SIEMENS

SME-Disketten-betriebssystem ISIS II

Bedienungsanleitung


DOPPEL

SIEMENS

SME-Diskettenbetriebssystem ISIS II

Bedienungsanleitung

SIEMENS AKTIENGESELLSCHAFT

Herausgegeben von Siemens AG, Bereich Bauelemente, Produkt-Information, Balanstraße 73, D-8000 München 80

Fragen über Technik, Preise und Liefermöglichkeiten richten Sie bitte an unsere Zweigniederlassungen im Inland, Abteilung VB oder an unsere Landesgesellschaften im Ausland (siehe Geschäftsstellenverzeichnis).

Inhaltsverzeichnis

Diese Bedienungsanleitung behandelt das Diskettenbetriebssystem ISIS II des SME's (Siemens-Mikrocomputer-Entwicklungssystem). Sie beschreibt die Bedienungsmaßnahmen für den Betrieb des SME's unter der Steuerung von ISIS II, die Eingabe von ISIS-II-Kommandos über die Konsole sowie die Anwendung von ISIS-II-Systemaufrufen in PL/M 80 und 8080-Assemblerprogrammen.

Für die Inbetriebnahme des Systems, die Eingabe von Kommandos und die Dateibearbeitung mit Hilfe des Texteditors sind keine Programmierkenntnisse erforderlich. Die Beherrschung von PL/M 80 bzw. der 8080-Assemblersprache wird jedoch vorausgesetzt. In folgenden Broschüren findet sich eine Beschreibung von PL/M 80 bzw. der 8080-Assemblersprache:
PL/M-Programmiersprache, System SAB 8080
Assembler-Programmiersprache, System SAB 8080

Diese Bedienungsanleitung verlangt vom Leser Grundkenntnisse über das SME, wie in den folgenden Büchern erläutert:
Bedienungsanleitung, Siemens-Mikrocomputer-Entwicklungssystem
Hardware-Beschreibung, Siemens-Mikrocomputer-Entwicklungssystem
Diese Beschreibung enthält keinerlei Anweisungen, wie die Floppy-Disk-Steuerung in das SME-Gehäuse eingebaut und wie der Floppy-Disk-Speicher an das SME-System angeschlossen wird. Diese Anweisungen sind im MDS-DOS Hardware Reference Manual enthalten.

Die drei Programme ASM80, ETA80 und UPM, die auf der Systemdiskette mitgeliefert werden, sind in dieser Bedienungsanleitung nicht behandelt. Diese drei Programme werden in folgenden Büchern beschrieben:

Bedienungsanleitung, ISIS II 8080/8085 Makro-Assembler
Bedienungsanleitung, SME Emulations- und Testadapter
Bedienungsanleitung, Universelles PROM-Programmierprogramm

1. Einführung in das Diskettenbetriebssystem

Floppy-Disk-Speicher und ISIS-II-Betriebssystem machen das SME (Siemens-Mikrocomputer-Entwicklungssystem) zu einem wirkungsvollen und komfortablen Mikrocomputer-Entwicklungssystem. Dabei bildet ISIS II die softwaremäßige Schnittstelle zum Floppy-Disk-Speicher und zu allen anderen Standard-Peripheriegeräten. Die Kommunikation mit ISIS II erfolgt durch Eingabe von Kommandos über die Konsole oder mit ISIS-II-Systemaufrufen in PL/M- oder 8080-Assembler-Programmen.

Die ISIS-II-Funktionen zur Dateiverwaltung gestatten, Daten mit hoher Geschwindigkeit auf Disketten zu speichern und wiederzugewinnen. Eingaben über ein mechanisches Gerät, wie einen Lochstreifenleser, können in eine Diskettendatei kopiert werden und dann unter Angabe des ihr zugewiesenen symbolischen Namens von der Diskette wiedergewonnen werden. Mit dem Texteditor können Diskettendateien erstellt und bestehende geändert werden. Primärprogramme in 8080-Assemblersprache oder PL/M, die in Form von Dateien auf Diskette gespeichert sind, können mit dem 8080-Assembler bzw. dem PL/M-80-Übersetzer verarbeitet werden. Dabei entsteht relativierbarer Code, der wiederum in Dateien, sogenannten Objektdateien, auf Disketten abgespeichert wird. Zusätzlich wird ein Übersetzungsprotokoll erzeugt, das auf Zeilendrucker oder auf eine Diskettendatei ausgegeben wird. Nach Eingabe des entsprechenden Kommandos über die Konsole läuft eine Übersetzung ohne weitere Eingriffe des Anwenders ab.

