 4-Byte-Bereichs Aussprung: Hex.Wert in DE. HL ist um 4 erhöht. Ist Carry gesetzt, enthält A Fehlercode 15H (andere als die Zeichen 0-9, A-F können nicht umgesetzt werden, kleine Buchstaben werden nicht akzeptiert).
FD36	BUFALLOC	Funktion: Die Routine reserviert und übergibt Speicherbereiche unterhalb des Betriebssystems. Diese Bereiche werden vom Betriebssystem auch dann nicht überschrieben, wenn es z.B. für die Anlage von FCB's Puffer anlegt. Einsprung: HL = gewünschte Puffergröße in Bytes Aussprung: HL = Pufferadresse.
FD39	BUFDEALLOC	Funktion: Die durch BUFALLOC belegten Speicherbereiche werden von der Routine freigegeben. Da keine echte Freispeicherverwaltung vorgenommen wird, wird der zurückgegebene Speicher nur dann wieder neu verwendet, wenn alle später angelegten Puffer zuvor wieder freigegeben wurden. Einsprung: HL = Pufferadresse Aussprung: Der Pufferbereich wird an das Betriebssystem zurückgegeben.
FD3C	STIMER	Einsprung: DE = Gewünschtes Zeitintervall in 20ms-Einheiten HL = Adresse, die nach Ablauf des Zeitintervalles angesprungen wird. Siehe auch Abschnitt 9.3.

Adresse	Bezeichnung	Funktion und Parameterversorgung
FE2A	DELETE	Funktion: Löschen einer Datei Einsprung: A: Laufwerksnummer B: Format ('F','D','J','H') HL: Adresse des Dateinamens (Darstellung des Namens wie bei LOAD beschrieben) Aussprung: Wenn Carry gesetzt, Fehler (Code in A).
FE2D	RENAME	Funktion: Umbenennen einer Datei Einsprung: A: Laufwerknummer B: Format ('F','D','J','H') HL: Adresse des alten (gültigen) Dateinamens DE: Adresse des neuen Namens (Formate der Namen wie unter LOAD) Aussprung: Wenn Carry gesetzt, Fehler (Code in A).
FE54	UPCHAR	Funktion: Umwandlung eines kleinen in einen Großbuchstaben Einsprung: A: ASCII-Buchstabe Aussprung: A: ASCII-Buchstabe groß geschrieben Carry gesetzt -- Zeichen kein Kleinbuchstabe. Keine Umwandlung.
FE57	UPCASE	Funktion: Umwandeln eine Buchstabenfolge in Großbuchstaben. Andere Zeichen bleiben unverändert. Einsprung: B: Zeichenzähler HL: Adresse einer Zeichenfolge
FE5A	UPLINE	Funktion: Umwandlung eines (mit CR abgeschlossenen) Buchstabenstrings in Großbuchstaben. Andere Zeichen werden nicht umgewandelt. Einsprung: HL: Adresse einer Zeichenfolge

Adresse	Bezeichnung	Funktion und Parameterversorgung
FE5D	LDIG	Funktion: Prüfen, ob Code ASCII-Buchstabe oder Ziffer ist. Einsprung: A: Code Aussprung: Wenn Carry gesetzt, dann ist Code weder Ziffer noch Buchstabe
FE60	NIBBLE	Funktion: Laden einer ASCII-Ziffer in A und Umwandeln in hexadezimale Darstellung. HL wird um 1 erhöht. (Zulässig sind die Zeichen 0-9 und A-F). Einsprung: HL: Adresse einer ASCII-Ziffer Aussprung: A: ASCII-Ziffer als hexadezimaler Wert HL: um 1 erhöht Ist Carry gesetzt, so steht in A Fehlercode 15H (unzulässiger Eingabewert).
FE63	GETBYT	Funktion: Laden einer Speicherzelle (Adressierung mit Offset) Einsprung: HL: Basisadresse DE: Offset Aussprung: A: Inhalt der Speicherzelle (HL+DE)
FE66	GETADR	Funktion: Laden einer Adresse aus dem Speicher (Adressierung mit Offset) Einsprung: wie GETBYT Aussprung: BC: Inhalt der Speicherzellen (HL+DE) in (C),(HL+DE)+1 in (B)
FE69	PUTBYT	Funktion: Abspeichern eines Zeichens (Adressierung mit Offset) Einsprung: A: abzuspeicherndes Zeichen HL: Basisadresse DE: Offset Aussprung: A: abgespeichert auf (HL+DE)

Adresse	Bezeichnung	Funktion und Parameterversorgung
FE6C	PUTADR	Funktion: Abspeichern einer Adresse (Adressierung mit Offset) Einsprung: BC: abzuspeichernde Adresse HL: Basisadresse DE: Offset Aussprung: BC abgespeichert auf (HL+DE) und (HL+DE)+1
FE6F	INCBYT	Funktion: Inkrementieren eines Wertes im Speicher (Adressierung mit Offset) Einsprung: wie GETBYT Aussprung: Byte auf (HL+DE) um 1 erhöht A verändert! Carry gesetzt : Byte auf (HL+DE)=FFH
FE72	INCADR	Funktion: Inkrementieren einer Adresse im Speicher (Adressierung mit Offset) Einsprung: wie GETBYT Aussprung: Adresse auf (HL+DE),(HL+DE)+1 um 1 erhöht DE verändert! Carry gesetzt : Überlauf (Adresse war FFFFH)
FE75	DECBYT	Funktion: Dekrementieren eines Bytes im Speicher (Adressierung mit Offset) Einsprung: wie GETBYT Aussprung: Byte auf (HL+DE) um 1 verringert Carry gesetzt : Byte war 00H
FE78	DECADR	Funktion: Dekrementieren einer Adresse im Speicher (Adressierung mit Offset) Einsprung: wie GETBYT Aussprung: Adresse auf (HL+DE),(HL+DE)+1 um 1 verringert DE verändert! Carry gesetzt : Überlauf (Adresse war 0000H)

Adresse	Bezeichnung	Funktion und Parameterversorgung

FE7B TSTBYT

Funktion: Vergleichen eines Bytes mit einem Speicherwert (Adressierung mit Offset)

Einsprung:
HL: Basisadresse
DE: Offset
A: Vergleichswert

Aussprung:
A < Wert auf (HL+DE) → Carry gesetzt
A = Wert auf (HL+DE) → Zero-Bit gesetzt
A > Wert auf (HL+DE) → Carry = 0

Achtung: A enthält veränderten Wert.

FE7E TSTADR

Funktion: Vergleich einer Adresse mit 2-Byte-Wert im Speicher (Adressierung mit Offset)

Einsprung:
HL: Basisadresse
DE: Offset
BC: Vergleichsadresse

Aussprung:
BC < Adresse auf (HL+DE),(HL+DE)+1 → Carry gesetzt
BC = Adresse auf (HL+DE),(HL+DE)+1 → Zero-Bit gesetzt
BC > Adresse auf (HL+DE),(HL+DE)+1 → Carry = 0

FE81 CMPSTR

Funktion: Vergleich zweier Zeichenfolgen

Einsprung:
HL: Adresse des ersten Zeichenfolge
DE: Adresse des zweiten Zeichenfolge
BC: Länge der Zeichenfolge

Aussprung:
Sind die Zeichenfolgen gleich, ist
Carry = 0,
sonst Carry gesetzt.

FE84 FILLSTR

Funktion: Füllen eines Speicherbereiches mit einem Zeichen

Einsprung:
A: Füllkonstante
HL: Anfangsadresse des Speicherbereiches
BC: Länge des Speicherbereiches

Adresse	Bezeichnung	Funktion und Parameterversorgung
FE87	COPYSTR	Funktion: Übertragen einer Zeichenfolge in einen anderen Speicherbereich Einsprung: HL: Adresse des Quellstrings DE: Zieladresse BC: Länge des Quellstrings
FE8A	IDENT	Funktion: Prüfen eines Laufwerks/einer Diskette bzw. Festplatte. Es kann festgestellt werden, ob ein Laufwerk noch offen ist, ob die Diskette vom vermuteten Format oder noch unformatiert ist. Einsprung: A: Laufwerknummer B: Format ('F','D','J','H') Aussprung: Wenn Carry gesetzt, steht der Fehlercode in A.

6.3 Dateikontrollblöcke (FCB's)

Für jede sequentielle Datei, die eröffnet werden soll, muß (in Assemblerprogrammen) ein Dateikontrollblock (FCB) angelegt werden. Dies kann auf zwei Arten geschehen:

- durch Zuweisung von Dateien zu den verschiedenen logischen Einheiten bei Programmaufruf (z.B. SI, AI bei Eingabe-, SO, AO, AL oder SL bei Ausgabedateien). In diesen Fällen wird vom Betriebssystem ein FCB aufgebaut und die entsprechende Datei eröffnet. Bei Eingabedateien wird der erste Sektor in den FCB-Datenpuffer eingelesen. Die Adresse des FCB im Speicher wird vom BS1MP in den folgenden, für diesen Zweck reservierten Zellen abgelegt:

FF00	CI
FF02	CO
FF04	SI
FF06	SO
FF08	SL
FF0A	AI
FF0C	AO
FF0E	AL

FF10	I0
FF12	I1
FF14	I2
FF16	I3
FF18	O0
FF1A	O1
FF1C	O2
FF1E	O3
FF20	L0
FF22	L1
FF24	L2
FF26	L3
FF28	RP

Ist einer logischen Einheit keine Datei (bzw. Gerät) zugeordnet, so stehen statt einer FCB-Adresse binäre Nullen in den Speicherzellen.

Verläßt man das Anwenderprogramm mit RET oder JMP BS1MP, um in den Kommandomodus des Betriebssystems zurückzukehren, so wird die Datei automatisch geschlossen und die Zuweisung rückgängig gemacht, sofern sie nicht mit ASSIGN vorgenommen wurde.

- durch Aufruf der Routine PHYSINT (FDDFH). Das richtige Format des FCB leitet die Routine aus dem Dateinamen ab. Alle Ein- und Ausgabeoperationen können jetzt über Systemaufrufe bei Angabe der FCB-Adresse durchgeführt werden.

Ein FCB enthält u.a. Informationen über die Datei (Name, Format des Datenträgers), die internen Systemroutinen, die zur Behandlung des jeweiligen FCB-Typs benötigt werden sowie Pufferadressen.

FCB's für Festplatten- und für INTEL- und BS1M-formatierte Dateien haben zwei Pufferadressen, eine für einen Datenpuffer (INTEL: 128 Bytes, BS1M-Format und Festplatte: 256 Bytes), die zweite für einen Puffer, in dem ein Kettungssektor enthalten ist. Seine Größe entspricht dem des Datenpuffers.

FCB's für Dateien auf ECMA-54-("IBM")-formatierten Disketten haben nur einen Pufferverweis, der Puffer ist 128 Bytes lang.

Die FCB's sind inklusive Pufferbereiche daher ca. 550 Bytes (BS1M-Format und Festplatte), 300 Bytes (INTEL-Format) bzw. 160 Bytes (ECMA-54-Format) groß.

Der für FCB's benötigte Speicher wird vom Betriebssystem bei Anlage des Kontrollblocks reserviert. Durch Schließen der Datei (CLOSE) wird der FCB-Bereich wieder freigegeben. Für den Anwender steht er jedoch nur dann zur Verfügung, wenn es der letzte zuvor reservierte Speicher war oder alle später reservierten Puffer wieder freigegeben wurden.

7 Direktzugriffssystem RFM

7.1 Allgemeines

Der RFM ist ein Modul des Betriebssystems BS1MP.

Das Dateiverwaltungssystem RFM ermöglicht den Zugriff zu Direkt- und Indexdateien sowohl auf INTEL- als auch auf BS1M-formatierten Disketten. Direktzugriffsdateien, die mit RFM-Versionen 1.5 oder kleiner unter dem Betriebssystem BS1 der 6.610 auf INTEL-Disketten angelegt wurden, können unter dem BS1MP nicht behandelt werden. Mit Hilfe des Dienstprogrammes UMRFM können solche Dateien jedoch in "BS1MP-Dateien" umgewandelt werden, dabei ist auch die Übertragung auf BS1M-formatierte Disketten möglich, die umgewandelten Dateien können sich jedoch auch auf INTEL-Disketten befinden.

Benötigt man den RFM nicht, so kann dessen Hauptspeicherplatz für den Anwender zur Verfügung gestellt werden (RELRFM).

In diesem Kapitel wird die Schnittstelle des RFM beschrieben, wie sie sich einem Assemblerprogrammierer bietet, außerdem werden die mit dem RFM zusammenhängenden Dienstprogramme UMRFM, RELRFM, SORT.RFM, RCOVER und REORG sowie das Schnittstellenmodul RFMMOD.REL dokumentiert.

In COBOL entspricht die Schnittstelle zum Dateiverwaltungssystem dem ANSI-Standard (mit Ausnahme des Zugriffs über teilqualifizierten Schlüssel) und ist daher Teil der jeweiligen Sprachbeschreibung.

Um in Assemblerprogrammen auf den RFM-Baustein zugreifen zu können, muß das Schnittstellen-Modul RFMMOD.REL hinzugebunden werden, das die jeweiligen Aufrufe an den RFM weiterleitet. Durch das Zwischenschalten dieses Modules wird erreicht, daß die Anwenderschnittstelle des RFM unter dem BS1 der 6.610 und unter dem BS1MP nahezu gleich ist, einzige Ausnahme ist der größere Pufferbedarf unter dem BS1MP.

7.1.1 Datei-Arten

Es gibt zwei Arten von RFM-(=Direktzugriffs-)Dateien, sogenannte direkte (oder Direkt-)-Dateien und indizierte (oder Index-)Dateien. Die Unterschiede werden in 7.1.2 und 7.1.3 dargestellt.

Gemeinsam ist diesen Dateiarten, daß der direkte Zugriff auf beliebige Sätze möglich ist. Dadurch unterscheiden sie sich von den sequentiellen Dateien, auf deren Sätze nur mittelbar zugegriffen werden kann, indem man nämlich zunächst alle Sätze in der Reihenfolge einliest, in der sie zuvor geschrieben wurden, bis man den gewünschten Satz erreicht. Hinzu kommt, daß einmal erstellte sequentielle Dateien nicht verändert werden können, während in Direktzugriffsdateien Sätze sowohl eingefügt, geändert als auch gelöscht werden können.

Im Gegensatz zu sequentiellen Dateien auf BS1MP- oder INTEL-formatierten Disketten müssen RFM-Dateien vor ihrer ersten Benutzung mit einem Dienstprogramm (REORG) angelegt werden, wobei u.a. Satzlänge und Dateigröße festgelegt werden. Eine Erweiterung der Datei ist nur durch ihre Übertragung auf eine vorher angelegte größere Datei möglich.

RFM-Dateien werden im Inhaltsverzeichnis der Diskette mit einem X-Attribut gekennzeichnet. Hierdurch wird verhindert, daß die Dienstprogramme COPY und EDIT auf diese Dateien angewendet werden können.

RFM-Dateien können nur mit REORG oder DCOPY auf andere Disketten übertragen werden.

Mehrteilige Dateien

Beide Dateiarten können sich über mehr als eine Diskette erstrecken, oder können aus mehreren Teilen auf einer Diskette oder aus einer Kombination beider bestehen. In diesen Fällen wird die Datei als eine mehrteilige Datei bezeichnet und alle ihre Teile müssen zur korrekten Benutzung präsent sein.

Normalerweise gibt es keine Beschränkung für Dateinamen der Direktzugriffs-Dateien; ausgenommen, daß der Name mit einem Buchstaben beginnen muß; für mehrteilige Dateien aber muß eine Erweiterung angefügt sein, bestehend aus:

1. dem Zeichen "M" (groß geschrieben)

2. einer Ziffer im ASCII-Code, die die Reihenfolge des betreffenden Dateiteils innerhalb der Datei bezeichnet (z.B.: "2" bezeichnet den 2. Teil der Datei).

3. einer Ziffer im ASCII-Code, die die Gesamtzahl der Datei-Teile angibt (z.B. "3" in einer dreiteiligen Datei)

Auf die gesamte mehrteilige Datei kann man sich beziehen, indem man die Erweiterung "MLT" benutzt. Die Datei "ABCDEF.MLT" kann z.B. aus den drei Dateien "ABCDEF.M13", "ABCDEF.M23" und "ABCDEF.M33" bestehen.

7.1.2 Direktdateien

Direktdateien ermöglichen dem Benutzer den Zugriff zu einem beliebigen Satz in der Datei durch einmaligen Zugriff auf Diskette.

7.1.2.1 Eigenschaften von Direktdateien

- Feste Satzlänge zwischen 1 und 250 Bytes einschließlich, die beim Anlegen der Datei festgesetzt wird.
- Sätze werden - wenn möglich - geblockt. Die Blockgröße beträgt 250 Bytes auf INTEL- und 506 Bytes auf BS1M-formatierten Disketten. Sätze erstrecken sich nicht über Blockgrenzen.
- Die Datei erstreckt sich über einen fortlaufenden Bereich, der an einer Spur-Grenze anfängt und an einer solchen endet.
- Sätze können gelöscht werden.
- Auf Sätze kann fortlaufend oder über eine Satznummer (im Bereich 1 bis 65535) zugegriffen werden. (Die höchste und niedrigste Satznummer muß beim Anlegen der Datei angegeben werden). Der Schlüssel ist nicht Bestandteil des Satzes.

7.1.2.2 Operationen auf Direktdateien (Übersicht)

Eröffnen/Schließen (OPEN/CLOSE)

Sequentielles Lesen/Schreiben (SEQUENTIAL READ/WRITE)

Rückschreiben des letzten gelesenen Satzes (RE-WRITE)

Lesen/Schreiben im Direktzugriff (RANDOM READ/WRITE)

Rückschreiben im Direktzugriff (RANDOM REWRITE)

Satz löschen (DELETE)

7.1.3 Indizierte Dateien

Indizierte Dateien erlauben dem Benutzer den Zugriff zu einem beliebigen Satz der Datei in zwei Disketten-Zugriffen (im Normalfall) oder in mehreren Zugriffen, falls neue Sätze eingefügt und die Datei nicht reorganisiert wurde.

7.1.3.1 Eigenschaften indizierter Dateien

- Feste Satzlänge zwischen 1 und 250 Bytes inklusive. Sie wird bei Einrichtung der Datei festgelegt.
- Sätze werden - wenn möglich - geblockt. Die Blockgröße beträgt 250 Bytes bei INTEL- und 506 Bytes bei BS1M-formatierten Disketten. Sätze erstrecken sich nicht über Blockgrenzen.
- Die Datei erstreckt sich über einen fortlaufenden Bereich, der an einer Spurgrenze beginnt und an einer solchen endet.
- Sätze können als gelöscht markiert werden.
- Auf Sätze kann sequentiell oder durch ein für jeden Satz eindeutiges Satzkennzeichen *) zugegriffen werden. Dieses Kennzeichen kann 1 bis 24 Bytes lang sein; seine Lage und Länge werden beim Einrichten der Datei festgelegt.
- Auf Sätze kann mit teilqualifizierten (unvollständigen) Schlüsseln zugegriffen werden.
- Der Schlüssel darf im ersten Byte keine Hex-Werte 80 enthalten, da solche Schlüssel nicht mehr gelesen werden können und beim nächsten REORG gelöscht werden.
- Das Satzkennzeichen wird innerhalb des Satzes geführt. Seine Stellung im Satz wird beim Einrichten der Datei festgelegt.
- Für jede Datei gibt es nur einen Index.
- Gelöschte Sätze können wiedergewonnen werden, da sie beim Löschen nicht entfernt werden.
- Eine Datei-Reorganisation wird von Zeit zu Zeit notwendig, um:

  1. gelöschte Sätze zu entfernen und
  2. erzeugte (neue) Sätze, die in den Überlaufbereich geschrieben wurden, an ihren richtigen Platz einzufügen, um längere Zugriffszeiten zu vermeiden.

*) Andere Bezeichnungen: Identifikator, Schlüssel

7.1.3.2 Operationen auf indizierten Dateien (Übersicht)

Eröffnen/Schließen (OPEN/CLOSE)

Sequentielles Lesen/Schreiben (SEQUENTIAL READ/WRITE)

Rückschreiben (RE-WRITE)

Lesen/Schreiben im Direktzugriff (RANDOM READ/WRITE)

Rückschreiben im Direktzugriff (RANDOM RE-WRITE)

Satz löschen (DELETE)

Satz wieder einfügen (RESURRECT)

7.2 Assembler-Anschluss des RFM

Um den RFM in Assemblerprogrammen verwenden zu können, muß das Modul RFMMOD.REL zu dem Programm hinzugebunden werden. Es leistet zweierlei:

- es prüft, ob der RFM geladen ist
- es setzt Aufrufe in eine für dem RFM geeignete Form um.

Im Assemblerprogramm müssen eine Reihe von Variablen als PUBLIC sowie einige Bezüge zu Ansprungadressen mit EXTRN deklariert werden. Aufrufe an das Verwaltungssystem RFM erfolgen durch CALL-Aufruf auf die Adresse "RFM" (EXTRN deklariert), wobei im Doppelregister BC die Adresse eines Parameterblockes übergeben wird.

PUBLIC-Anforderungen

Jedes Programm, das zu RFMMOD.REL gebunden wird, muß folgende 'PUBLIC'-Deklarationen machen:

1. "RFMDLA", ein Datenwort, das die untere Adresse des Datenbereichs enthält, der für die Zusammenarbeit mit dem Verwaltungssystem benutzt wird. (Näheres siehe "Datenbereich").

2. "RFMDHA", ein Datenwort, das die obere Adresse des Datenbereichs enthält, der für die Zusammenarbeit mit dem Verwaltungssystem benutzt wird. Nach seiner Initialisierung benutzt das Verwaltungssystem dieses Datenwort als Zwischenspeicher.

3. "RFMWW", ein Datenwort, das als Zwischenspeicher benutzt wird.

4. "RFMWB", ein Byte, das zur Zwischenspeicherung benutzt wird.

5. "RFMBSY", ein Byte, das (wenn es gesetzt ist) anzeigt, daß das Verwaltungsprogramm arbeitet. Dies ist nur in einer Multi-Task-Umgebung von Bedeutung; "RFMBSY" muß jedoch immer eingerichtet und 'PUBLIC' deklariert sein.

EXTERNAL-Definitionen

Die folgenden Namen sind in RFMMOD.REL PUBLIC deklariert und müssen deshalb in jedem Programm, das sie benutzt, EXTRN deklariert werden.

"RFM", der normale Einsprungs-Punkt des RFM

"RFMIN", der Einsprungs-Punkt zur Initialisierung.

Datenbereich

Das Verwaltungsssystem RFM hat keinen lokalen Datenbereich, sondern benutzt den im Anwenderprogramm definierten (vgl. obige PUBLIC-Anforderungen).

Die Grenzen des Datenbereichs werden durch den Inhalt der beiden Datenwörter RFMDLA und RFMDHA (deren Adressen PUBLIC sein müssen) definiert. Sie enthalten die untere bzw. obere Adresse des Datenbereichs.

Diese Datenwörter dürfen ihren Inhalt nicht ändern, wenn:

1. eine Direktzugriffsdatei noch geöffnet ist,

2. keine Re-Initialisierung des Verwaltungssystems vorgenommen werden soll.

Das Verwaltungssystem benötigt 1040 Bytes für Puffer und Datenspeicher, und zusätzlich im Durchschnitt 60 Bytes für jede offene Datei. (Diese 60 Bytes sollten nur als Anhaltspunkt benutzt werden, da Dateikontrollblöcke in ihrer Größe variieren, abhängig vom Typ der Datei).

Es muß genügend Speicher für das Verwaltungssystem RFM angelegt werden, um den schlimmsten Fall (maximale Anzahl von gleichzeitig geöffneten Dateien) abzudecken, da der Datenbereich nicht vergrößert werden kann, solange noch eine Datei offen ist.

Initialisierung

Vor seiner ersten Benutzung muß das Verwaltungssystem RFM initialisiert werden. Dies erfolgt durch:

1. Laden der Datenwörter "RFMDLA" und "RFMDHA" (vgl. 'Datenbereich') und

2. Aufrufen "RFMIN" (ohne Parameter).

Anmerkung: Diese Prozedur darf nicht ausgeführt werden, wenn noch eine Direktzugriffsdatei offen ist.

Ist der RFM nicht geladen, so wird Carry gesetzt und im Register A der Code 38H (Routine nicht verfügbar) übergeben.

7.3 Benutzerschnittstelle

Normaler Rücksprung

Eine fehlerfreie Operation wird durch ein leeres Carry-Flag angezeigt. Sowohl das Byte, das durch den Status-Code-Zeiger im Parameterblock (s.u.) adressiert ist, als auch das A-Register enthalten den Status-Code, oder sie sind Null, falls kein signifikanter Status anzuzeigen ist (in diesem Fall wird das Zero-Flag gesetzt). Alle anderen Register werden zurückgespeichert, Flags nicht.

Das Byte, das durch den Fehler-Code-Zeiger im Parameterblock adressiert wird, ist Null.

Fehler-Rücksprung

Eine fehlerhafte Operation wird durch das gesetzte Carry-Flag angezeigt. Sowohl das Byte, das durch den Fehler-Code-Zeiger im Parameterblock adressiert ist, als auch das A-Register enthalten den Fehler-Code.

Alle anderen Register werden zurückgespeichert, Flags nicht.

Das Byte, das durch den Status-Code-Zeiger im Parameterblock adressiert ist, enthält den laufenden Status.

Parameterblock

Jeder Aufruf an den RFM muß sich auf einen Parameterblock beziehen, der eine Liste von Adressen und 1-Byte-Codes mit Details über die geforderte Operation enthält.

Der Parameterblock hat das folgende Format:

1. (OCB) Operations-Code
2. (SCP) Status-Code-Zeiger
3. (ECP) Fehler-Code-Zeiger
4. (FNP) Datei-Nummern-Zeiger
5. (UBP) Benutzer-Puffer-Zeiger
6. (IP) Identifikations-Zeiger
7. (ILB) Identifikations-Länge

Für einige Operationen werden nicht alle Parameter benutzt, sie werden vom RFM ignoriert. Die Position der Zeiger im Parameterblock ist deshalb festgelegt.

Operations-Code

Dies ist ein Byte, dessen Inhalt die auszuführende Operation bestimmt. Jede Operation wird im Folgenden näher beschrieben. Die Operations-Codes (mit eventuell zusätzlich benötigten Parametern neben OCB, SCP, ECP und FNP) sind:

1	Lesen sequentiell (UBP)
2	Schreiben sequentiell (UBP)
3	Nicht belegt
4	Rückschreiben des letzten gelesenen Satzes (UBP)
5	Nicht belegt
6	Eröffnen (IP)
7	Schließen
8	Nicht belegt
9	Lesen direkt (UBP,IP, ILB *)
10	Schreiben direkt (erzeugen) (UBP,IP)
11	Nicht belegt
12	Rückschreiben direkt (UBP, IP)
13	Löschen direkt (IP)
14	Wiederherstellen direkt (IP) *)

*) Nur für indizierte Dateien

Status-Code-Zeiger

Dies ist die Adresse eines Bytes, in dem der Status-Code nach jedem Aufruf vom RFM zurückgegeben wird.

Fehler-Code-Zeiger

Dies ist die Adresse eines Bytes, in dem der Fehler-Code nach jedem Aufruf vom RFM zurückgegeben wird.

Datei-Nummern-Zeiger

Dies ist die Adresse eines Bytes, dessen Inhalt die zu benutzende Datei bezeichnet. Bei der Eröffnung wird die Dateinummer vom Verwaltungssystem RFM in dieses Byte abgespeichert.

Benutzer-Puffer-Zeiger

Dies ist die Adresse des Puffers, in/aus dem ein Satz zu lesen oder zu schreiben ist.

Identifikations-Zeiger

Dies ist die Adresse des Identifikators (=Satzschlüssel) in folgendem Format:

1. Für 'Eröffnen' (OPEN, Code 6), eine 11 Bytes lange Zeichenkette mit der Bezeichnung für das Laufwerk und den Dateinamen im INTEL-(BS1M-)-Format, sonst

2. für Direktdateien ein Wort, das die Satznummer in binärer Darstellung enthält.

3. für indizierte Dateien eine Zeichenkette von Bytes, die den Satzschlüssel enthält.

Identifikations-Länge

Dies ist nur von Bedeutung für direktes Lesen auf einer indizierten Datei.

Es ist ein Byte, welches die Anzahl von Bytes des Identifikators angibt, die beim Lesen berücksichtigt werden sollen. Wird Null spezifiziert, so bedeutet dies, daß die volle Länge des Identifikators zu benutzen ist. Wenn die Länge kleiner als die volle Identifikatorlänge ist, wird der in alphabetischer Reihenfolge erste Satz eingelesen, dessen Identifikator in der angegebenen Länge mit dem vorgegebenen Identifikator übereinstimmt.

7.4 Operationen

Jede RFM-Operation, wird im Folgenden detailliert beschrieben.

Anmerkung: Der Begriff 'Identifikator' wird durchgängig benutzt für:

1. den Identifikator (Satzschlüssel) für eine Indexdatei,

oder

2. die Satznummer für eine Direktdatei.

Für jede Operation kann vom RFM ein Wiederholungsstatus gegeben werden ('Retry'). Dies bedeutet, daß mehr als ein Versuch erforderlich war, um Daten auf die oder von der Diskette zu transferieren. Obwohl dieser Transfer möglicherweise nach mehreren mißlungenen Versuchen korrekt ausgeführt wurde, wird dem Anwenderprogramm die Fehlermeldung der Disketten-Behandlungsprozedur übergeben.

7.4.1 Operations-Code 1: Lesen sequentiell

Parameter: OCB, SCP, ECP, FNP, UBP

Liest den nächsten Satz von der bezeichneten Datei in den bezeichneten Puffer des Benutzerprogramms. Das bedeutet:

1. für neu eröffnete Dateien den ersten Satz, sonst
2. für Direktdateien den Satz, dessen Nummer um eins höher ist als der zuletzt bearbeitete Satz; oder
3. für Indexdateien den Satz, dessen Identifikator der nächste in aufsteigender Folge (Wertigkeit des ASCII-Codes) ist.

Kommentar

Für eine Direktdatei wird für einen aus binären Nullen bestehenden Satz ein 'Leersatz-Status' ausgegeben.

Wird der Fehler-Code ´69´ ("Satz gelöscht") ausgegeben, nachdem man einen Satz einer indizierten Datei gelesen hat, so enthält der Benutzer-Puffer die ursprüngliche ("Lebend"-) Version des Satzes.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'69' Satz gelöscht

'6E' Dateiende (EOF)

sowie die Fehlermeldungen des BS1MP.

7.4.2 Operations-Code 2: Schreiben sequentiell

Parameter: OCB, SCP, ECP, FNP, UBP

Schreibt den Satz vom bezeichneten Puffer des Benutzerprogramms an das aktuelle Ende der bezeichneten Datei.

Das bedeutet:

1. für Direktdateien den Satz, dessen Satznummer um eins größer ist, als die Satznummer des zuletzt bearbeiteten Satzes, und

2. für Indexdateien den nächsten Satz nach demjenigen mit dem höchsten Identifikator in ASCII-Folge. Der Identifikator in dem zu schreibenden Satz muß größer sein als der augenblicklich höchste Identifikator in der Datei.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'6A' ungültiger Identifikator

'6E' Dateiende

sowie die Fehlermeldungen des BS1MP.

7.4.3 Operations-Code 4: Rückschreiben zuletzt gelesener Satz

Parameter: OCB, SCP, ECP, FNP, UBP

Schreibt den Satz vom bezeichneten Puffer des Benutzerprogramms in den Satz, welcher zuletzt (fehlerfrei oder bei Indexdateien als gelöscht markiert) von der bezeichneten Datei gelesen wurde.

Kommentar

Ein gelöschter Satz einer indizierten Datei kann mit diesem Kommando zurückgeschrieben werden. Ist jedoch der letzte gelesene Satz ein "Null"-Satz einer direkten Datei, so wird der Fehler "6D" ("unerlaubter Operationscode") ausgegeben.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'6A' ungültiger Identifikator

'6D' ungültiger Operations-Code (es wurde vorher kein Satz gelesen)

sowie die Fehlermeldungen des BS1MP.

7.4.4 Operations-Code 6: Eröffnen Datei

Parameter: OCB, SCP, ECP, FNP, IP

Eröffnet die Datei, deren Name an der bezeichneten Identifikator-Adresse steht und ordnet der Datei eine Dateinummer zu.

Ist die Datei früher einmal eröffnet worden, ohne daß sie nach Beendigung der Bearbeitung geschlossen wurde, so wird der Fehler 64 ausgegeben. Die Datei muß mit dem Programm RCOVER geschlossen werden, bevor sie wieder verwendet werden kann.

Wird zusätzlich noch der Benutzer-Puffer-Zeiger (UBP) versorgt, so werden nach dem Eröffnen einige Informationen über die Datei im Benutzer-Puffer übergeben, nämlich:

Byte	Bit	Direktdatei	Indexdatei
0	0	Dateityp (0=Direkt, 1=Index)	
0	4	0=Datei geschlossen, 1=Datei offen	
0	5	0=Datei neu, 1=auf Datei wurde zugegriffen	
1		Satzlänge	
2		erste Datenspur *)	
3		letzte Datenspur *)	
4		höchste ge-	Anzahl der
		schriebene	Sätze
		Satznummer	
5			
6		niedrigste	Erste Index-
		Satznummer	Spur *)
7			Letzte Daten- Spur *) = Byte 3
8		höchste	Schlüssel- länge
9		Satznummer	Schlüsselposition im Satz

Die Angaben sind immer hexadezimal.

*) Bei Direktzugriffsdateien auf BS1M-Disketten wird die logische Spur angegeben. Die physikalische Spur ist die logische Spur geteilt durch zwei. Ist der Rest 0, so befindet sich die Spur auf Seite 0, andernfalls auf Seite 1 der Diskette.

Kommentar

Auf eine Datei kann nur zugegriffen werden, nachdem sie eröffnet wurde. Alle folgenden Operationen müssen auf die Datei durch ihre Dateinummer verweisen.

Der Dateiname ist eine 11 Bytes lange (fixe Länge!) Zeichenkette im ASCII-Code und enthält:

1. (2 Bytes): Laufwerk (z.B. "D1"). Der Disketten-Typ (z.B. "D") muß immer angegeben werden. Die Laufwerksnummer kann wahlweise angegeben werden und kann durch ein Leerzeichen ersetzt werden. Dies bedeutet: eine Suche über alle Laufwerke des korrekten Typs in aufsteigender Folge der Nummern bis zum ersten Auftreten der Datei.

2. (6 Bytes): Name (z.B. "ABCDEF"); falls notwendig mit binär Null aufgefüllt.

3. (3 Bytes): Erweiterung (z.B. "XYZ"); falls notwendig mit binär Null aufgefüllt.

Beispiel in Assembler:

```
        db    'KUND',0,0,'001'
```

Der Dateiname ist KUND.001.

Die Dateinummern liegen im Bereich 1-99 einschließlich und werden beginnend mit 99 vergeben.

Falls die Datei mehrteilig ist, werden sämtliche Teile nur dann eröffnet, wenn die Erweiterung 'MLT' benutzt wird. In diesem Fall wird die Laufwerksnummer (s.o.) ignoriert und alle Laufwerke werden für jeden Teil durchsucht (jede Suche beginnt bei Laufwerk 0). Sonst wird nur ein einziger Teil eröffnet und dieser wird als eine einteilige Datei behandelt. (d.h. unabhängig von irgendeinem anderen Teil).

Für jede eröffnete Datei wird ein Dateikontrollblock reserviert. Jedem Teil einer mehrteiligen Datei wird ein eigener Dateikontrollblock zugeordnet.

Fehlermeldungen

'0A' Datei nicht gefunden

'63' Datei ist keine Direktzugriffsdatei

'64' Datei wurde offengelassen

'6F' ungenügender Speicher, um Datei zu öffnen

sowie die Fehlermeldungen des BS1MP.

7.4.5 Operations-Code 7: Schließen Datei oder Dateien

Parameter: OCB, SCP, ECP, FNP

Schließt die bezeichnete Datei (oder alle Dateien, wenn die Dateinummer 0 ist). Gibt den/die Dateikontrollblock/-blöcke und Dateinummer(n) frei.

Kommentar

Der Dateikopf auf der Diskette wird aktualisiert, um eine Änderung im Status der Datei festzuhalten (z.B. Aktualisieren 'Höchster geschriebener Satz') und um die Datei als 'geschlossen' zu markieren.

Eine Diskette darf nicht aus ihrem Laufwerk entfernt werden, während eine auf ihr befindliche Direktzugriffsdatei geöffnet ist, da

1. diese Datei nicht wiedereröffnet werden kann und
2. eine eventuell an ihrer Stelle eingelegte andere Datei zerstört werden kann.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

sowie die Fehlermeldungen des BS1MP.

7.4.6 Operations-Code 9: Lesen direkt

Parameter: OCB, SCP, ECP, FNP, UBP, IP, ILB (nur bei Indexdateien)

Liest den Satz mit dem bezeichneten Identifikator von der bezeichneten Datei in den bezeichneten Puffer des Benutzerprogramms.

Kommentar

Es ist bei Indexdateien möglich, eine Identifikator-Länge anzugeben, die kürzer ist, als die bei dem Erstellen der Datei spezifizierte (vgl. Parameter 'ILP' (Kapitel 'Benutzerschnittstelle')). In diesem Fall ist der gelesene Satz der erste (falls überhaupt einer gefunden wird), dessen Identifikator mit dem bezeichneten Identifikator in der angegebenen Länge übereinstimmt. Es handelt sich hier also um Lesen mit einem teilqualifizierten Identifikator.

Wird der Fehler-Code "69" ("gelöschter Satz") ausgegeben, wenn man in einer indizierten Datei gelesen hat, so enthält der Benutzer-Puffer die ursprüngliche ("Lebend"-) Version des Satzes.