Ablauffähige Programme werden einfach durch Eingabe des Dateinamens geladen und gestartet.

1.1. Hardware-Ausstattung

Das Diskettenbetriebssystem ISIS II erfordert folgende Hardware-Ausstattung:

SME-Zentraleinheit mit Monitor und Arbeitsspeicherausbau von 32 bis 64 KB
Konsole (Bedienungsblattschreiber oder Datensichtgerät)
wahlweise weitere vom Monitor bediente Peripheriegeräte (z.B. Matrixdrucker)
Floppy-Disk-Speicher mit zwei Laufwerken
je eine Baugruppe für die Schnittstellenanpassung und zur Kanal- und Gerätesteuerung, die in die SME-Zentraleinheit einzubauen sind
1 Systemdiskette mit dem ISIS-II-Betriebssystem
1 Verbindungskabel

1.2. Software-Komponenten

Den Kern des ISIS-II-Diskettenbetriebssystems bildet das Organisationsprogramm (Ablaufteil), das die Software-Schnittstelle zum Floppy-Disk-Speicher realisiert und dabei wahlfreien Zugriff zu den gespeicherten Daten gestattet. Für die Bedienung der Standardperipheriegeräte des SME's greift das Organisationsprogramm auf die entsprechenden Ein-/Ausgaberoutinen des Monitors zu.

Für den Betrieb eines SME's unter der Steuerung von ISIS II ist ein minimaler Arbeitsspeicherausbau (RAM) von 32 KB erforderlich.

Zusätzlich zu dem Organisationsprogramm werden mit dem Diskettenbetriebssystem ISIS II folgende Programme ausgeführt:

Bibliotheksverwaltungsprogramm (LIB)
Bindeprogramm (LINK)
Entrelativierungsprogramm (LOCATE)
Texteditor (EDIT)
Assembler (ASM80)
Betriebsprogramm für den Emulations- und Testadapter (ICE80)
Universelles PROM-Programmierungsprogramm (UPM)

1.2.1. Speicherbedarf

ISIS II erfordert, wie bereits erwähnt, ein SME mit mindestens 32 KB Arbeitsspeicher (RAM). Die ersten 12 KB sind für den residenten Teil von ISIS II reserviert. Der Speicherplatz oberhalb 12 K wird abwechselnd von den nichtresidenten ISIS-II-Routinen und von Anwenderprogrammen belegt. Eine detaillierte Beschreibung der Speicherorganisation findet sich in Kapitel 11 (ISIS-II-Programmierung).

1.2.2. Monitor

Arbeitet das SME mit einem Floppy-Disk-Speicher unter der Steuerung des ISIS-II-Diskettenbetriebssystems, ist der SME-Monitor dem ISIS-II-Organisationsprogramm untergeordnet. Voraussetzung dafür ist – neben dem korrekten Anschluß der Floppy-Disk-Hardware – daß bei der Inbetriebnahme des SME's, sich eine ISIS-II-Systemdiskette in Laufwerk Ø des Floppy-Disk-Speichers befindet. Dann wird über das Urladeprogramm (eine PROM-Routine) das ISIS-II-Organisationsprogramm initialisiert und nicht der Monitor.

Dennoch stehen auch unter ISIS II die Monitor-Testfunktionen zur Verfügung. Dazu wird ein Programm mit Hilfe des ISIS-II-Kommandos DEBUG geladen und anschließend mit den entsprechenden Monitor-Kommandos (z.B. Speicher- und/oder Registerinhalte anzeigen und/oder ändern) ausgestattet.