Für eine Direktdatei wird ein 'Leer-Satz'-Status angegeben, wenn der Satz vollständig aus binären Nullen besteht.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'68' Satz nicht gefunden

'69' Satz gelöscht

'6A' ungültiger Identifikator

'6E' Dateiende

sowie die Fehlermeldungen des BS1MP.

7.4.7 Operations-Code 10: Schreiben direkt (Erzeugen)

Parameter: OCB, SCP, ECP, FNP, UBP, IP

Schreibt den Satz aus dem bezeichneten Puffer des Benutzerprogramms auf die bezeichnete Datei mit dem bezeichneten Identifikator.

Kommentar

Ein Satz mit demselben Identifikator darf nicht existieren (bei Indexdateien auch nicht gelöscht).

Bei Indexdateien wird der Schlüssel aus dem Anwenderpuffer genommen.

Falls der bezeichnete Identifikator der bis dahin höchste ist, so wird der Satz am Dateiende angefügt (vgl. oben: 'Schreiben sequentiell').

Ein 'Überlauf-Warnungs'-Status wird gegeben, wenn der letzte Überlauf-Block benutzt wird (nur bei Indexdateien). Jede weitere Anfügung eines neuen Satzes kann zu einem 'Überlauf'-Fehler führen. Man beachte, daß der verlorene Satz nicht der aktuelle sein muß.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'66' Überlauf-Fehler

'67' Satz existiert bereits

'6A' ungültiger Identifikator

'6E' Dateiende

sowie die Fehlermeldungen des BS1MP.

7.4.8 Operations-Code 12: Rückschreiben direkt

Parameter: OCB, SCP, ECP, FNP, UBP, IP

Schreibt den Satz aus der bezeichneten Datei aus dem bezeichneten Puffer des Benutzerprogramms mit dem bezeichneten Identifikator zurück.

Kommentar

Der bezeichnete Satz muß bereits existieren (im Falle von indizierten Dateien kann er auch gelöscht sein).

Man achte bei Indexdateien darauf, daß der Schlüssel im Satz vor dem Rückschreiben nicht verändert wurde. Es wird der Schlüssel aus dem Anwenderpuffer genommen.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'68' Satz nicht gefunden

'6A' ungültiger Identifikator

sowie die Fehlermeldungen des BS1MP.

7.4.9 Operations-Code 13: Löschen direkt

Parameter: OCB, SCP, ECP, FNP, IP

Löscht den Satz mit dem bezeichneten Identifikator auf der bezeichneten Datei.

Kommentar

Der bezeichnete Satz wird bei Indexdateien als gelöscht markiert. Bei Direktdateien ist dies ein nicht-reversibler Vorgang, bei Indexdateien kann der Satz jedoch wieder hergestellt werden (s. Operationscode 14).

Gelöschte Sätze werden bei Indexdateien nicht aus der Datei entfernt. Dies geschieht erst durch Überschreiben oder Reorganisieren.

Ist der Satz bereits gelöscht, so wird ein 'bereits ausgeführt'-Status gegeben.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'68' Satz nicht gefunden

'6A' ungültiger Identifikator

sowie die Fehlermeldungen des BS1MP.

7.4.10 Operations-Code 14: Wiederherstellen direkt

Parameter: OCB, SCP, ECP, FNP, IP

Stellt den Satz mit dem bezeichneten Identifikator in der bezeichneten Datei wieder her.

Kommentar

Diese Operation ist nur für Indexdateien verfügbar. Falls der Satz nicht gelöscht ist, wird ein 'schon ausgeführt'-Status gegeben.

Fehlermeldungen

'62' Dateinummer nicht in Benutzung

'68' Satz nicht gefunden

'6A' ungültiger Identifikator

'6D' unzulässiger Operations-Code (für Direktdateien)

sowie die Fehlermeldungen des BS1MP.

7.5 Status/Fehler-Codes

Das Verwaltungsprogramm für Direktzugriffsdateien gibt an das Anwenderprogramm jeweils Fehler- (0, wenn fehlerfrei) und Status-Codes zurück.

7.5.1 Status

Ein Status-Code wird stets in dem Byte, das durch den Status-Code-Zeiger im Parameter-Block adressiert ist, zurückgegeben. Falls keine Fehlerbedingung vorliegt (Carry = 0) wird der Status-Code auch im A-Register gespeichert; gibt es keine Statusmeldung, wird das Null-Flag gesetzt, A auf 0 gesetzt und das Carry-Flag gelöscht. Ein von Null abweichender Status sollte als Warnung, nicht als Fehler betrachtet werden, da in allen Fällen die geforderte Operation beendet wurde.

Der (hexadezimale) Status-Code wird durch inklusives ODER gebildet aus:

'01' Wiederholungswarnung
Es war mehr als ein Versuche nötig, um die Daten auf/von Diskette zu transferieren.

'02' Überlaufwarnung
Der letzte Überlaufblock wurde benutzt (nur bei Indexdateien). Weiteres Einfügen von Sätzen kann Fehler '66' produzieren. Die Datei sollte reorganisiert werden.

'04' Bereits ausgeführt
Der bezeichnete Satz ist schon im geforderten Status (gelöscht/vorhanden).

'08' Leer-Satz
Der gerade gelesene Satz besteht nur aus Nullen (nur bei Direkt-Dateien).

7.5.2 Fehler

Ein Fehler-Code wird stets in dem Byte zurückgegeben, das durch den Fehler-Code-Zeiger im Parameter-Block adressiert wird. Ist der Inhalt ungleich null (d.h. ein Fehler ist aufgetreten), so wird der Fehler-Code auch im A-Register gespeichert und das Carry-Flag gesetzt. In allen Fällen wurde die geforderte Operation nicht vollendet.

Die (hexadezimalen) Fehler-Codes sind:

'0A' Datei nicht gefunden
Die bezeichnete Datei ist auf der Diskette nicht vorhanden.

'3B' Falscher Disketten-Typ
Die Diskette ist nicht im BS1M- oder INTEL-Format beschrieben.

'38' RFM nicht geladen

'60' Beschäftigt
RFM ist mit einer anderen Aufgabe beschäftigt. Erneut versuchen, wenn das Byte 'RFMBSY' null ist.

'62'
Dateinummer nicht in Benutzung
Die bezeichnete Dateinummer wurde keiner Datei zugeordnet.

'63' Keine Direktzugriffs-Datei
Datei-Attribut nicht gleich X.

'64' Datei nicht geschlossen
Die Datei wurde nicht korrekt geschlossen. Auf sie kann nicht zugegriffen werden.

'65' Ungültiger Dateiname
Der bezeichnete Dateiname ist nicht korrekt spezifiziert.

'66' Überlauf-Fehler
Kein Platz in der Datei für diesen Satz. Durch die letzte Operation können Sätze verlorengegangen sein. Die Datei sollte reorganisiert werden.

'67' Satz bereits vorhanden
Ein Satz mit dem bezeichneten Identifikator existiert bereits ("Rückschreiben" muß zur Änderung dieses Satzes benützt werden).

'68' Satz nicht gefunden
Der bezeichnete Satz kann in der Datei nicht gefunden werden (nur bei Indexdateien).

'69' Satz gelöscht
Der bezeichnete Satz wurde als gelöscht markiert (nur bei Indexdateien). Seine Lebendversion steht aber im

Puffer.

'6A' Ungültiger Identifikator
Der bezeichnete Identifikator ist nicht korrekt spezifiziert.

'6D' Unzulässiger Operations-Code
RFM kennt diesen Operations-Code nicht, oder er ist in diesem Zusammenhang ungültig (z.B. Wiederherstellen eines Satzes in einer Direktdatei).

'6E' Dateiende
Das Dateiende wurde erreicht.

'6F' Zu wenig Speicher
es ist nicht genügend Speicherplatz vorhanden, um einen Dateikontrollblock für die bezeichnete Datei anzulegen (Speicherplatz wird freigegeben, wenn Dateien geschlossen werden).

'99' Falscher Parameter
Falsche Parameterangabe in einem Dienstprogramm.

Neben den oben beschriebenen Fehlermeldungen werden außerdem die Fehlermeldungen des Betriebssystems BS1MP weitergereicht.

7.6 Dienstprogramme

Die folgenden Dienstprogramme stehen im Zusammenhang mit dem Direktzugriffssystem zur Verfügung:

RELRFM

UMRFM

REORG

RCOVER

SORT.RFM

7.6.1 RELRFM

Das Programm dient dazu, den vom RFM-Modul belegten Speicherplatz für den Anwender freizugeben.

Aufruf: RELRFM

Hinweise: Mit dem Programm HELP kann man sich über den Umfang des freien Hauptspeichers informieren, das Programm gibt auch an, ob der RFM geladen ist oder nicht.

Versucht man, auf den RFM zuzugreifen, obwohl er nicht geladen ist, wird der Fehler 38H (Routine nicht verfügbar) ausgegeben. Der RFM kann nur durch erneutes Laden des BS1MP geladen werden.

7.6.2 UMRFM

Das Programm dient dazu, Direktzugriffsdateien, die mit RFM-Versionen 1.5 oder kleiner unter dem Betriebssystem BS1 der 6.610 erstellt wurden, auf die vom BS1MP verlangte Form umzusetzen. Die umgesetzten Dateien können sich sowohl auf INTEL- als auch auf BS1M-formatierten Disketten befinden.

Aufruf: UMRFM

oder UMRFM$:Fi:<ISAM.ALT>, {:Fj: / :Dj:} <ISAM.NEU>

oder UMRFM$:Fi:<ISAM.ALT>

In allen Fällen ist die Laufwerksnummer i ≠ j, <ISAM.ALT> die Quell- und <ISAM.NEU> die Zieldatei.

In der ersten Form wird die Angabe der Quell- und Zieldatei nachträglich erwartet, in der dritten Aufrufvariante nur noch die der Zieldatei.

UMRFM legt die Zieldatei nicht selbst an, dies muß zuvor mit REORG geschehen. Dabei ist auf die ausreichende Größe der Zieldatei zu achten.

Weiter ist zu beachten, daß Multi-Volume-Dateien nicht als ganzes sondern nur Teil für Teil umgesetzt werden können.

UMRFM kann mit Eingabe von '*' abgebrochen werden.

Das Programm kann in einer EXEC-Kommandodatei ablaufen. Nach Aufruf von UMRFM sollte '*' folgen, damit im Fehlerfall der UMRFM-Lauf abgebrochen wird.

Fehlermeldungen:

Zusätzlich zu den BS1MP- und RFM-Fehlermeldungen kann das Programm UMRFM folgende Fehler ausgeben:

RFM not available	Der RFM wurde mit RELRFM freigegeben. Das Programm UMRFM wird beendet.
Input Param wrong	Eingabedatei wurde falsch eingegeben (falsches Format).
Input too long	Es wurden mehr als 15 Zeichen für den Datennamen angegeben.
No Multi-Volume File allowed	Es wurde eine Multi-Volume-Datei zugewiesen, obwohl dies nicht zulässig ist.
Open Error: XX	Beim Eröffnen trat der RFM-Fehler XX auf.
Files are not equal	Die zugewiesenen Dateien sind unterschiedlich (Nicht beides Index- oder Direktdateien).
Record length is not equal	Die Dateien haben unterschiedliche Satzlänge.
Key length is not equal	Bei Indexdateien sind die Schlüssellängen unterschiedlich.
Key displacement is not equal	Bei Indexdateien ist die Lage des Schlüssels ungleich.
RFM-Input Error: XX	Beim Lesen trat der RFM-Fehler XX auf.
RFM-Output Error: XX	Beim Schreiben trat der RFM-Fehler XX auf.
Job aborted	Wegen eines Fehlers wird das Programm abgebrochen.

7.6.3 REORG

Mit diesem Programm werden Direktzugriffsdateien angelegt, kopiert oder Indexdateien reorganisiert, wenn bei ihnen keine freien Überlaufbereiche mehr vorhanden sind. Man beachte, daß Direktzugriffsdateien nur mit REORG, nicht jedoch mit COPY kopiert werden können.

Folgende Punkte sind bei Benutzung des Programmes zu berücksichtigen:

1. Bei einer Reorganisation darf die Zieldatei noch nicht angelegt sein.

2. Wenn beide Dateien bei einer Reorganisation sich auf derselben Disketten befinden, dürfen sie nicht denselben Namen (einschließlich Erweiterung) haben.

3. Bei mehrteiligen Dateien müssen die einzelnen Teile einzeln reorganisiert werden.

4. Reorganisiert werden kann von Index auf Index, von Index auf Direkt und von Direkt auf Direkt.

Aufruf:

1. REORG
   Dabei werden benötigte Parameter vom Benutzer abgefragt.

2. REORG$:xx:Datei,:yy:
   In dieser Variante wird die angegebene Datei auf ein anderes Laufwerk unter dem gleichen Dateinamen und gleichbleibenden Dateiparametern reorganisiert.

3. REORG$:xx,yy:
   In dieser Variante werden alle Dateien des angegebenen Laufwerks auf ein anderes Laufwerk reorganisiert. Der Name der gerade bearbeiteten Datei wird angegeben durch:

   CHECKING: Datei

4. Aufruf durch ein anderes Programm. Dazu müssen Daten im zum REORG gehörenden Umgebungsmodul wie folgt vorbesetzt sein.

   1. laden mit CALL LOAD
   2. in HL steht jetzt die Startadresse
   3. Startadresse retten
   4. Startadresse +3 ungleich 0 setzen
   5. die Kommandofolge z.B.: :D1:FILE.RFM,:D2:(cr) ab Startadresse +4 übertragen
   6. Startadresse nach HL bringen und Programm mit PCHL starten.

Nach Aufruf des Programmes muß der Anwender eine Reihe von Fragen beantworten (sie werden hier durch Unterstreichung markiert):

Reorganize or Create?(R/C):

Gibt man "C" an, so wird eine neue Datei angelegt. "R" muß dann angegeben werden, wenn eine bestehende Datei reorganisiert oder - was im Effekt dasselbe ist - kopiert werden soll. Eine Reorganisation kann z.B. erwünscht sein, wenn die alte Datei keine weiteren Sätze mehr aufnehmen kann.

Auch bei einer Reorganisation wird eine neue Datei angelegt. In sie werden anschließend alle nicht gelöschten Sätze der zu reorganisierenden Datei hineingeschrieben. Da mit REORG die Größe der neuen Datei beliebig festgelegt werden kann, kann man auf diesem Weg einer Umkopierung eine Datei vergrößern. Die alte Datei kann anschliessend mit DELETE gelöscht werden.

Source File Type?(I = IBM, F = INTEL, D = BS1M DD):

Die Frage tritt nur bei Reorganisation auf. Mit Quelldatei (Source File) ist die zu reorganisiernde Datei gemeint.

Z.Zt. können Direktzugriffsdateien nicht auf IBM-formatierten Disketten angelegt werden, I kann daher nicht eingegeben werden.

Source File Unit?(0 - 3):

Nur bei Reorganisation.

Hier muß die Laufwerksnummer der Quelldatei eingegeben werden.

Source File Name?(INTEL - xxxxxx.xxx):

Nur bei Reorganisation.

Der Name muß hier entsprechend den Konventionen des BS1MP angegeben werden (maximal 6 Zeichen vor und maximal 3 Zeichen hinter dem Punkt). Außerdem darf der Name nur mit einem Buchstaben beginnen. Man beachte außerdem die für Multi-Volume- Dateien geltenden Einschränkungen hinsichtlich der Namenserweiterung.

Destination File Structure?(I = Index, D = Direct):

Reorganisiert man eine Datei, so wird man i.a. bei der Quell- und bei der Zieldatei die gleiche Struktur wünschen. Man kann jedoch eine Indexdatei in eine Direktdatei kopieren. Dabei wird dem in alphabetischer Reihenfolge niedrigsten Satz der Index-datei die niedrigste Satznummer der Direktdatei zugeordnet, die weiteren Sätze der Index-Datei werden den Satznummern der Direkt-datei in aufsteigender Reihenfolge lückenlos zugeordnet.

Die Reorganisation einer Direkt- in eine Indexdatei ist nicht möglich.

Pack/Unpack?(P/U):

Die Frage tritt nur bei der Reorganisation von Indexdateien auf.

Die Eingabe hat Einfluß auf das Verhalten des RFM beim Einfügen neuer Sätze in die reorganisierte Datei. Ist abzusehen, daß in die reorganisierte Datei (es muß sich um eine Indexdatei handeln) bald in ungeordneter Reihenfolge weitere Sätze geschrieben werden, sollte man "U" angeben. Handelt es sich jedoch um eine Datei, in die nur wenig neue Sätze eingetragen werden (z.B. eine Bestands-datei), sollte man "P" eingeben. In diesem Fall werden die Daten-spuren lückenlos aufgefüllt, ohne daß Überlaufblöcke in diesen Spuren reserviert werden.

Record length?(1 - 250):

Nur bei Ersterstellung einer Datei, nicht bei Reorganisation.

Bei Festlegung der Satzlänge sollte man den Verschnitt möglichst gering halten. Darunter versteht man folgendes:

Sätze werden in Blöcken von 506 Bytes (beim BS1M-Format) zusammen-gefaßt. Bei einer Satzlänge von 250 Bytes passen 2 Sätze in einen Block, 6 Bytes gehen verloren (d.h. 6 Bytes Verschnitt). Bei einer Satzlänge von 150 Bytes würden 3 Sätze in den Block passen, jedoch 56 Bytes ungenutzt bleiben.

Number of Data records?(1 - 65000):

Hier muß die maximale Anzahl von Sätzen des Datenbereiches ange-geben werden. Für diese Sätze wird auf der Diskette Speicher reserviert, sofern genug Platz zur Verfügung steht (andernfalls wird eine Fehlermeldung nach Abschluss aller Eingaben ausgegeben). Bei Indexdateien kommt zum Datenbereich gegebenenfalls noch der Überlaufbereich hinzu.

Zur Platzbelegung von Direktzugriffsdateien s. 7.7.

First Record No.?(1 - 65000):

Nur bei Direktdateien.

Die Satznummer des ersten Satzes einer Direktdatei kann willkürlich festgelegt werden. Hat die Datei M Sätze, der erste Satz die Nummer N, so kann man nur auf die Sätze N, N+1, N+2,..., N+M-1 zugreifen.

Identifier length?(1 - 24):

Nur bei Erstanlage einer Indexdatei, nicht bei Reorganisation.

Der alphanumerische Schlüssel (Identifikator) kann ein bis maximal 24 Zeichen lang sein. Er wird im Satz gespeichert.

Identifier displacement?(0 - (Record length - 1)):

Nur bei Erstanlage einer Indexdatei, nicht bei Reorganisation.

Hier muß die Position des Schlüssels innerhalb des Satzes angegeben werden. Steht er ganz am Anfang, wird 0 angegeben.

Nach Eingabe aller Parameter versucht REORG, die Ziel- bzw. neue Datei anzulegen. Stellt sich dabei heraus, daß kein ausreichender Platz auf der Diskette zur Verfügung steht, wird eine Fehlermeldung ausgegeben und das Programm beendet.