1.2.3. Ein-/Ausgabeschnittstelle

Das ISIS-II-Ein-/Ausgabesystem verarbeitet grundsätzlich Dateien, wobei nicht nur der Speicherplatz auf Disketten, sondern auch die SME-Standardperipheriegeräte als Dateien verwaltet und über symbolische Namen angesprochen werden. Dazu steht für Anwenderprogramme eine umfassende logische Schnittstelle in Form der Systemaufrufe zur Verfügung, über die stets mehrere Bytes vom Speicher zur Datei bzw. von der Datei zum Speicher übertragen werden. Im Gegensatz dazu erfolgt der Datentransfer über die Ein-/Ausgaberoutinen des Monitors nur byteweise.

Die Realisierung dieser Ein-/Ausgabeschnittstelle sowie die Verwaltung der Dateien auf Diskette ist als eine der wesentlichen Aufgaben von ISIS II anzusehen. Eine detaillierte Beschreibung der Funktionen des Ein-/Ausgabesystems findet sich in Kapitel 3 (Dateiverwaltung).

1.2.4. Relativieren und Binden

Eines der Hauptmerkmale von ISIS II ist das Relativieren und Binden. Die ISIS-II-Übersetzer ASM80 und PL/M 80 erzeugen relativierbare (verschiebbare) Objektmoduln. Diese Moduln können durch das LINK-Programm zu kompletten Programmen gebunden werden. Absolute Adressen werden nach dem Binden durch das LOCATE-Programm zugewiesen. Die Objekt-

moduln können durch das LIB-Programm auch in eine Bibliotheksdatei übertragen werden, damit sie zum Einbinden in andere Programme zur Verfügung stehen.

1.2.5. ISIS-II-Texteditor

Der Texteditor wird auf der Systemdiskette geliefert und ist in Kapitel 5 beschrieben. Der Texteditor erzeugt dynamisch Dateien auf einer Diskette oder einem Standardausgabegerät, abhängig von den beim Aufruf angegebenen Parametern.

1.2.6. ISIS-II-8080-Sprachübersetzer

Der 8080-Makroassembler wird ebenfalls auf der Systemdiskette geliefert. Unter der Steuerung von ISIS II wird der Assembler durch Eingeben eines Kommandos geladen, wobei über Parameter die Primärprogrammdatei sowie wahlweise die Objekt- und die Listdatei festgelegt werden.

Der PL/M-80-Compiler ist ein gesondertes Produkt und wird nicht auf der Systemdiskette mitgeliefert; er ist separat zu beziehen. Der PL/M-80-Compiler erzeugt ebenfalls Objektmoduln mit relativierbarem Code.

1.2.7. ISIS-II-Anwendungsbeispiel

Zur Demonstration der Anwendung einiger ISIS-II-Funktionen sei angenommen, daß ein in Assembler geschriebenes Primärprogramm auf Lochstreifen vorliegt. Dieses Programm ist für den Ablauf auf dem SME vorgesehen und ruft dabei Ein-/Ausgaberoutinen des Monitors auf. Dieses Programm soll nun auf ISIS II mit gleichzeitiger Ausnutzung der wirkungsvolleren Ein-/Ausgabemöglichkeiten umgestellt werden.

Der erste Schritt dieser Umstellung besteht im Einlesen des Lochstreifens über einen schnellen Lochstreifenleser und Kopieren des Programms auf die Diskettendatei PROGA.SRC. Dazu dient folgendes ISIS-II-Kommando:

-COPY :HR: TO PROGA.SRC

Anschließend wird der Texteditor aufgerufen:

-EDIT PROGA.SRC

Jetzt können die Programmänderungen für den Aufruf von ISIS-II-Ein-/Ausgaberoutinen durchgeführt werden. Anschließend wird der Assembler aufgerufen:

-ASM80 PROGA.SRC OBJECT (PROGA)

Das übersetzte Programm wird in der Diskettendatei PROGA abgelegt, und ein Übersetzungsprotokoll auf dem Drucker ausgegeben. Nun kann das Programm mit Hilfe des Monitors getestet werden:

-DEBUG PROGA

3100

.G

Falls Fehler aufgetreten sind, ist das Primärprogramm mittels Texteditor zu korrigieren, erneut zu übersetzen und zu testen.