Fehlermeldungen:

PARAMETER ERROR - PLEASE TRY AGAIN
Fehler bei einem Parameter. Eingabe kann wiederholt werden.

FILE TYPES DIFFER
Dateiarten unterschiedlich.

RECORD LENGTH DECREASED
Satzlänge der Zieldatei ist kleiner als die der Quelldatei.

RECORD LENGTH INCREASED
Satzlänge der Zieldatei ist größer als Satzlänge der Quelldatei.

IDENTIFIER DECREASED
Satzschlüssel der Quelldatei ist größer als Satzschlüssel der Zieldatei

IDENTIFIER INCREASED
Satzschlüssel der Quelldatei ist kleiner als Satzschlüssel der Zieldatei.

OUTPUT FILE MARKED FULL
Die Zieldatei ist auch als voll markiert.

FILE(S) INCREASED

Sie bedeuten für die Variante 1, daß die als Parameter angegebene Anzahl von Sätzen für die vom REORG neu erstellte Datei kleiner ist als die Anzahl von Sätzen der alten Datei. Diese Anzahl wird als Parameter übernommen. Es kann dazu kommen, daß die reorganisierte Datei kleiner ist als die alte.

Wird von vornherein ein sehr großer Parameter angegeben, wird die Datei vergrößert, aber keine Meldung ausgegeben.
Für die anderen Varianten bedeutet die Meldung, daß infolge von Sätzen in Überlaufblöcken die neu erstellte Datei evtl. vergrößert wurde.

Die Meldung wird bei der dritten Variante nur einmal ausgegeben.

FEHLER 91
Datei bereits vorhanden

FEHLER 92
Zu wenig Platz auf der Diskette

FEHLER 93
Kompatibilitätsfehler

FEHLER 94
Ausgabedatei bereits vorhanden

Beipiele:

Die folgenden Beispiele zeigen EXEC-Dateien für den REORG-Aufruf:

1. Anlegen einer Indexdatei (Satzanzahl 1, Satzlänge 250, Schlüssellänge 5, Schlüsselposition 1)

2. Anlegen einer Direktdatei (Satzanzahl 20, Satzlänge 80, 1. Satznummer 6)

3. Reorgnisation einer Indexdatei (50 Datensätze)

```
REORG           REORG           REORG
cd1isam         cd0direkt       rd0isam
i250            d80             d1isam.neu
1               20              iu50
5               6               30
1
```

7.6.4 RCOVER

Dieses Dienstprogramm kann zur Wiederherstellung nicht geschlossener bzw. zerstörter Direktzugriffsdateien verwendet werden.

Beschreibung

RCOVER versucht, die angegebene Datei zu eröffnen (oder wahlweise, jede Direktzugriffsdatei der Diskette). Gelingt dies oder ist der Grund des Fehlers die Tatsache, daß die Datei bereits offen ist, werden die Dateiblöcke gelesen und die Blockverkettungen wie auch die Indexeinträge (wenn vorhanden) geprüft. Wird kein Fehler gefunden, wird die Datei geschlossen.

Kann die Datei nicht eröffnet werden oder treten Fehler beim Lesen der Diskette auf oder ist die Blockverkettung fehlerhaft (nur bei indexsequentiellen Dateien), werden alle verwendbaren Daten (einschließlich der gelöschten Sätze in indexsequentiellen Dateien) sequentiell in eine Datei geschrieben, deren Name von RCOVER angefordert wird. Die angegebene Datei kann, falls erforderlich, auch von RCOVER selbst angelegt werden. Ist sie jedoch bereits vorhanden, muß sie die weiter unten aufgezählten Einschränkungen erfüllen.

Wird die Ausgabedatei von RCOVER angelegt, so hat sie die gleichen Parameter wie die Eingabedatei, ausgenommen der Anzahl der Datensätze, welche möglicherweise vergrößert werden kann.

Einschränkungen

1. Jeder Teil einer mehrteiligen Datei muß einzeln behandelt werden. Die Erweiterung "MLT" ist unzulässig.

2. Eine Datei, deren Eintrag im Disketten-Inhaltsverzeichnis oder deren Dateikopf zerstört ist, kann von RCOVER nicht wiederhergestellt werden.

3. Wenn die von RCOVER angeforderte Ausgabedatei bereits existiert, darf auf sie in keiner Weise zugegriffen worden sein und sie muß von derselben Art (indexsequentielle oder Direktdatei) wie die ursprüngliche Datei sein.

   Für eine Direktdatei muß die kleinste Satznummer dieselbe wie in der ursprünglichen Datei sein; die größte Satznummer darf nicht kleiner als in der ursprünglichen Datei sein.

   Für eine indexsequentielle Datei muß die Schlüssellänge und Position aus der ursprünglichen Datei übernommen werden. Die Anzahl der Datensätze darf nicht kleiner sein als jene, welche erforderlich ist, um alle Sätze der ursprünglichen Datei zu erfassen.

Aufruf

RCOVER$ <EIN>

mit

<EIN> = Eingabedatei.
Wenn kein Dateiname angegeben wird, werden alle Direktzugriffsdateien des angegebenen Laufwerks bearbeitet.
Wird kein Laufwerk angegeben, so wird das Systemlaufwerk angenommen.

Wenn eine Ausgabedatei erforderlich ist, bringt das Dienstprogramm folgende Meldung:

"GIVE NAME OF FILE FOR RECOVERED DATA"

Die Antwort ist

<AUS> = Ausgabedatei.
Wird kein Laufwerk angegeben, so wird das System-Laufwerk angenommen.
Wird kein Dateiname angegeben, wird die Herstellung der Datei nicht durchgeführt und falls alle Direktzugriffsdateien eines Laufwerkes bearbeitet werden, wird das Dienstprogramm zur Prüfung der nächsten Datei übergehen.

Meldungen

Jede Meldung des RFM oder des Betriebssystems kann im Laufe des Dienstprogrammes RCOVER auftreten. Die Fehlermeldung 2F (Diskette voll) oder 09 (Disketten-Inhaltsverzeichnis voll) kann auch während des Versuches, eine Ausgabedatei anzulegen, auftreten.

Werden alle Direktzugriffsdateien einer Diskette bearbeitet, bewirkt ein nicht behebbarer Fehler in einer Datei, daß RCOVER mit der Prüfung der nächsten Datei fortsetzt.

Die folgenden Meldungen können von RCOVER ausgegeben werden:

RCOVER COMPLETED (RCOVER BEENDET)	Dienstprogramm wurde programmgemäß beendet.
RCOVER ABANDONED (RCOVER ABGEBROCHEN)	Beim Wiederherstellen einer einzelnen Datei wurde ein Fehler erkannt, der nicht behoben werden konnte. Der Grund des Abbruchs wird von einer vorgehenden Meldung angegeben.

FILE IS OK (DATEI IST IN ORDNUNG)	Die angegebene Datei ist in Ordnung und benötigt keine Änderungen. Werden alle Direktzugriffsdateien einer Diskette bearbeitet, kann diese Meldung für jede geprüfte Datei gegeben werden.
FILE CORRECTED (DATEI WURDE BERICHTIGT)	Änderungen im Dateikopf oder im Index-Block waren erforderlich.
NO RECOVERY POSSIBLE (KEINE WIEDERHERSTELLUNG MÖGLICH)	Die Datei konnte nicht eröffnet werden; dies lag jedoch nicht daran, daß sie schon geöffnet war.
FILE WAS LEFT OPEN (DATEI WURDE OFFEN-GELASSEN)	Die untersuchte Datei wurde von einem vorhergehenden Programm nicht geschlossen. Dies kann, muß aber nicht die Wiederherstellung der Datei veranlassen.
OUTPUT FILE CREATED (AUSGABEDATEI WURDE ANGELEGT)	Die angeforderte Ausgabedatei wurde angelegt.
DATA TRACK(S) ADDED (DATENSPUR(EN) WURDE (N) HINZUGEFÜGT)	Die von RCOVER angelegte Datei hat mehr Datenspuren als die ur sprüngliche Datei (um alle wieder hergestellten Daten zu erfassen).
NO RECORDS FOUND FOR RECOVERY (KEINE SÄTZE FÜR WIEDER-HERSTELLUNG GEFUNDEN)	In der ursprünglichen Datei wurden während des Lesevorganges keine Sätze gefunden. Zu diesem Zeitpunkt wurde noch keine Ausgabedatei angelegt oder auf sie zugegriffen.
FILE RECOVERED (DATEI WURDE WIEDER-HERGESTELLT)	Die Datei konnte wiederhergestellt werden.
FILE NOT RECOVERED (DATEI WURDE NICHT WIEDERHERGESTELLT)	Die Datei konnte nicht wiederhergestellt werden, weil der Bediener keine Ausgabedatei angegeben hat.
FILENAME SYNTAX ERROR, TRY AGAIN (FALSCHER DATEINAME, EINGABE WIEDERHOLEN)	Der Dateiname entsprach nicht den den Regeln des BS1M und muß korrigiert erneut eingegeben werden.
MULTIPART FILES NOT ALLOWED, TRY AGAIN (MEHRTEILIGE DATEIEN SIND UNZULÄSSIG, NEU EINGEBEN)	Name der Eingabedatei enthält die Erweiterung "MLT".

OUTPUT FILE ACCESSED PREVIOUSLY (AUF DIE AUSGABEDATEI WURDE BEREITS ZUGE-GRIFFEN)	Die Ausgabedatei ist vorhanden und auf sie wurde zuvor schon zugegriffen.
OUTPUT FILE NOT COMPATIBLE WITH INPUT (DIE AUSGABEDATEI IST MIT DER EINGABEDATEI NICHT KOMPATIBEL)	Die Ausgabedatei hat nicht die gleichen Parameter wie die ursprüngliche Datei oder sie ist nicht groß genug.
INDEX ENTRY WRONG (FALSCHER INDEX-EINTRAG)	
INDEX CANNOT BE READ (INDEX KANN NICHT GELE-SEN WERDEN)	
INDEX ENTRIES IN WRONG ORDER (INDEXEINTRÄGE IN FALSCHER REIHENFOLGE)	
DATA BLOCK CANNOT BE READ (DATENBLOCK KANN NICHT GELE-SEN ERDEN)	
ERROR IN DATA BLOCK (FEHLER IN EINEM DATENBLOCK)	
WRONG DATA BLOCK LINKING (FALSCHE VERKETTUNG VON DATENBLÖCKEN)	
FILEHEADER DEFECT (DATEIKOPF MIT FALSCHEM EINTRAG)	
RECORDS FOUND AFTER END OF DATA (SÄTZE WURDEN NACH DATEN-ENDE GEFUNDEN)	

Die oben genannten Ursachen geben die verschiedenen Dateifehlertypen an, welche von RCOVER erkannt werden.

7.6.5 SORT.RFM

1. Allgemeines

Das Programm SORT.RFM ermöglicht das Umsortieren einer Direktzugriffsdatei in eine andere. Beide Dateien können von gleicher Struktur, z.B. indexsequentiell, oder von unterschiedlicher Struktur sein. Folgende Möglichkeiten sind zulässig:

INDEX → INDEX
INDEX → DIREKT
DIREKT → DIREKT
DIREKT → INDEX

Die Dateien können auf Disketten im BS1M bzw. INTEL-Format vorliegen. Für den Sortiervorgang wird kein zusätzlicher Platz auf der Diskette benötigt.

2. Beschreibung

SORT.RFM sortiert den Inhalt einer Eingabedatei nach einem Schlüssel, der

- für eine direkte Ausgabedatei vom Anwender beim Aufruf angegeben wird (s.u.),

- bei indexsequentiellen Ausgabedateien mit dem Schlüsselbegriff übereinstimmt.

Gelöschte Sätze (Index-Dateien) und Leersätze (Direkt-Dateien) werden nicht übertragen. Beim Sortieren in eine Index-Datei können Sätze - beim doppelten Vorkommen eines Sortierschlüssels - verloren gehen.

3. Einschränkungen

- Beide Dateien müssen bereits existieren.

- Beide Dateien müssen die gleiche Satzlänge haben.

- Auf die Ausgabedatei darf zuvor noch nie zugegriffen worden sein.

- Wenn eine oder beide Dateien mehrteilig ist, müssen alle Teile der Datei(en) verfügbar sein.

4. Aufruf

SORT.RFM wird für direkte und indexsequentielle Ausgabedateien unterschiedlich aufgerufen (der Typ der Eingabedatei ist unwesentlich):

SORT.RFM$<ein>,<aus>,<sl>,<sp>

mit

<ein> = Eingabedatei
<aus> = Ausgabedatei
<sl> = Sortierschlüssellänge (nur bei direkten Ausgabedateien)
<sp> = Sortierschlüsselposition (nur bei direkten Ausgabedateien)

5. Meldungen

Falls während des Programmablaufs keine Fehler auftreten, wird das Beenden mit

Sort Completed

gemeldet.

Ansonsten erfolgt eine der folgenden Fehlermeldungen (hex) und

Sort Abandoned.

- 99 : Syntaktisch falsche Eingabe der Parameter, z.B. eines Dateinamens
- 0A : Datei nicht gefunden
- 63 : Datei ist keine Direktzugriffsdatei
- 64 : Datei wurde offengelassen
- 69 : Datei gelöscht
- 6F : Nicht genügend Speicher, um Datei zu eröffnen
- Output File Already Accessed

  : Auf die Ausgabedatei wurde bereits zugegriffen
- Record Lengths Differ

  : Die Satzlänge bei Ein- und Ausgabe ist unterschiedlich

sowie Fehlermeldungen des BS1MP.

7.7 Platzbedarf von Direktzugriffsdateien

Indexdateien

Indexdateien bestehen aus den folgenden Teilen:

Dateiteil	Anzahl Sektoren
Dateikopf	1
Indexspur	26 (= 1 Spur)
Datenspuren	x*26 (x Spuren)
Dateikettungssektoren	z

Datenspuren:

Eine Datenspur besteht aus 10 Daten- und 3 Überlaufblöcken. Ein Daten- oder Überlaufblock besteht aus 2 Sektoren. Im BS1M-Format hat ein Block daher 506, im INTEL-Format 250 Bytes (6 Verwaltungsbytes jeweils abgezogen).

Datensätze werden in Blöcken zusammengefaßt und können Blockgrenzen nicht überschreiten. Ist die Datensatzlänge z.B. 150 Bytes, so passen in einen Block (im BS1M- Format) 3 Sätze, 56 Bytes werden nicht genutzt.

Bei der Anlage der Datei wird für jeweils 10 Datenblöcke eine Spur reserviert. Für 10 Datenblöcken werden daher automatisch 3 zusätzliche Überlaufblöcke angelegt (außer dann, wenn P für PACK angegeben wurde, s. Abschnitt 7.6.3 REORG).

Die Zahl x der reservierten Datenspuren errechnet sich also aus:

$$x = \frac{\text{Anzahl der bei REORG angegebenen Datensätze}}{\text{Sätze/Block} * 10}$$

Die Zahl Sätze/Block wird gegebenenfalls abgerundet, x selbst muß bei nicht ganzzahligem Ergebnis aufgerundet werden, da immer ganze Spuren belegt werden. Zusätzlich wird noch eine Spur als Indexspur belegt.

Dateikettungsblöcke:

Hat man den Speicherbedarf der Indexdatei berechnet, muß jetzt noch berücksichtigt werden, daß für jeweils 62 Sektoren ein Kettungssektor angelegt wird, also

$$z = \frac{1 + 26 + x*26}{62}$$

Sektoren hinzukommen.

Bei nicht ganzzahligem Ergebnis muß z aufgerundet werden.

Beispiel (BS1M-Format):

Eine Indexdatei mit 3000 Datensätzen soll angelegt werden. Die Satzlänge beträgt 100 Bytes.

Es werden benötigt:

	Anzahl Sektoren
1 Sektor für den Dateikopf	1
26 Sektoren für die Indexspur	26
60 Spuren für Datensätze	1560
Zwischensumme:	1587
für jeweils 62 Sektoren einen Kettungssektor (aufgerundet)	26
Summe:	1613

Bei der Anlage der Datei ist zu beachten, daß Index- und Datenspuren auf der Diskette einen zusammenhängenden Speicherbereich benötigen. Es kann daher geschehen, daß die Datei nicht angelegt werden kann, obwohl genug freie Sektoren vorhanden sind.

Die Methode, mit der ein Satz in die Datei eingefügt wird, wird im Folgenden kurz dargestellt:

Soll ein Satz in die Datei eingefügt werden, obwohl an dessen Schlüsselposition bereits Sätze lückenlos liegen, so wird zunächst versucht, durch Verschieben der Sätze innerhalb des gleichen Datenblockes Platz zu schaffen.

Kann hierdurch kein ausreichender Platz bereitgestellt werden, ist jedoch der folgende Datenblock frei, so werden die Sätze des vollen Datenblockes auf zwei Blöcke verteilt (Blocksplitting). Der Teil mit den niedrigeren Schlüsseln bleibt im ursprünglichen Block.

Beim direkten Schreiben kann der neue Satz auch in einen Überlaufblock dieser Spur verkettet werden.

Ist auch der letzte Block belegt, so wird die gesamte Spur aufgeteilt: die Hälfte der Daten wird in die nächste Datenspur übertragen, so daß jetzt in beiden Spuren Platz für Einfügungen zur Verfügung steht (Tracksplitting). Im alten Spurbereich bleiben dabei die Sätze mit den niedrigeren Schlüsseln.

Achtung: Die letzte Datenspur kann nicht mehr gesplittet werden.

Überlaufwarnung erfolgt, wenn die letzte Datenspur durch Schreiben eines Satzes oder durch Tracksplitting angefangen wird.

Direktdateien

Eine Direktdatei besteht aus folgenden Teilen:

Dateiteil	Anzahl Sektoren
Dateikopf	1
Datenspuren	x*26 (x Spuren)
Dateikettungs-sektoren	z

Datenspuren:

Jede Datenspur besteht aus 13 Datenblöcken zu je 2 Sektoren. Im BS1M-Format hat ein Block 506, im INTEL-Format 250 Bytes (bereits abgezogen sind 6 Verwaltungsbytes).

Wie bei Indexdateien werden Datensätze in Datenblöcken zusammengefasst und können Blockgrenzen nicht überschreiten. x errechnet sich daher aus:

$$x = \frac{\text{Anzahl in REORG angegebener Sätze}}{\text{Sätze/Datenblock} * 13}$$

Sätze/Datenblock muß gegebenenfalls ab-, x selbst aufgerundet werden, wenn der jeweilige Wert nicht ganzzahlig ist.

Dateikettungssektoren:

Wie bei Indexdateien muß auch hier berücksichtigt werden, daß für jeweils 62 Sektoren der Direktdatei ein Kettungssektor angelegt wird. z ist also

$$z = \frac{\text{Anzahl Sektoren}}{62}$$

z muß gegebenenfalls aufgerundet werden.

Beispiel (BS1M-Format):

Es soll eine Direktdatei mit 3500 Sätzen angelegt werden. Die Satzlänge betrage 100 Bytes.