Bei korrektem Ablauf kann das Programm nach Abschluß des Testens in eine Bibliotheksdatei übertragen werden. Von dort erfolgt das Laden und Starten des Programms unter ISIS II durch Eingabe des Dateinamens:

-PROGA

2. Bedienungsmaßnahmen

In diesem Kapitel werden der Floppy-Disk-Speicher und die notwendigen Bedienungsmaßnahmen für den Betrieb des SME's mit dem Diskettenbetriebssystem ISIS II beschrieben.

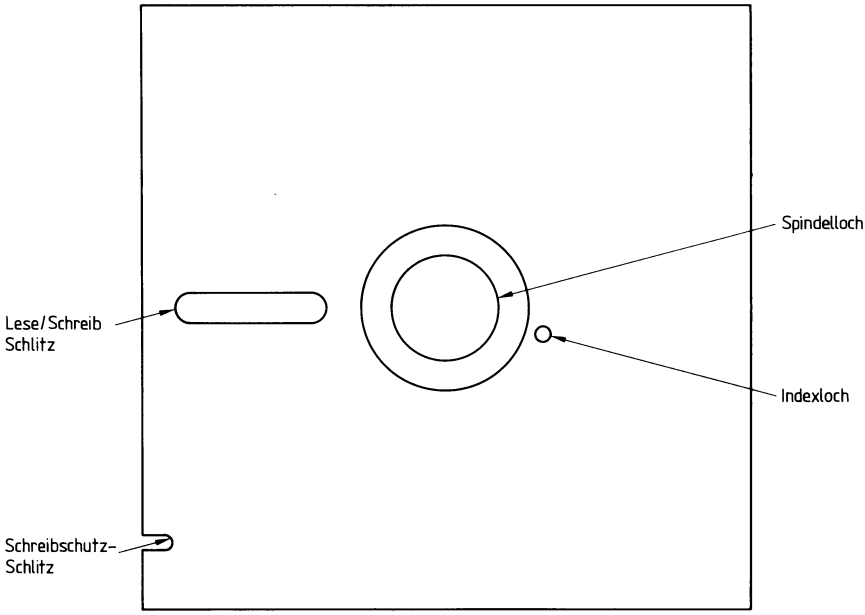
2.1. Beschreibung des Floppy-Disk-Speichers

Wird der Netzschalter am Bedienungsfeld des Floppy-Disk-Speichers gedrückt, so daß er auf EIN steht, ist dieser für den Betrieb bereit. Jedes Laufwerk besitzt eine Türe mit Verriegelung und eine Anzeige, die immer dann aufleuchtet, wenn dieses Laufwerk vom System angesprochen wird.

2.1.1. Beschreibung der Diskette

Die Diskette ist in einer quadratischen Hülle mit einer Kantenlänge von ca. 20 cm untergebracht. Das Material der Hülle ist so beschaffen, daß es elektronische Aufladungen reduziert. Wird die Diskette nicht benutzt, sollte sie in der Transporthülle gelagert werden (Lagertemperatur 10 bis 50 °C).

Im folgenden Bild ist eine Diskette dargestellt.



Die Diskette wird mit dem Schreib-/Leseschlitz voraus und dem leicht von der Mitte versetzten Indexloch nach unten in das Laufwerk eingeführt. Es ist darauf zu achten, daß der Floppy-Disk-Speicher eingeschaltet ist. Die Diskette wird erst dann in Rotation versetzt, wenn die Türe des Laufwerks geschlossen ist.

Während eine Diskette sich im Laufwerk befindet, dürfen weder das Zentralgerät SME 800 noch der Floppy-Disk-Speicher ein- oder ausgeschaltet werden.

Disketten sollten nur unter der ISIS-II-Kommandostufe, also wenn ein Bindestrich (-) als Aufforderungszeichen ausgegeben wurde, den Laufwerken entnommen werden. Dateien können außer mit dem Schreibschutzschlitz auch durch programmiert zugewiesene Attribute gegen unerwünschtes Beschreiben geschützt werden. Um spätere Lesefehler und damit Datenverlust zu vermeiden, ist sorgfältig darauf zu achten, daß die in den Öffnungen erscheinende magnetisch beschichtete Aufzeichnungsoberfläche nicht berührt oder beschädigt wird. Das Diskettenformat ist in Kapitel 3 (Dateiverwaltung) beschrieben.