	Anzahl Sektoren
Dateikopf	1
54 Datenspuren	1404
Zwischensumme:	1405
Dateikettungsektoren (aufgerundet)	23
Summe:	1428

Man beachte auch hier, daß die Datenspuren zusammenhängend hintereinander auf der Diskette angelegt werden müssen.

7.8 Beispielprogramm

Das Programm eröffnet die Indexdatei "isam", wobei es das Laufwerk selbständig sucht, auf dem sich die Datei befindet. Anschließend liest es den Satz mit dem Schlüssel "MEIER " und schließt die Datei wieder.

```
$title *** Beispiel RFM-Zugriff ***
$pw 80
$pl 72
;
;
        cseg                ;Programm-Modul
        public   rfmdla     ;untere Adr. Daten-
                            ;bereich
        public   rfmdha     ;obere Adr. Daten-
                            ;bereich
        public   rfmww      ;Arbeitsspeicher
                            ; 1 Wort
        public   rfmwb      ; Arbeitsspeicher
                            ; 1 Byte
        public   rfmbsy     ;Merker "RFM beschäf-
                            ;tigt"
        extrn    rfmin      ;RFM-Einsprung für
                            ;Initialisierung
        extrn    rfm        ;normaler Einsprung
                            ;für RFM
```

```
;*
;*** RFM-Initialisierung  ***
;*
start:    lxi       h,datast  ;h/l=Anfang des
                                ;Datenbereichs
          shld      rfmdla    ;Anfang des Daten-
                                ;reichs --  rfmdla
          lxi       h,dataen  ;h/l=Ende des Daten-
                                ;bereichs
          shld      rfmdha    ;Ende des Datenbe-
                                ;reichs --  rfmdha
          call      rfmin     ;Initialisierung
                                ;des RFM
;*
;***  Öffnen  ***
;*
open:     mvi       a,6       ;A=open-Code
          sta       ocb       ;open-Code --
                                ;Parameterblock
          lxi       h,filnam  ;h/l=Adresse
                                ;des Dateinamens
          shld      ip        ;Adresse des Datei-
                                ;namens --  Parame-
                                ;terblock
          lxi       b,pblock  ;b/c=Adresse des
                                ;Parameterblocks
          call      rfm       ;Aufruf des RFM
          jnc       lesen     ;cy=0 --  kein Fehler
          jmp       rfmerr    ;Fehlerroutine
;*
;***  Satz lesen  ***
;*
lesen:    mvi       a,9       ;A=Lesen(direkt)Code
          sta       ocb       ;Lesecode --
                                ;Parameterblock
          lxi       h,name    ;h/l=Adresse des
                                ;Parameterblocks
          shld      ip        ;Adresse des Satz-
                                ;schlüssels --
                                ;Parameterblock
;
;(Dateinummer und Pufferzeiger sind unverändert)
;
          lxi       b,pblock  ;b/c=Adresse des
                                ;Parameterblocks
          call      rfm       ;Aufruf des RFM
          jnc       close     ;cy=0 --  kein Fehler
          jmp       rfmerr    ;Fehlerroutine
```

```
;*
;***  Schliessen  ***
;*
close:    mvi       a,7         ;A=Schliessen-Code
          sta       ocb         ;Code --  Parameter-
                                ;block
          lxi       b,pblock    ;b/c=Adresse des Para-
                                ;meterblocks
          call      rfm         ;Aufruf des RFM
          jc        rfmerr      ;cy=1 --  Fehlerroutine
          ret                   ;Ende des Programms
;*
;***  Fehlerbehandlung  ***
;*
rfmerr:   ret                   ;ins Betriebssystem
$E
;*
;***  Parameterblock  ***
;*
pblock:
ocb:      db        0           ;Befehlscode
scp:      dw        status      ;Adresse des Statuscodes
ecp:      dw        error       ;Adresse des Fehlercodes
fnp:      dw        filenb      ;Adresse der Dateinummer
ubp:      dw        buffer      ;Adresse des Anwender-
                                ;puffers
ip:       dw        0           ;Adresse des Satzschlüssels
ilb:      db        0           ;volle Schlüssellänge
status:   db        0           ;Statuscode
error:    db        0           ;Fehlercode
filenb:   db        0           ;Dateinummer
buffer:   ds        250         ;Anwenderpuffer
filnam:   db        'D isam',0,0,0,0,0
name:     db        'MEIER      '
rfmdla:   dw        0           ;untere Adresse des
                                ;Datenbereiches
rfmdha:   dw        0           ;obere Adresse des
                                ;Datenbereiches
rfmww:    dw        0           ;Arbeitsspeicher 1 Wort
rfmwb:    db        0           ;Arbeitsspeicher 1 Byte
rfmbsy:   db        0           ;Merker "RFM beschäftigt"
datast:   ds        1100        ;Datenbereich von 1100
                                ;Bytes
dataen:   ds        1
          end       start
```

Das Dateiverwaltungssystem steht noch nicht zur Verfügung.

9 Hinweise

9.1 "Patchen" eines Programmes

Mit der im folgenden beschriebenen Methode kann der Objektcode eines Programmes korrigiert und unter einem neuen Namen abgespeichert werden.

1. Laden des BS1MP (falls nicht schon geschehen).
2. DEBUG$<Programmname> (ohne Dateizuweisungen!).
3. Änderung des Objektcodes mit Hilfe von Monitor-Kommandos.
4. BS1MP durch Kommando R:Xn: erneut laden oder durch .GFD00 direkt anspringen
5. Aufruf des Programmes MEMDMP mit entsprechenden Zuweisungen und Parametern.

9.2 Starten eines Programmes innerhalb eines Programms

Mit Hilfe der Systemroutine LOAD (FD06H) ist es möglich, per Programm Module in den Speicher zu laden und zu starten, wodurch ein Overlay-Betrieb auch in Assemblerprogrammen möglich ist.

Der Call-Aufruf zur LOAD-Routine benötigt im Doppelregister H/L die Adresse des Namens der Datei, in der sich das Modul befindet. Bei der Darstellung des Namens wird der Punkt, der Namen und Namenserweiterung trennt, weggelassen. Name und Erweiterung werden jeder für sich links ausgerichtet und gegebenenfalls mit binär Null aufgefüllt. Die Länge des Namens muß neun Bytes betragen.

Für eine sinnvolle Overlay-Unterstützung ist die Verwendung von Publics und Externals sowohl innerhalb des Root-Teils als auch in den einzelnen Overlays erforderlich. Gleichzeitig muß die Lade- und Startadresse der Overlays festgelegt werden können. Dies kann beim Binden mit LINK durch Verwendung des X-Schalters erreicht werden:

Binderlauf für das Root-Segment:

Es müssen alle Hex-Dateien (inklusive aller Overlays) angegeben werden. Vor dem ersten und nach dem letzten Overlay muß der X-Schalter gesetzt bzw. rückgesetzt werden. Die Overlays werden dann nicht mit ausgegegeben.

Der Adreßpegel (*C und *D) muß für jedes Overlay neu gesetzt werden, andernfalls wird er fortlaufend hochgezählt, wodurch eine Überlagerung nicht möglich wäre.

Binderlauf für Overlay-Segmente:

Für jedes Overlay-Segment muß ein Binderlauf durchgeführt werden, in dem jeweils Root und Overlay gebunden werden. Das Root-Segment wird zwischen zwei X-Schalter gesetzt, damit es nicht mit ausgegeben wird. Für das Overlay-Segment kann dabei eine Startadresse angegeben werden.

In Anhang F befindet sich ein Beispiel für die Overlaytechnik.

9.3 Benutzung des Timers

Ein Anwenderprogramm kann sich die interne Uhr der 6.611 zunutze machen, indem es alle 20ms oder Vielfache von 20ms eine vorgegebene Programmroutine (Interruptroutine) durchlaufen läßt. Dazu muß das Anwenderprogramm beim Aufruf der Routine STIMER (Set Timer, FD3CH) im Doppelregister H/L die Adresse seiner Interruptroutine, im Doppelregister DE die Anzahl der 20ms-Einheiten übergeben. Dabei ist darauf zu achten, daß die Interruptroutine Inhalte von Registern und Akkumulator sichert, bevor sie die Register verwendet. Vor dem Rücksprung ins unterbrochene Programm müssen die Register wieder geladen werden.

Außer in der oben beschriebenen Art kann der Timer auch zur automatischen Versorgung der Tageszeit benutzt werden. Zuvor müssen die folgenden Speicher hexadezimal mit den folgenden Werten belegt werden:

```
27B6H Monat
27B5H Jahr
27B7H Tag
27B8H Tageszähler (mit  1   vorbesetzt)
27B9H Stunde        "  24       "
27BAH Minute        "  60       "
27BBH Sekunde       "  60       "
27BCH 20ms-Zähler   "  50       "
```

Monat, Jahr und Tag werden nicht verändert, der Tageszähler kann theoretisch bis 255 gezählt werden.

Wird der 20ms-Zähler nicht verändert, werden (bis auf Monat, Tag und Jahr) alle Zähler ständig aktualisiert.

Im BS1MP steht eine Anwenderschnittstelle (CALL FE09H) zur Verfügung, mit der Uhrzeit und Datum gelesen werden:

Einsprung: HL: Adresse eines Ausgabebereichs
A : 0 - Datum und Uhrzeit
1 - Datum
2 - Uhrzeit

Aussprung: HL: Endadresse des Ausgabebereichs + 1
A : 38H und Carry=1 Ausgabefeld geloescht, da Datum und Uhrzeit nicht gesetzt wurden.

Zur Benutzung des Timers siehe Beispiel 2 in Anhang F.

Hinweis: Die 6.611 besitzt zwei verschiedene Timer, die gleichzeitig benutzt werden können.

Der Timer 2 wird im Folgenden beschrieben:

EXFUNC (~~0FDC4H~~ D3H)
SETTIM (01H) Setzen Timer

Parameter: A - Modus
0 - Ruhestellung, keine weiteren Unterbrechungen
1 - fortlaufende Unterbrechung alle xx ms
2 - einmalige Unterbrechung nach xx ms
C - Funktions-Code (01H)
DE - Anzahl der halben Millisekunden (2-FFFFH)
HL - Adresse der Unterbrechungsroutine

Zusätzlich zum 20ms Timer hat der Benutzer hiermit einen weiteren programmierbaren Timer.
Die Anzahl der Millisekunden kann zwischen 1ms und einer halben Minute sein. Wenn diese Funktion aufgerufen wird, und der Timer schon von einem anderen Benutzer besetzt ist, wird die Fehlermeldung 43H (Timer already in use) ausgegeben. Aus dem Benutzer-Unterbrecher-Handler wird in das unterbrochene Programm zurückgesprungen. Der Benutzer muß die Unterbrechung ermöglichen (EI).

Parameter zum Deaktivieren des Timers durch den Benutzer:

A - 0 (Modus Ruhestellung)
C - 01 (Funktions-Code)
HL- 00 (Adresse der Unterbrechungsroutine muß Null sein)

Beispiel:

```
EXFUNC  EQU     0FDC4H
SETTIM  EQU     01

Anfang: MVI     C,SETTIM  ;Funktions-Code
        LDA     MODUS     ;gewünschter Modus
        LXI     D,WERT    ;Zeitintervall
        LXI     H,TIMINT  ;Adresse der Unterbrechungsroutine
        CALL    EXFUNC
        JC      ERROR     ;kein Erfolg
```

9.4 Benutzung der Code-Tabellen

Im Betriebssystem BS1MP gibt es zwei Code-Tabellen, welche die Umwandlung von ASCII- in EBCDI-Code und umgekehrt ermöglichen.

Alle nicht nach ISO umsetzbaren EBCDIC-Werte (> 80H) werden nach SUB (1AH) konvertiert.

Die Adressen der Tabellen werden beim Laden des BS1MP auf den Stellen 277CH (ASCII in EBCDIC) und 277EH (EBCDIC in ASCII) eingetragen. Um ein Zeichen umzuwandeln, muß es binär zu der Basis-Adresse der jeweiligen Tabelle addiert werden, in der auf diese Weise adressierten Stelle befindet sich das Pendant des anderen Codes. Siehe hierzu Beispiel 3 im Anhang F.

Hinweis:

Um mit den Code-Tabellen arbeiten zu können, muß der Handler für ECMA-54-("IBM") Disketten (ISDHAN) geladen sein. Ist der Handler nicht geladen, sind die beiden Adreßspeicher 277CH und 277EH auf 0 gesetzt.

Wünscht der Anwender irgendwelche Änderungen in den Code-Tabellen, kann das Programm KONV (s. 2.15) verwendet werden.

9.5 Bildschirmattribute

9.5.1 Betriebsart ATTR21 (kompatibel zur 6.610):

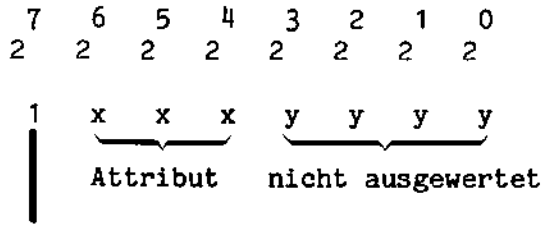
Die Bildschirmsteuerung erlaubt 6 verschiedene Darstellungsarten der Zeichen auf dem Bildschirm. Die Darstellung muß über ein Attributbyte gesteuert werden.

Prinzipieller Aufbau eines Textes auf dem Bildschirm:

Attributbyte T E X T Attributbyte

Die Attributbytes werden auf dem Bildschirm als Leerzeichen dargestellt.

Aufbau des Attributbytes



xxx	Bedeutung
000	nicht verwendet
001	nicht verwendet
010	Halbhelle Darstellung
011	Blinken
100	Invertierte Darstellung
101	Unterstrichene Darstellung
110	Dunkel (unsichtbar)
111	Normale Darstellung

Daraus ergibt sich die folgende Hex-Darstellung:

A0 = Halbhell
B0 = Blinken
C0 = Invers
D0 = Unterstrichen
E0 = Dunkel
F0 = Normal

Ein Attribut bleibt solange gültig, bis es durch ein neues aufgehoben wird oder das Bildschirmende erreicht ist.

9.5.2 Betriebsart ATTR23 (nicht kompatibel zur 6.610):

Das Attributbyte nimmt in dieser Betriebsart keinen Platz ein. Näheres hierzu ist der Bedienungsanleitung "Bildschirm-Computer 6.611" zu entnehmen.
~~Die verschiedenen Betriebszustände können durch folgende Hex-Code-Folgen eingestellt werden:~~

Betriebszustand	Hex-Code-Folge
Normal	~~0E 30 0F~~
Invertiert	~~0E 31 0F~~
Halbe Intensität	~~0E 32 0F~~
Halbe Intensität, invertiert	~~0E 33 0F~~
Unterstreichung	~~0E 34 0F~~
Unterstreichung, invertiert	~~0E 35 0F~~
Unterstreichung, Halbe Intensität	~~0E 36 0F~~
Unterstreichung, Halbe Intensität, invertiert	~~0E 37 0F~~
Blinken Normal/Aus	~~0E 38 0F~~
Blinken Normal/invertiert	~~0E 39 0F~~
Blinken Normal/Halbe Intensität	~~0E 3A 0F~~
Blinken Halbe Intensität/ Halbe Intensität; invertiert	~~0E 3B 0F~~
Blinken Normal/Unterstreichung	~~0E 3C 0F~~
Blinken Unterstreichung/ Unterstreichung invertiert	~~0E 3D 0F~~
Dehnung	~~0E 3E 0F~~
Unsichtbar	~~0E 3F 0F~~ 0F
~~Rücksetzen der Attributfolge~~	~~0E 40 0F~~

Anmerkung: Vor jeder Zeichenfolge mit dem Attribut "Dehnung" muß ein Leerzeichen eingefügt werden. Außerdem muß das erste Zeichen einer Zeile mit "Dehnung" ein Leerzeichen sein.

9.6 Programmierung der Software-Schalter im Menü + Funktionstaster

Die Software-Schalter der 6.611 können von der Tastatur geändert werden (s. Bedienungsanleitung "Bildschirm-Computer 6.611"). Alle Schalter können vom Programm gelesen und mit Ausnahme von Menü 1, temporär neu programmiert werden.

Im Anhang G befinden sich eine numerische und eine alphabetische Liste der möglichen Software-Schalter sowie die Belegungscodes der einzelnen Software-Schalter.

Mit den Monitorroutinen DCWRIT (Schreiben über Bildschirmsteuerung) und DCREAD (Lesen von Bildschirmsteuerung) können die Software-Schalter (wie unter 5.2 beschreiben) programmiert werden.

Folgende Funktionen sind zur Programmierung der Software-Schalter nötig:

SSWROP (37H) - Öffnen Software-Schalter zur Umstellung.

Eingabe: Nummer des Software-Schalters
Ausgabe: keine

Der angegebene Software-Schalter wird zur Umstellung eröffnet, d. h. es ist jetzt möglich, den Wert des Software-Schalters per Programm durch den Aufruf von SSWR (s. unten) zu ändern. Wenn schon ein anderer Software-Schalter zur Umstellung eröffnet war, wird dieser geschlossen und der neue eröffnet.

SSWR (38H) - Umstellung Software-Schalter

Eingabe: Software-Schalterstellung
Ausgabe: keine

Diese Funktion kann nur durchgeführt werden, wenn vorher SSWROP aufgerufen wurde. Der ausgewählte Schalter erhält die angegebene Schalterstellung (wenn zulässig) und wird dann geschlossen. Wenn sich die neue Schalterstellung von der früheren unterscheidet, wird diese temporär wie über das Menü festgeschrieben.

SSRD (A9H) - Lesen Software-Schalter

Eingabe: Nummer des Software-Schalters
Ausgabe: aktuelle Schalterstellung

Die aktuelle Schalterstellung des gewünschten Software-Schalters wird ausgegeben. Wenn der gewünschte Software-Schalter nicht im-

plementiert ist, wird bei dieser Funktion 0 ausgegeben.

Hinweis: Wenn der Wert eines Schalters per Programm geändert wird, ist nur die temporäre Schalterstellung betroffen. Um eine permanente Schalterstellung zu erreichen, muß der Bediener auf das Menü über Tastatur zugreifen.

TRANSDATA 920-Kompatibilität

Auf der 6.611 können Disketten erstellt und verarbeitet werden, die - mit gewissen Einschränkungen - kompatibel sind zu denen, die auf dem Erfassungsplatz TRANSDATA 920 bzw. IBM 3740 benutzt werden (ECMA-54-Format).
Diese Disketten werden auf der 6.611 mit dem BS1MP durch

FORMAT$:Jn: Name ,I (n = Laufwerknummer)

initialisiert.

Beim Initialisieren wird automatisch eine Datei DATA eingerichtet. Weitere Dateien können mit dem Programm ALLOC erstellt werden.

Dateien auf 920/3740-kompatiblen Disketten werden mit :Jn: Dateiname angesprochen.

Bei Eingabe der physikalischen Einheit :Jn: findet automatische Code-Umwandlung von ASCII in EBCDIC statt, da TRANSDATA 920 bzw. IBM 3740 im EBCDIC-Code arbeiten.

Folgende Dienstprogramme können auf ECMA-54-Floppies angewandt werden:

ALLOC, COPY, DELETE, DIR, EDIT und RENAME

Mit dem Editor EDIT kann man direkt auf 920/3740-kompatible Disketten schreiben, hat aber zu berücksichtigen, daß die 920 bzw. 3740 nur Großbuchstaben kennt, während der Editor auch Kleinbuchstaben zuläßt.

Die 6.611 verarbeitet alle abdruckbaren Zeichen der 920/3740 mit folgenden Ausnahmen:

#	wird auf der 6.611 als £ dargestellt
C	(Cent-Zeichen) bewirkt auf der 6.611 Invertierung des Bildschirms

In der folgenden Übersicht wird dargestellt, welche Eintragungen im Daten- bzw. Dateiträgerkennsatz auf der 6.611 möglich sind und welche ignoriert werden.

I = Bedeutung wie bei 920/3740

II = Auf der 6.611 veränderbar, aber ohne Bedeutung

III = Auf der 6.611 nicht beeinflußbar und ohne Bedeutung

	I	II	III
Datenträgerkennsatz			
Datenträgerkenn-satzinformation			x
Zugriffsschutzfeld			x
physik. Sektorlänge			x
Sektorfolge			x
Dateikennsatz			
Log.Satzlänge		x	
Bereichsbeginn (BOE)	x		
Bereichsende (EOE)	x		
Übergehen der Datei			x
Zugriffsmöglichkeit			x
Schreibschutz		x	
Mehrdiskettendatei-anzeiger			x
Fortlaufende Nummer			x
Schreibdatum	x		
Freigabedatum		x	
Prüfzeichen			x
Datenende	x		

BS1M-Disketten-Format

BS1M-Disketten sind zweiseitig beschriebene Datenträger mit einer Speicherkapazität von 1.021.696 Bytes:

Seite 0: Spur 0 mit 26 Sektoren à 128 Bytes
Spuren 1 bis 76 mit je 26 Sektoren à 256 Bytes

Seite 1: Spur 0 bis 76 mit 26 Sektoren à 256 Bytes.

Zieht man die verschiedenen reservierten Bereiche und Systemdateien (s.u.) ab, so stehen dem Anwender 3902 Sektoren mit jeweils 256 Bytes, zusammen also 998.912 Bytes zur Verfügung. Dabei ist zu berücksichtigen, daß das Disketten-Inhaltsverzeichnis zusätzlichen Platz belegt, wenn man mehr als 160 Dateien anlegt. Insgesamt können vom Benutzer 1024 Dateien angelegt werden.

Hinweis: Der Platz, den eine Datei im Inhaltsverzeichnis belegt, wird durch ihre Löschung nicht automatisch freigegeben. Daher muß daß Verzeichnis gelegentlich mit dem Programm DIRPAC reorganisiert werden.

Aufbau der Disketten

Seite	Spur	Sektor	
0	0		reserviert
1	0		reserviert
0	1	1 - 2	Belegungsverzeichnis
0	1	3 - 5	Verweise auf Fehlermeldungs- und Inhaltsverzeichnis
0	1	6 - 25	Inhaltsverzeichnis für die ersten 160 Dateien
0	1	26	Datenträgerkennsatz
0	2	1	Datenbereich
	.		
	.		
0	76	26	
1	1	1 - 24	Fehlermeldungen
1	1	25	Datenbereich
	.		
	.		
1	76	26	

Wird eine Datei angelegt, so wird ihr Inhalt in eine Kette von Sektoren abgelegt. Es werden dabei immer ganze Sektoren für diese Datei belegt.

Die Information über die Verkettung - um die sich der Anwender nicht kümmern muß - steht in Dateikettungssektoren. Pro 62 Datensektoren - rund 15800 Bytes - wird ein Kettungssektor benötigt.

INTEL-Disketten-Format

INTEL-Disketten sind einseitig beschriebene Datenträger mit einer Speicherkapazität von 256.256 Bytes (77 Spuren mit jeweils 26 Sektoren mit je 128 Bytes).

55 Sektoren mit 7040 Bytes werden für die Systemdateien ISIS.ERR (Fehlerverzeichnis), ISIS.MAP (Belegungsverzeichnis), ISIS.DIR (Inhaltsverzeichnis) und ISIS.LAB (Datenträgername) benötigt. Auf einer formatierten, jedoch noch unbenutzten Diskette stehen dem Anwender also 1947 Sektoren zur Verfügung.

Auf einer INTEL-Diskette können maximal 196 Dateien vom Anwender angelegt werden.

Aufbau der Diskette

Spur	Sektor	
0	1	Datenbereich
0	25	(vom BS1MP benutzt für Systemfehlermeldungen)
0	26	Datenträger-Kennsatz
1	1	Inhaltsverzeichnis
1	26	der Diskette
2	1	Belegungsverzeichnis
2	3	
2	4	Datenbereich
.	.	
.	.	
76	26	

Dateien werden genauso angelegt wie auf BS1M-Disketten, jedoch wird bei diesem Format für jeweils ca. 7900 Bytes (62 Sektoren a 128 Bytes) ein Kettungsblock angelegt.

Anhang D

ECMA 54-Disketten-Format

Genauere Informationen können der Beschreibung des Siemens-Datenerfassungssystems 920 entnommen werden.

Aufbau der Diskette

Spur	Sektor	
0	1	Reserviert
0	4	
0	5	ERMAP
0	6	Reserviert
0	7	Datenträger-Kennsatz
0	8	Dateikennsätze
0	26	
1	1	Datenbereich
74	26	
75	1	Ersatzspur 1
75	26	
76	1	Ersatzspur 2
76	26	

Im folgenden sind nur Felder beschrieben, welche bei der Benutzung ECMA 54- ("IBM"-) formatierter Disketten im BS1M relevant sind. Nicht beschriebene Felder sind als reserviert zu betrachten.

ERMAP

Spur 0 Sektor 5

Stelle

1- 5	ERMAP
7- 8	Leer oder Nummer der ersten defekten Spur. Dezimal!
11-12	Leer oder Nummer der zweiten defekten Spur.

Datenträger-Kennsatz

Spur 0 Sektor 7

Stelle

1 - 4	VOL1
5 - 10	Datenträger-Kennsatzinformation
11	Zugriffsschutzfeld)*
38 - 51	Eigentümerinformation)*
76	Phy. Sektorlänge der Spuren 1 - 7)*
	leer - 128 Bytes
	1 - 256 Bytes
	2 - 512 Bytes
77 - 78	Phys. Sektorfolge

Dateikennsatz

Spur 0 Sektoren 8 - 26, pro Dateikennsatz ein Sektor.
Kennsatzaufbau:

Stelle

1 - 4	HDR1 (Benutzer-Anfangsetikett)
6 - 13	Dateiname
29 - 33	Bereichsbeginn (BOE), erster zu einer Datei gehörender Sektor
29 - 30	Spurnummer, 32 - 33 Sektornummer
34	Phys. Satzlänge)*
	Vergl. Stelle 76 des Datenträger-Kennsatzes
35 - 39	Bereichsende (EOE), letzter für diese Datei reservierte Sektor, Format wie BOE
42	Zugriffsschutzfeld)*
43	Schreibschutz
	Leerfeld - ungeschützt
	P - geschützt
48 - 53	Schreibdatum JJMMTT
54 - 57	Satzlänge)*
67 - 72	Freigabedatum JJMMTT
75 - 79	Datenende (EOD), Adresse des nächsten unbelegten Sektors Format wie BOE

)* vorläufig nicht benutzt

Anhang E

Zusammenhang zwischen den E/A-Routinen im BS1MP

Die Ein- und Ausgaberoutinen lassen sich in drei Gruppen untergliedern:

- Zeichenebene:
  INCHAR
  OUTCHAR

- Zeilen/Gruppenebene:
  GET
  GETBIN (BINFILE=1)
  PUT
  PUTBIN (BINFILE=1)
  PUTSTR

- Blockebene:
  READ
  WRITE

Die Arbeitsweise dieser Routinen werden durch zwei Kontroll-Bits modifiziert, die im FCB (Dateikontrollblock) gesetzt werden.

Die beiden Kontroll-Bits werden mit BINFILE und BLOCKLINE bezeichnet.

Ihre Positionen sind (Byte 0 ist das erste Byte es FCB):

BINFILE	Bit 2 in Byte 1 des FCB (Bit 0 ist das niedrigstwertige Bit)
BLOCKLINE	Bit 6 in Byte 27 des FCB

BINFILE und BLOCKLINE können nur von einem Assemblerprogramm aus gesetzt werden.

In der folgenden Aufstellung wird die Wirkung von BINFILE und BLOCKLINE auf die Routinen INCHAR und OUTCHAR dargestellt:

Routine	BINFILE	BLOCKLINE	Bedeutung
INCHAR	0	0	Alle 00H und 0AH (LF) werden beim Einlesen unterdrückt.
	0	1	Wird ein End-of-Block erkannt, wird es durch CR ersetzt. 0AH (LF) und 00H werden unterdrückt.
	1	0	Es werden keine Zeichen unterdrückt.
	1	1	Bei End-of-Block wird Fehler-Code 5C (End-of-Block) gegeben.
OUTCHAR	0	0	Alle Zeichen werden ausgegeben.
	0	1	Beim Auftreten von CR wird CR unterdrückt, der Rest des Blokkes wird mit Nullen gefüllt, der Block wird weggeschrieben.
	1	0	Alle Zeichen werden ausgegeben.
	1	1	wie (0,1).

Die Routinen GET, GETBIN, PUT, PUTSTR und PUTBIN lassen sich jetzt mit Hilfe der Routinen INCHAR und OUTCHAR darstellen. Für diese Darstellung wird hier eine Art höherer Programmiersprache benutzt.

```
procedure GET(A); address A;
          begin character C;
                    repeat    begin
                              C:=INCHAR;
                              memory(A):=C
                              A:=A+1
                              end
                    until C=CR
          end
```

```
procedure GETBIN(A,N); address A; Byte N
          begin Bit B;
                    B:=BINFILE;
                    BINFILE:=1;
                    repeat    begin
                              memory(A):=INCHAR;
                              A:=A+1;
                              N:=N-1;
                              end
                    until N=0;
                    BINFILE:=B;
          end;


procedure PUTSTR(A);address A;
          begin Byte C;
                    repeat    begin
                              C:=memory(A);
                              if C=CR then exit;
                              OUTCHAR(C);
                              A:=A+1;
                              end;
          end;


procedure PUTBIN(A,N); address A;Byte N;
          begin Byte C;
                    repeat    begin
                              C:=memory(A);
                              OUTCHAR(C);
                              N:=N-1;
                              A:=A+1;
                              end;
                    until N=0;
          end;


procedure PUT(A); address A;
          begin
                    PUTSTR(A);
                    OUTCHAR(CR);
                    OUTCHAR(LF);
          end
```

Die Routinen READ und WRITE arbeiten unabhängig von BINFILE und BLOCKLINE, da sie auf Blockbasis arbeiten.

Beispiele zu Kapitel 9

Beispiel 1

Starten eines Programmes innerhalb eines Programms

Das Root-Programm fragt nach der Nummer des Overlays, das nachgeladen werden soll. Die Overlays geben eine Meldung aus und springen in den Root zurück.

Die assemblierten Module ROOT.HEX (Root-Programm), SCROP.HEX (Subroutine für Zeichenausgabe), OV1.HEX (Overlay 1) und OV2.HEX (Overlay 2) werden folgendermaßen gebunden:

Rootprogramm

```
LINK SO=ROOT,SL=:LP:
*C=4000
ROOT.HEX
SCROPT.HEX
/
```

Die Overlaymodule müssen in diesem Fall nicht im Binderlauf berücksichtigt werden, weil das Rootprogramm sich auf keine Adressen in Overlays bezieht und die Overlays hinter das Rootprogramm geladen werden.

Overlayprogramme

```
LINK SO=OV1,SL=:LP:
*C=4000
*X
ROOT.HEX
SCROP.HEX
*X
OV1.HEX
/
```

und

```
LINK SO=OV2,SL=:LP:
*C=4000
*X
ROOT.HEX
SCROPT.HEX
*X
OV2.HEX
/
```

In dem Beispiel ist nur Overlay 2 gezeigt.

Root-Segment

```
$TITLE ****   ROOT - OVERLAY LINK TEST : ROOT - TEIL ****
;*
;***      PUBLIC/EXTRN für ROOT/OV1/OV2
;*
          EXTRN     SCROPT    ;Unterprogramm Bildschirmausg.
          PUBLIC    ID        ;Datenbereich für OV2
          PUBLIC    ROOT      ;Rücksprungadresse für OVx
;*
;**       Feste Zuordnungen
;*
LOAD      EQU       0FD06H    ;Laderoutine
TTI       EQU       40H       ;Tastatureingabe
TTO       EQU       43H       ;Bildschirmausgabe
CR        EQU       0DH       ;CR
LF        EQU       0AH       ;LF
;
;*
;**       Speicherresidenter Rootteil (mit SCROPT)
;*
ROOT:     CALL      SCROP     ;Ausgabe Startmeldung
          DW        MSG1
          CALL      TTI       ;Antwort
          CALL      TTO       ;
          CPI       '1'       ;1?
          JZ        LDOV1     ;Ja: --  1 (Laden OV1)
          CPI       '2'       ;2?
          JZ        LDOV2     ;Ja: --  2 (Laden OV2)
          CPI       'e'
          JZ        ROOT99    ; --  99 wenn 'e' (Ende)
          CPI       'E'
          JZ        ROOT99    ; --  99 wenn 'E' (Ende)
          CALL      SCROPT
          DW        EMSG      ;Fehlermeldung
          JMP       ROOT
;*
;*****    Programmende einleiten ******
;*
ROOT99:   CALL      SCROPT    ;Endemeldung ausgeben
          DW        MSG2
          ORA       A         ;Cy zurücksetzen
          RET                 ;zurück ins BS1MP
;*
;***** Vorbereitung: Laden OV1  *****
;*
LDOV1:    LXI       H,OV1     ;Datennamensadresse
          JMP       LDOV      ;Laden Overlay
;*
;****  Vorbereitung: Laden OV2  *****
;*
LDOV2:    LXI       H,OV2     ;Datennamensadresse
;*
;**       Laden eines Overlays von Laufwerk 0 ****
;*
```

```
LDOV:     PUSH    H          ;Sichern Datennamensadresse
          CALL    SCROPT     ;Neue Zeile auf Bildschirm
          DW      CRLF
          POP     H          ;Dateinamensadresse
          XRA     A          ;Gerätenummer = 0
          CALL    LOAD       ;Laden des Overlays
          RC                 ;--  BS1M wenn Fehler
          PCHL               ;Overlay starten
;*
;***** Text- und Variablenbereich
;*
CRLF:     DB      CR,LF,-1
MSG1:     DB      'LINK-ROOT-OVERLAY TEST: CALL OVn'
          DB      ' (1/2/E=Ende)? ',-1
MSG2:     DB      CR,LF,'ROOT Ende',-2
EMSG:     DB      CR,LF,'Falsche Eingabe',-2
OV1:      DB      'OV1',0,0,0,0,0,0,0,0,0,0
OV2:      DB      'OV2',0,0,0,0,0,0,0,0,0,0
          DSEG
ID:       DB      'Overlay: ',CR      ;Wird in OV2 benutzt
          END     ROOT
```

Overlay 2

```
$TITLE ****   ROOT-OVERLAY LINKER TEST: Overlay 2 ****
;*
;**       EXTRN-Deklarationen für Teile, die im Root liegen
;*
          EXTRN   SROPT      ;Bildschirmausgabe
          EXTRN   ID         ;Text: "Overlay:" im ROOT
          EXTRN   ROOT       ;Rücksprung
;*
;**       Feste Datendeklarationen
;*
CR        EQU     0DH        ;CR
LF        EQU     0AH        ;LF
;*
;**       Start Overlay 2
;*
OV2:
;*        Übertragen ID aus ROOT (DSEG)
          LXI     H,ID       ;ID aus ROOT
          LXI     D,MSG1     ;MSG-Adresse
MOVE:     MOV     A,M
          CPI     CR         ;Endekennung?
          JZ      OVAUSG     ;Ja --  OVAUSG
          STAX    D          ;Nein -- Zeichen speichern
          INX     H
          INX     D
          JMP     MOVE
OVAUSG:   CALL    SCROPT     ;Meldung ausgeben
          DW      MSG1
          JMP     ROOT       ;Rücksprung ins Root
```

```
;*
;**      Textbereich
;*
MSG1:   DB      '******** 2 gestartet',-2
        END     OV2         ;Startadresse
```

Beispiel 2

Benutzung des Timers

Das Programm zählt nach dem Start einen Sekundenzähler hoch, der bis zehn zählt und dann wieder mit 0 beginnt. Das Programm wird durch eine beliebige Tastatureingabe beendet.

```
$title *****     BEISPIEL FÜR BS1MP-STIMER-SCHNITTSTELLE     *****
$pw 110
$pl 48
stime:    cseg
tto       equ       43h;      Bildschirmausgabe
ttinw     equ       0DCh;     Tastatureingabe (ohne Warten)
stimer    equ       0fd3ch;   BS1MP-STIMER-Schnittstelle
clr       equ       19h;      Löschen Bildschirm
;*
;** Hauptprogramm
;*
anfang:   ds        0;        Programm-Start
;
;*****     Laden des Timers     *****
;
          lxi       h,timer;  Timer Programm Teil
          lxi       d,50;     50 x 20 ms = 1 s
          call      stimer;   Intervall setzen
;
;*****     Meldung ausgeben     *****
;
ausg:     lxi       h,mld;    H/L = Adresse der Meldung
ausgs:    mov       a,m;      A = Zeichen
          ora       a;        Zeichen = 0 ?
          jz        warten;   JA --  WARTEN (Meldung ausgegeben)
          call      tto;      Zeichen --  Bildschirm
          inx       h;        Adresse +1
          jmp       ausgs;    --  Schleife
;
;*****     Hauptprogramm : Verarbeitung     *****
;
warten:   call      ttinw;    Zeichen im Puffer
          jc        warten;   Nein
          jmp       ende;     --  ENDE
$e
```

```
;*
;** Ablauf bei Anstoß nach Zeitintervallablauf
;*
timer:    lda       mldc;     A = Zähler
          inr       a;        Zähler +1
          cpi       3ah;      Zähler = 10
          jnz       timerw;   JA --  WEITER
          mvi       a,'0';    A = 0 setzen
timerw:   sta       mldc;     Zähler abspeichern
          lxi       h,timer
          lxi       d,50
          call      stimer;   Intervall wieder setzen
          lxi       h,ausg;   H/L = Rücksprungsadresse
          xthl      ;         Rücksprung --  STACK
          ret       ;         Rücksprung
;*
;** Programm Ende
;*

ende:     ds        0;        Ende des Programms
          mvi       a,clr
          call      tto;      Bildschirm löschen
          ora       a;        CY rücksetzen
          ret       ;         --  BS1MP
;

;*****     Texte     *****
;
mld:      db        clr,'Zeitgeber : Zähler = '
mldc:     db        '0',0
          end       anfang
```

Beispiel 3

Benutzung der Code-Tabelle

Das Programm erwartet Eingaben in der Form Axx bzw. Exx, wobei xx ein Hex-Code ist. Der Hex-Code wird im ersten Fall von ASCII- in EBCDI-Code, im zweiten Fall von EBCDI- in ASCII-Code umgewandelt und hexadezimal ausgegeben.
Hinweis: Vor der Benutzung der Tabellen muß der "IBM"-Handler geladen worden sein.

```
$title *****     Beispiel : Codeumsetzung (USASCII - EBCDIC)*****
$pw 110
$pl 48
;*
;*              Codeumsetzung über BS1MP-Codetabellen          *
;*                                                             *
;*************************************************************
;
;
;*
;** Equates
;*
adrebc    equ       277ch
adrusa    equ       277eh
lf        equ       0ah
cr        equ       0dh
clr       equ       19h
;*
;** Externals
;*
extrn     srcopt;   Bildschirm-Ausgabe
extrn     srcipt;   Bildschirm-Eingabe
extrn     srcvah;   Umsetzung ASCII ---> HEX
extrn     srcvha    Umsetzung HEX ---> ASCII
$e
;*
;** Eingabe- und Analyse Programmteil
;*
anfang:   ds        0;        Programmstart
          call      srcopt;   Anfangsmeldung ausgeben
          dw        msg0
eingab:   call      srcopt;   Eingabeaufforderung ausgeben
          dw        msg1
          call      srcipt;   max. 4 Zeichen eingeben
          db        4
          dw        eber
          ora       a;        Zeichen eingegeben ?
          jz        efehl1;   NEIN ---> Fehler 1
          dcr       a;        Zeichenzahl -1
          jnz       a1;       --->1 wenn mehr als 1 Zeichen
```

```
;
;*****     Test ob Programmende     *****
;
          lda       eber;     A = Zeichen
          call      caplet;   Umsetzen in Großbuchstaben
          cpi       'H';      Zeichen = H ?
          jnz       efehl2;   NEIN ⟶ Fehler 2
;
;*****     Programmende     *****
;
          call      srcopt;   Endemeldung ausgeben
          dw        msg4
          ora       a;        CY rücksetzen
          ret       ;         ⟶ BS1MP

;
;*****     Mehr als 1 Zeichen eingeben     *****
;
a1:       dcr       a
          dcr       a;        Zeichenzähler insgesamt -3
          jnz       efehl1;   --  Fehler 1 wenn    3 Zeichen
;
;*****     3 Zeichen eingegeben (Format : Kennung und Code)*****
;
          lda       eber;     A = 1. Zeichen
          call      caplet;   Umsetzen in Großbuchstaben
          cpi       'A';      Zeichen = A ?
          jz        aseb;     JA --  ASEB (USASCII --  EBCDIC)
          cpi       'E';      Zeichen = E ?
          jz        ebas;     JA --  EBAS (EBCDIC --  USASCII)
          jmp       efehl2;   NEIN --  Fehler 2
$e
;*
;** Umsetzung USASCII --  EBCDIC
;*
aseb:     ds        0;        Umsetzung ASCII
          lxi       h,eber+1; H/L = Adresse der USASCII-Zeichen
          call      srcvah;   Umsetzung USASCII --  HEX
          jc        efehl3;   --  Fehler 3 wenn CY = 1
          mov       c,a
          mvi       b,0;      B/C = Distanz innerhalb der Umsetz-
                              ;tabelle
          lhld      adrebc    H/L = Anfangsadresse der Codetabelle
          dad       b;        H/L = Adresse des umgesetzten
                              ;Zeichens
          mov       a,m;      A = HEX-WERT des umgesetzten Zeichens
          lxi       h,msg2z
          call      srcvha;   Umsetzen in abdruckbare Hex-Zeichen
          call      srcopt;   Umgesetztes Zeichen ausgeben
          dw        msg2
          jmp       eingab;   --  EINGAB (nächsten Wert eingeben)
```

```
;*
;** Umsetzung EBCDIC --  USASCII
;*
ebas:
          ds        0;          Umsetzung EBCDIC --  USASCII
          lxi       h,eber+1;   H/L = Adresse der USASCII-Zeichen
          call      srcvah;     Umsetzen USASCII --  HEX
          jc        efehl3;     --  Fehler 3 wenn CY = 1
          mov       c,a
          mvi       b,0;        B/C = Distanz zum Anfang der
                                ;Codetabelle
          lhld      adrusa;     H/L = Anfangsadresse der
                                ;USASCII-Tabelle
          dad       b;          H/L = Adresse des umzusetzenden
                                ;Zeichens
          mov       a,m;        A = Umgesetztes Zeichen
          lxi       h,msg3z
          call      srcha;      Umsetzen in abdruckbare Hex-Zeichen
          call      srcopt;     Umgesetztes Zeichen ausgeben
          dw        msg3
          jmp       eingab;     --  EINGAB (neuen Wert eingeben)

;*
;** Fehler 1 : Formatfehler
;*
efehl1:   call      srcopt;     Fehlermeldung 1 ausgeben
          dw        emsg1
          jmp       eingab;     --  EINGAB (Eingabe wiederholen)
;*
;** Fehler 2 : Falscher Kennbuchstabe
;*
efehl2:   call      srcopt;     Fehlermeldung 2 ausgeben
          dw        emsg2
          jmp       eingab;     --  EINGAB (Eingabe wiederholen)
;*
;** Fehler 3 : Keine USASCII-HEX-Zeichen
;*
efehl3:   call      srcopt;     Fehlermeldung 3 ausgeben
          dw        emsg3
          jmp       eingab;     --  EINGAB (Eingabe wiederholen)
$e
;*
;** Unterprogramm Umwandlung in Großbuchstaben
;*
caplet:   cpi       60h;        Großbuchstabe ?
          rc        ;           JA --  Rücksprung
          ani       5fh;        NEIN : Umwandeln in Großbuchstabe
          ret       ;           Rücksprung
```

```
;*
;** Texte
;*
msg0      db  clr,'Bsp:Codeumsetzung über BS1MP-Tabellen',cr,lf,0
msg1:     db        'Wert eingeben (H/Axx/Exx) ',0
msg2:     db        'EBCDIC-Wert = '
msg2z:    db        '**',cr,lf,0
msg3:     db        'USASCII-Wert = '
msg3z:    db        '**',cr,lf,0
msg4      db        'Programmende',cr,lf,o
emsg1:    db        'Eingabe falsches Format',cr,lf,0
emsg2:    db        'Falscher Kennbuchstabe',cr,lf,0
emsg3:    db        'Keine Hexadezimalzeichen',cr,lf,0
;*
;** Bereiche
;*
eber:     ds        4;        Eingabebereich
          end       anfang
```

Codes der Software-Schalter in aufsteigender Reihenfolge

Hex Code	Dezimal Code	Programmierbar (-=nein, +=ja)	Kurz-zeichen	Name d. Software-Schalter
00h	0	-	CT	Cursor Type
01h	1	-	CB	Cursor Blink
02h	2	-	KC	Key Click
03h	3	-	MB	Margin Bell
04h	4	-	AR	Auto Repeat
05h	5	+	TIM	Video Timeout
06h	6	+	BEL	Bell
07h	7	+	AS	Attribute System
08h	8	+	EOL	End of Line Wrap
09h	9	+	RPM	Roll/Page Mode
0Ah	10	-	RT	Roll Type
0Bh	11	+	CLL	Clear Lamps
0Ch	12	+	CAR	Char./Attr. Relation
0Dh	13	+	TKM	Transparent Keyboard Mode
0Eh	14	+	ONL	Online
0Fh	15	+	CCA	Communication Clock, ch.A
10h	16	+	MOD	Modem
11h	17	+	TCLA	Transm. Code Lenght, ch.A
12h	18	+	TCPA	Transm. Code Parity, ch.A
13h	19	+	TCSA	Transm. C. Stop Bits,ch.A
14h	20	+	TSA	Transm. Speed, ch.A
15h	21	+	OIA	Optional Interface, ch.A
16h	22	+	CCB	Communication Clock, ch.B
17h	23	+	TCLB	Transm. Code Lenght, ch.B
18h	24	+	TCPB	Transm. Code Parity, ch.B
19h	25	+	TCSB	Transm. C. Stop Bits,ch.B
1Ah	26	+	TSB	Transm. Speed, ch.B
1Bh	27	+	OIB	Optinal Interface, ch.B
1Ch	28	+	MDT	Monitor Device Type
1Dh	29	+	MU	Monitor Unit
1Eh	30	+	SS	System Select
1Fh	31	+	SDT	System Device Type
20h	32	+	SU	System Unit

Hex Code	Dezimal Code	Programmierbar (-=nein,+=ja)	Kurz-zeichen	Name d.Software-Schalter
21h	33	+	UDS1	User Defined Switch no.1
22h	34	+	UDS2	User Defined Switch no.2
23h	35	+	UDS3	User Defined Switch no.3
24h	36	+	UDS4	User Defined Switch no.4
28h	40	-	ROV	Key Rollover
2Ah	42	-	DOT9	Dot 9
2Dh	45	+	BPU	Boot Power up
2Eh	46	-	BREAK	Break
2Fh	47	+	KAM	Keyboard Action Mode
30h	48	+	ME	Menu Entrance

Codes der Software-Schalter in alphabetischer Reihenfolge

Name der Software-Schalter	Kurz-zeichen	Hex Code	Dezimal Code	Programmierbar (-=nein,+=ja)
Attribute System	AS	07h	7	+
Auto Repeat	AR	04h	4	-
Bell	BEL	06h	6	+
Boot Power Up	BPU	2Dh	45	+
Break	BREAK	2Eh	46	-
Char./Attr. Relation	CAR	0Ch	12	+
Clear Lamps	CLL	0Bh	11	+
Communication Clock, ch.A	CCA	0Fh	15	+
Communication Clock, ch.B	CCB	16h	22	+
Cursor Blink	CB	01h	1	-
Corsor Type	CT	00h	0	-
Dot 9	DOT9	2Ah	42	-
End of Line Wrap	EOL	08h	8	+
Key Click	KC	02h	2	-
Key Rollover	ROV	28h	40	-
Keyboard Action Mode	KAM	2Fh	47	+
Margin Bell	MB	03h	3	-
Menu Entrance	ME	30h	48	+
Modem	MOD	10h	16	+
Monitor Device Type	MDT	1Ch	28	+
Monitor Unit	MU	1Dh	29	+
Online	ONL	0Eh	14	+
Optional Interface, ch.A	OIA	15h	21	+
Optional Interface, ch.B	OIB	1Bh	27	+
Roll Type	RT	0Ah	10	-
Roll/Page Mode	RPM	09h	9	+
System Device Type	SDT	1Fh	31	+
System Select	SS	1Eh	30	+
System Unit	SU	20h	32	+

Name der Software-Schalter	Kurz-zeichen	Hex Code	Dezimal Code	Programmierbar (-=nein,+=ja)
Transmission Code Lenght,ch.A	TCLA	11h	17	+
Transmission Code Lenght,ch.B	TCLB	17h	23	+
Transmission Code Parity,ch.A	TCPA	12h	18	+
Transmission Code Parity,ch.B	TCPB	18h	24	+
Transm. Code Stop Bits, ch.A	TCSA	13h	19	+
Transm. Code Stop Bits, ch.B	TCSB	19h	25	+
Transmission Speed, ch.A	TSA	14h	20	+
Transmission Speed, ch.B	TSB	1Ah	26	+
Transparent Keyboard Mode	TKM	0Dh	13	+
User Defined Switch no. 1	UDS1	21h	33	+
User Defined Switch no. 2	UDS2	22h	34	+
User Defined Switch no. 3	UDS3	23h	35	+
User Defined Switch no. 4	UDS4	24h	36	+
Video Timeout	TIM	05h	5	+

Belegung der Software-Schalter

\- bedeutet nicht programmierbar
\+ bedeutet programmierbar
Unterstreichung bedeutet Standardeinstellung

Convenience Switches

Cursor Type	0	Line
CT 00 -	1	Block
Cursor Blink	0	Blink
CB 01 -	1	Steady
Dot 9	0	= Blank
DOT9 42 -	1	= Dot 8
Roll Type	0	Step
RT 10 -	1	Smooth
Key Click	0	On
KC 02 -	1	Off
Key Rollover	0	Enabled
ROV 40 -	1	Disabled
Auto Repeat	0	On
AR 04 -	1	Off
Margin Bell	0	On
MB 03 -	1	Off

Function switches

Video Timeout	0	ON
TIM 05 +	1	Off
Bell	0	On
BEL 06 +	1	Off
Attribute System	0	Attr21
AS 07 +	1	Underline
	2	Attr23
End of Line Wrap	0	Stop
EOL 08 +	1	Wrap
Roll/Page Mode	0	Roll
RPM 09 +	1	Page
Clear Lamps	0	Both
CLL 11 +	1	Key
	2	Syn
Char./Attr. Relation	0	Both
CAR 12 +	1	Character
	2	Attribute
Transp.Keyboard Mode	0	No
TKM 13 +	1	Yes

Communication switches channel A

Modem	0	Leased
MOD 16 +	1	Disabled
Online	0	Online
ONL 14 +	1	Toggle
Communication Clock	0	Internal
CCA 15 +	1	External
Transm. Code Length	0	7bit
TCLA 17 +	1	8bit
Transm. Code Parity	0	None
TCPA 18 +	1	Even
	2	Odd
Transm. Code Stop Bits	0	1bit
TCSA 19 +	1	2bit
Transmission Speed	0	50 Baud
TSA 20 +	1	75 Baud
	2	110 Baud
	3	134,5 Baud
	4	200 Baud
	5	300 Baud
	6	600 Baud
	7	1200 Baud
	8	2400 Baud
	9	4800 Baud
	10	9600 Baud
	11	19200 Baud
Optional Interface,ch.A	0	No
OIA 21 +	1	Yes

Communication switches channel B

Communication Clock	0	Internal
CCB 22 +	1	External
Transm. Code Length	0	7bits
TCLB 23 +	1	8bits
Transm. Code Parity	0	None
TCPB 24 +	1	Even
	2	Odd
Transm. Code Stop Bits	0	1 bit
TCSB 25 +	1	2 bits
Transmission Speed	0	50 baud
TSB 26 +	1	75 baud
	2	110 baud
	3	134,5 baud
	4	200 baud
	5	300 baud
	6	600 baud
	7	1200 baud
	8	2400 baud
	9	4800 baud
	10	9600 baud
	11	19200 baud
Optional Interface	0	No
OIB 27 +	1	Yes

Load Switches

Monitor Device Type	0	F
MDT 28 +	1	D
	2	H
Monitor Unit	0	Unit 0
MU 29 +	1	Unit 1
	2	Unit 2
	3	Unit 3
System Select	0	ZZMON
SS 30 +	1	BIOS
Sytem Device Type	0	F
SDT 31 +	1	D
	2	H
System Unit	0	Unit 0
SU 32 +	1	Unit 1
	2	Unit 2
	3	Unit 3

User Defined Switches

User Defined Switches no. 1 0 - 254
UDS1 33 +

User Defined Switches no. 2 0 - 254
UDS2 34 +

User Defined Switches no. 3 0 - 254
UDS3 35 +

User Defined Switches no. 4 0 - 254
UDS4 36 +

Invisible Switches

(1) Boot Power Up		0	No
BPU	45 +	1	Yes
(2) Break		0	Not break
BREAK	46 -	1	Break
(3) Keyboard Action Mode		0	Enabled
KAM	47 +	1	Disabled
(4) Menu Entrance		0	Allowed
ME	48 +	1	Prohibited

Nur mit Break-taste zu betätigen

1,3,4 = Programmiert les.u.änderbar
DCREAD, DCLORIT

Anhang H

Tastaturbelegung

Internationale Tastatur

Symbol	ISO-7-Bit Code	Hex-Code	Bedeutung
XMT	000 0001	01	Datenaustausch mit Rechner 1)
STX	000 0010	02	Steuerzeichen zur Übertragung eines Blockes 2)
ETX	000 0011	03	Steuerzeichen am Ende eines Blockes 3)
	000 0100	04	Zeile löschen, Cursor auf Zeilenanfang
	000 0101	05	Ab Cursor Zeile löschen 1)
	000 0110	06	Ab Cursor Bildschirm löschen 1)
	000 0111	07	Positionieren auf nächsten Tab links 1)
	000 1000	08	Cursor eine Stelle nach links
	000 1001	09	Positionieren auf nächsten Tab rechts 4)
 (LF)	000 1010	0A	Cursor auf nächste Zeile gleiche Stelle
	000 1011	0B	Cursor eine Zeile tiefer
	000 1100	0C	Bildschirminhalt eine Zeile nach oben
 (CR)	000 1101	0D	Cursor auf erste Stelle der aktuellen Zeile (Carriage Return)
WORD 	000 1110	0E	Wort einfügen. Zeile nach rechts verschieben 1)

1) Wird standardmäßig vom BS1MP nicht ausgewertet 2) Bildschirm unsichtbar (Video off) 3) Video on 4) Im Kommandomodus des BS1MP jeweils 7 Stellen bei jedem Tab-Sprung

Symbol	ISO-7-Bit-Code	Hex-Code	Bedeutung
	000 1111	0F	Zeile einfügen - verschieben nach unten 1)
CHAR →	001 0000	10	Zeichen einfügen - verschieben nach rechts
← CHAR	001 0001	11	Zeichen löschen - verschieben nach links
← WORD	001 0010	12	Wort löschen - Zeile verschieben nach links 1)
	001 0011	13	Zeile löschen - verschieben nach oben 1)
	001 0100	14	Bildschirminhalt ausdrucken 1)
SEND	001 0101	15	Daten zum Rechner übertragen 1)
REQ ANF	001 0110	16	Daten vom Rechner anfordern 1)
	001 0111	17	Bildschirminhalt eine Zeile nach unten
→	001 1000	18	Cursor ein Zeichen nach rechts
	001 1001	19	Bildschirm löschen
CF AR	001 1010	1A	Sprung zur Anwenderroutine 1)
ESC	001 1011	1B	Umschaltung 1)
	001 1100	1C	Cursor eine Zeile höher
	001 1101	1D	Cursor auf Bildschirmanfang
RS ÷	001 1110	1E	Nächstes Zeichen ist ein Modus-Auswahlzeichen 1)
CE	001 1111	1F	Löschen letzten Wert 1)
	010 0000	20	Blank
!	010 0001	21	

1) Vom BS1MP standardmäßig nicht ausgewertet.

Symbol	ISO-7-Bit-Code	Hex-Code	Bedeutung
"	010 0010	22	
#	010 0011	23	
$	010 0100	24	
%	010 0101	25	
&	010 0110	26	
'	010 0111	27	
(	010 1000	28	
)	010 1001	29	
*	010 1010	2A	
+	010 1011	2B	
,	010 1100	2C	
-	010 1101	2D	
.	010 1110	2E	
/	010 1111	2F	
0	011 0000	30	
1	011 0001	31	
2	011 0010	32	
3	011 0011	33	
4	011 0100	34	
5	011 0101	35	
6	011 0110	36	
7	011 0111	37	
8	011 1000	38	
9	011 1001	39	
:	011 1010	3A	
;	011 1011	3B	

Symbol	ISO-7-Bit-Code	Hex-Code	Bedeutung
<	011 1100	3C	
=	011 1101	3D	
>	011 1110	3E	
?	011 1111	3F	
@	100 0000	40	
A	100 0001	41	
B	100 0010	42	
C	100 0011	43	
D	100 0100	44	
E	100 0101	45	
F	100 0110	46	
G	100 0111	47	
H	100 1000	48	
I	100 1001	49	
J	100 1010	4A	
K	100 1011	4B	
L	100 1100	4C	
M	100 1101	4D	
N	100 1110	4E	
O	100 1111	4F	
P	101 0000	50	
Q	101 0001	51	
R	101 0010	52	
S	101 0011	53	
T	101 0100	54	
U	101 0101	55	
V	101 0110	56	

Symbol	ISO-7-Bit-Code	Hex-Code	Bedeutung
W	101 0111	57	
X	101 1000	58	
Y	101 1001	59	
Z	101 1010	5A	
[	101 1011	5B	
\	101 1100	5C	
]	101 1101	5D	
^	101 1110	5E	
_	101 1111	5F	
`	110 0000	60	
a	110 0001	61	
b	110 0010	62	
c	110 0011	63	
d	110 0100	64	
e	110 0101	65	
f	110 0110	66	
g	110 0111	67	
h	110 1000	68	
i	110 1001	69	
j	110 1010	6A	
k	110 1011	6B	
l	110 1100	6C	
m	110 1101	6D	
n	110 1110	6E	
o	110 1111	6F	
p	111 0000	70	

Symbol	ISO-7-Bit-Code	Hex-Code	Bedeutung
q	111 0001	71	
r	111 0010	72	
s	111 0011	73	
t	111 0100	74	
u	111 0101	75	
v	111 0110	76	
w	111 0111	77	
x	111 1000	78	
y	111 1001	79	
y	111 1010	7A	
{	111 1011	7B	
|	111 1100	7C	
}	111 1101	7D	
—	111 1110	7E	
DEL	111 1111	7F	Zeichen löschen (im Kommandomodus wird die Kommandozeile gelöscht)

Codes größer 7FH:

Symbol	Hex-Code	Bedeutung Softwarekennung
F1	A0	PUSH-KEY (programmierbare Funktionstaste)
F2	A1	" 02
F3	A2	" 03
F4	A3	" 04
F5	A4	" .
F6	A5	" '
F7	A6	" '
F8	A7	" '
F9	A8	" '
F10	A9	" '
F11	AA	" '
F12	AB	" .
F13	AC	"
F14	AD	"
F15	AE	" 0F
F16	AF	" 10

Symbol	Hex-Code	Bedeutung
SHIFT+MODE	F8	Einsprung Menü
LINE	FA	
CLEAR	FB	

Abweichungen der deutschen Tastatur von der internationalen Tastatur:

Symbol	ISO-7-Bit-Code	Hex-Code	Bedeutung
DÜ	000 0001	01	wie XMT
§	100 0000	40	statt @
Ä	101 1011	5B	statt [
Ö	101 1100	5C	statt \
Ü	101 1101	5D	statt]
ä	111 1011	7B	statt {
ö	111 1100	7C	statt |
ü	111 1101	7D	statt }
ß	111 1110	7E	statt ~

Beim Aussprung aus einer Systemroutine ist das Carrybit gesetzt, falls sich im Ablauf ein Fehler ergeben hat. Der Fehlercode befindet sich im A-Register.

Falsche Operationen können verschiedene Fehlermeldungen zur Folge haben. Eine Liste der häufigsten oder wahrscheinlichsten Ursachen ist jeder Fehlermeldung angefügt.

Hexadezimaler
Wert

00 END OF FILE

- Dateiende erreicht, d. h. keine weiteren Daten vorhanden.
- Die Datei ist leer (enthält überhaupt keine Daten).
- EOE erreicht beim Schreiben auf eine "IBM"-Datei.

01 SECTOR MISSING

- Datenträger defekt.

02 CRC ERROR

- CRC Fehler im Sektorkopf, Datenträger defekt.
- CRC Fehler im Datenblock. Ein erneutes Schreiben auf diesem Sektor könnte helfen.

03 TIMING ERROR READ/WRITE XFER

- DMA Fehler, verursacht durch einen Controller- oder CPU-Fehler.

04 NO ADDRESS MARK

- Versuch, auf einen anderen Datenträgertyp als den angegebenen zuzugreifen (:Dn: statt :Fn:).
- Datenträger defekt.
- Schlechte Justierung des Laufwerks-Kopfes.

05 CONTROLLER OPERATION TIMEOUT

- Datenträger defekt.
- Der Monitor wartet auf Unterbrechung.
- Versuch, auf einen anderen als den angegebenen Datenträgertyp zuzugreifen.

06 TIMING ERROR

- Sektor defekt.
- Versuch, auf einen anderen als den angegebenen Datenträgertyp zuzugreifen (:Fn: statt :Dn:).

07 BUSY-BIT TIMEOUT

- Datenträger defekt.
- Versuch, auf einen anderen als den angegebenen Datenträgertyp zuzugreifen.
- Der Monitor wartet bis der Controller fertig ist.

08 DRIVE NOT READY

- Die Laufwerks-Klappe ist nicht geschlossen.
- Datenträger falsch eingelegt.
- Das Laufwerk ist nicht angeschlossen.
- Fehlende Stromversorgung zum Laufwerk.

09 NAME NOT IN DIRECTORY AND DIRECTORY FULL

- Programm im Inhaltsverzeichnis nicht auffindbar.
- Kein Platz mehr im Inhaltsverzeichnis für eine neue Datei. Wahrscheinlich belegt eine Anzahl gelöschter Einträge Platz im Inhaltsverzeichnis. Mit DIRPAC kann dieser Platz wieder freigegeben werden.

0A NAME NOT IN DIRECTORY

- Datei im Inhaltsverzeichnis nicht auffindbar.

0B PROGRAM NAME NOT FOUND

- Programm im Inhaltsverzeichnis nicht auffindbar.

0C ILLEGAL DIRECTORY ENTRY

- Falsche Identifizierung des Datenträgers.
- Falsche Parameter im Eintrag des Inhaltverzeichnisses.

0D DIRECTORY BLOCK ERROR DURING LOAD

- Unerwartetes Dateiende in einer ladbaren Datei.
- Falsches Ladeformat.

0E DRIVE NOT CONNECTED

- Laufwerk physikalisch nicht angeschlossen.
- Laufwerk falsch angeschlossen.
- Fehlende Stromversorgung zum Laufwerk.

0F VOLUME MAP MISSING

- Die Datei ISIS.MAP fehlt auf einer Intel-Diskette.

10 NON-VALID CONTROLLER COMMAND

- Controller erhielt falsches Kommando, was normalerweise im System nicht vorkommt.

11 IMPROPER UNIT SPECIFICATION

- Die Laufwerksbezeichnung ist nicht von der Form :Xn:, wobei :Xn: alphanumerische Zeichen sind.
- Laufwerksbezeichnung existiert nicht oder ist nicht implementiert.
- Falsche Laufwerksnummer (nicht 0-3).

12 CONVERSION TABLE MISSING

- Keine ASCII- oder EBCDIC-Konvertierungstabelle vorhanden.

13 DRIVE NUMBER OUT OF RANGE

- Gerätenummer nicht 0,1,2,3.

14 FILE PROTECTED

- Die umzubenennende oder zu löschende Datei ist schreibgeschützt (W- oder S-Attribut).

15 DE-EDIT NONHEX DIGIT (0-9,A-F)

- Das Zeichen, das in einen 4-Bit hexadezimalen Wert verwandelt werden soll, ist eine ungültige hexadezimale Zahl.

16 INPUT FROM NON-INPUT FILE

- Im FCB ist das Bit für Eingabe nicht gesetzt. Es handelt sich um einen Ausgabe FCB.
- Versuch, auf eine Direktzugriffsdatei zuzugreifen (X-Attribut).

17 READ AFTER END-OF-FILE

- Bei der vorhergehenden Leseoperation wurde Dateiendestatus gemeldet (Carry gesetzt und A-Reg. = 0 bedeutet EOF).
- Versuch, aus einer leeren Datei zu lesen.

18 OUTPUT TO NON-OUTPUT FILE

- Schreibversuch in eine Datei, deren FCB nicht vom Typ 2,3,5 oder 6 ist.
- IM FCB ist das Bit für Ausgabe nicht gesetzt. FCB für Eingabedatei.

19 UNASSIGNED DEVICE

- Logische Einheit (SI, SO, etc.) nicht zu der erwarteten Datei zugewiesen.

1A UNKNOWN DEVICE TYPE

- FCB-Typ nicht 2 bis 6.

1B 1ST CHARACTER OF COMMAND OR FILE NAME NOT A LETTER/DIGIT

- Das erste Zeichen im Kommando oder Dateinamen ist kein Buchstabe oder keine Zahl.

1C COMMAND OR FILE NAME LENGHT OUT OF RANGE

- Kommando lautet nach dem Dateibezeichner nicht richtig.
- Keine oder zuviele Zeichen in der Kommandozeile.

1D IMPROPER COMMAND UNIT

- Vom angegebenen Laufwerk kann kein Programm geladen werden.

1E SYSTEM NOT LOADED

- Versuch, eine Systemroutine aufzurufen, obwohl kein BS1MP geladen ist.

1F COMMAND NOT TERMINATED PROPERLY

- Kommandoabschluß ist nicht '/', 'CR', '$' oder Leerzeichen.

20 DELETED DATA RECORD

- Versuch, einen gelöschten Sektor von einer "IBM"-Diskette zu lesen.
- Datenträger und Laufwerksbezeichnung stimmen nicht überein.

21 LOGICAL UNIT NAME UNKNOWN

- Das BS1MP kennt diese logische Einheit (SI, SO, etc.) nicht.

22 WRITE ON NON-BLOCK FILE

- Schreibversuch auf nicht blockorientierte Datei.

23 WRITE/READ MISSING IN CTRL-BLK

- Keine Schreib- oder Leseroutine im Kontrollblock angegeben.

24 READ ON FILE TYPE OTHER THAN 4 OR 5

- Leseversuch aus Ausgabedatei/von Ausgabegerät.

25 LINE OVERFLOW

- Pufferüberlauf.

26 IMPROPER PARAMETER OR PARAMETER STRING

- Falscher Parameter bei einer BS1MP-Routine.
- Falscher Parameter oder Parameterstring in der Kommandozeile oder bei Antwortanforderung.
- Falsches Trenn- oder Endezeichen in der Kommandozeile.
- Falsche Kombination von Parametern im Parameterstring.

27 FILE ALREADY OPEN

- Versuch, eine bereits offene Datei zu eröffnen.

28 NO OPEN-ROUTINE IN CTRL-BLK

- Keine OPEN-Routine im Kontrollblock angegeben.

29 CLOSE ON CLOSED FILE

- Versuch, eine schon geschlossene oder nicht eröffnete Datei zu schließen.

2A NO CLOSE-ROUTINE IN CTRL-BLK

- Keine CLOSE-Routine im Kontrollblock angegeben.

2B BUFFER CHAIN DESTROYED

- Ein Puffer in der BS1MP-Pufferkette ist nicht mit 00,80 oder FF gekennzeichnet. Interner BS1MP-Fehler, oder Puffer vom Anwenderprogramm überschrieben. Die Obergrenze des verfügbaren Speichers steht unter BUFLIM (272EH).

2C HARDWARE ERROR

- Controller-Fehler.

2D IMPROPER FILE DESIGNATOR

- Der angegebene Dateibezeichner (:D1: name , :LP:, etc.) ist nicht dem System nicht bekannt.
- Der angegebene Dateibezeichner hat ein falsches Format.

2E IMPROPER ASSIGNMENT FORMAT

- Zuweisung im BS1MP unbekannt.
- Falsche Begrenzung der Zuweisung. Gültig ist ',', '$' oder CR.
- Kombination von Zuweisungen beim Anwenderprogramm nicht erlaubt.
- Kombination von Zuweisungen und Parametern im Anwenderprogramm nicht erlaubt.

2F NO ROOM ON VOLUME

- Keine freien Blöcke mehr im Belegungsverzeichnis. Mit REMAP kann auf unbenutzte Sektoren geprüft werden.

30 ILLEGAL BUFFER TYPE

- Ein Puffer in der BS1MP-Pufferkette ist nicht mit 00 oder 80H gekennzeichnet. Der Puffer wurde wahrscheinlich vom Anwenderprogramm überschrieben. Die Obergrenze des verfügbaren Speichers steht unter BUFLIM (272EH).

31 NON-EXISTENT INPUT FILE

- Keine Eingabedatei mit diesem Namen im Inhaltsverzeichnis. Klein- und Großschreibung im Dateinamen überprüfen.

32 OUTPUT FILE ALREADY EXISTS

- Versuch, eine Ausgabedatei zu eröffnen, deren Name schon existiert. Der Fehler kann bei OPEN und CLOSE auftreten.
- Versuch, eine Datei in einen schon existierenden Namen umzubenennen.

33 NO ROOM ON VOLUME

- Keine freien Sektoren mehr vorhanden. Ende der Diskette (EOV) erreicht.

34 IBM FILE NAME 8 CHARACTERS

- Ein "IBM"-Dateiname (:Jn:) darf nur aus ein bis acht alphanumerischen Zeichen bestehen.

35 IMPROPER TRACK NUMBER

- Spurnummer zu groß oder zu klein (INTEL oder BS1M-Diskette 0-76, "IBM"-Diskette 0-74).

36 IMPROPER SECTOR NUMBER

- Sektornummer kleiner 1 oder größer 26.

37 VOLUME REJECTED

- Initialisierungsversuch nach einigen Wiederholungen abgebrochen. Anderen Datenträger versuchen.

38 ROUTINE NOT AVAILABLE

- Zugriff auf nicht vorhandene Monitorroutine.
- Zugriff auf nicht geladene oder nicht vorhandene BS1MP-Routine (z. B. RFM).

39 READ/WRITE ERROR

- Lese- oder Schreiboperation abgebrochen. Wahrscheinlich defekter Datenträger.

3A ATTEMPT TO OVER-LOAD SYSTEM

- Versuch, das BS1MP zu überladen. Die System-Startadresse steht in BUFLIM (272EH).

3B READ-AFTER-WRITE ERROR

- Lesetest nach Schreiboperation abgebrochen. Wahrscheinlich defekter Datenträger.

3C CHECKSUM ERROR

- CHECKSUM-Fehler im Monitor.

3D ASYNC I/O ERROR

- Ein-/Ausgabefehler beim Senden oder Empfangen von Zeichen über das asynchrone Interface. Geschwindigkeit, Parity, Anzahl Stopp-Bits usw. sollten überprüft werden.

3E ILLEGAL TO START NEXT OPERATION

- Versuch, eine Controller-Operation zu starten, während die vorige Operation noch läuft oder nicht mit einem CALL DSWT abgeschlossen ist. Alle Operationen, die nicht automatisch mit einem CALL DSWT abgeschlossen sind, müssen beendet sein, bevor eine neue Controller-Operation gestartet wird.

3F DOOR OPENED

- Das Laufwerk wurde seit der letzten Lese- oder Schreiboperation geöffnet.

40 WRITE PROTECTED VOLUME

- Die Diskette ist schreibgeschützt. Zum Schreiben muß das Schreibschutzloch zugeklebt sein.

41 PROTECTED FUNCTION

- Versuch, eine für BS1MP reservierte Funktion auszuführen.

42 IMPROPER FUNCTION CODE

- Funktions-Code nicht zugelassen.

54 LOGICAL UNIT ALREADY ASSIGNED

- Die logische Einheit ist bereits zugewiesen (FCB vorhanden). Die Datei muß vor einer anderen Zuweisung geschlossen werden.

55 ATTEMPT TO RE-OPEN NON INPUT FILE

- Versuch, eine Ausgabedatei für Eingabe wieder zu eröffnen.

56 WARNING: FILE NOT DELETED

- Interne BS1MP Meldung, die anzeigt, daß eine Datei nach Anfrage nicht gelöscht wurde.

57 UNRECOGNIZED SYSTEM ERROR CODE RETURNED

- Der Geräte-Handler kennt diese Fehlermeldung des Controllers nicht.

58 NEW NAME ALREADY EXISTS

- Versuch, eine Datei in einen bereits existierenden Namen umzubenennen.

59 PHYSICAL FUNCTION TABLE ENTRY INDEX OUT OF RANGE

- Zugriff auf einen zusätzlichen Geräte-Handler, der nicht in der Sprungtabelle eingetragen ist.

5A IMPROPER SECTOR LENGTH

- Controller meldet falsche Blockgröße. Wahrscheinlich paßt der Datenträger nicht zum Geräte-Handler.

5B IMPROPER VOLUME FORMAT

- Datenträger paßt nicht zum aktuellen Geräte-Handler.
- Controller meldet falsche Dichte der Diskette.

5C END-OF-BLOCK

- Meldung des BS1MP, wenn BLOCKLINE + BINFILE gesetzt sind und auf einer "IBM"-Datei Blockende erreicht ist.

5D Unzulässiger Direktzugriff.

5E Ausgabedatei nicht vorhanden.

60 Weiterer Aufruf an den RFM nicht möglich, da ein Auftrag bearbeitet wird.

62 Dateinummer wird nicht benutzt (RFM).

63 Keine Direktzugriffsdatei (RFM).

64 Datei wurde offen gelassen (RFM).

65 Ungültiger Dateiname (RFM).

66 Überlauffehler (RFM).

67 Satz schon vorhanden (RFM).

68 Satz nicht gefunden (RFM).

69 Satz gelöscht (RFM).

6A Ungültiger Satzschlüssel/Nummer (RFM).

6D Unzulässiger Operationscode (RFM).

6E Dateiende (RFM).

6F Zu wenig Speicher (RFM).

Stichwortverzeichnis

D

E

S

T

An
Siemens AG
E 363
Nägelsbachstr. 14
8520 Erlangen

Vorschläge

Korrekturen

für Druckschrift:

Siemens Systeme 6·000
AMBOSS 1
Betriebssystem BS 1MP
und Dienstprogramme

Bedienungsanleitung

Bestell-Nr. E-36/7100-01
Ausgabe August 1982

Absender:

Name _____

Firma/Dienststelle _____

Anschrift _____

Telefon /

Sollten Sie beim Lesen dieser Unterlage auf Druckfehler gestoßen sein, bitten wir Sie, uns diese mit diesem Vordruck mitzuteilen. Ebenso dankbar sind wir für Anregungen und Verbesserungsvorschläge.

Vorschläge und/oder Korrekturen:

Siemens AG
Geschäftsgebiet Mittlere Datentechnik
Postfach 3240, D-8520 Erlangen

SIEMENS AKTIENGESELLSCHAFT

Bestell-Nr. E-36/7100-01
Printed in the Federal
Republic of Germany
8810 AG 8820.4 (15000) TD 2012