2.1.2. Systemdiskette

Eine der mitgelieferten Disketten enthält das ISIS-II-Betriebssystem und wird deshalb als Systemdiskette bezeichnet. ISIS II kann nur dann in das SME geladen werden, wenn die Systemdiskette im Laufwerk Ø eingesetzt ist. Es ist auch möglich, eine Systemdiskette in das Laufwerk 1 einzusetzen, sie wird dann aber wie eine Datendiskette behandelt. Neue Systemdisketten können mit dem Kommando FORMAT erzeugt werden (Kapitel 4).

2.2. Bedienungsmaßnahmen zum Laden von ISIS II

Im folgenden wird vorausgesetzt, daß ein Floppy-Disk-Speicher und eine Konsole (Bedienungsblattschreiber oder Datensichtgerät) an das SME angeschlossen sind.

2.2.1. Urladen des Systems

SME, Konsole, Floppy-Disk-Speicher und alle anderen Peripheriegeräte (soweit vorhanden) einschalten.

Systemdiskette in das Laufwerk Ø schieben und Türe schließen.
Falls erforderlich, Diskette in das Laufwerk 1 einschieben und Türe schließen.
Urladeschalter am Bedienungsfeld des SME nach oben drücken.
Rücksetzschalter am Bedienungsfeld des SME nach oben drücken (Unterbrechungsanzeige 2 muß aufleuchten).
Leerzeichen von der Konsole aus eingeben (Unterbrechungsanzeige 2 muß verlöschen).
Urladeschalter zurücksetzen.

Ist der Urladevorgang erfolgreich verlaufen, wird ISIS II von der Systemdiskette im Laufwerk Ø geladen. Auf der Konsole wird die ISIS-II-Meldung sowie ein Bindestrich als Eingabeaufforderung ausgegeben.

ISIS II, Vx.y

Mit x wird die Versionsnummer und mit y die Freigabenummer dargestellt.
Jetzt kann man die in Kapitel 4 erklärten ISIS-II-Kommandos eingeben.
Der Urladevorgang kann, falls erforderlich, wiederholt werden, darf aber immer nur auf der ISIS-II-Kommandostufe durchgeführt werden, d. h., wenn das ISIS-II- oder das Monitor-Aufforderungszeichen als letztes Zeichen auf dem Bediengerät ausgegeben ist. Die Daten auf der

Diskette können zerstört werden, wenn der Urladevorgang eingeleitet wird, während die Diskette gerade beschrieben oder gelesen wird. Bevor das SME oder der Floppy-Disk-Speicher ausgeschaltet wird, sind unbedingt die Disketten zu entnehmen, weil sonst unkontrollierbare Fehler auftreten können.

Wenn im Laufwerk Ø keine Diskette eingeschoben ist oder der Floppy-Disk-Speicher nicht eingeschaltet oder nicht an das SME angeschlossen ist, initialisiert der Urladevorgang den Monitor anstelle von ISIS II. Falls im Laufwerk Ø eine Nichtsystemdiskette eingeschoben ist, wird eine dementsprechende Meldung am Konsolgerät ausgegeben und die Steuerung ebenfalls dem Monitor übergeben.

2.2.2. Konsolgerät

Bei System-Initialisierung (Urladen) legt das System ein logisches Konsolgerät fest, von dem ab diesem Zeitpunkt alle Kommandos erwartet werden. Diese sogenannte »ursprüngliche – oder Primärzuweisung« kann durch ein A-Kommando (Monitor), durch einen CONSOL-Systemaufruf oder durch das SUBMIT-Kommando geändert werden (d.h. dem logischen Konsolgerät wird ein neues physikalisches Gerät zugewiesen).
(Siehe Kapitel 4 und 12)

Steuerzeichen CTRL-E

In einer SUBMIT-Datei wird das spezielle Steuerzeichen CTRL-E verwendet, um die Konsolgerätefunktion zwischen der SUBMIT-Datei und dem ursprünglichen Konsolgerät (vorübergehend) auszutauschen. Durch Eingabe eines Steuerzeichens CTRL-E am ursprünglichen Konsolgerät kann man die Konsolgerätefunktion wieder an die SUBMIT-Datei zurückgeben.

Das Steuerzeichen CTRL-E wird vom Texteditor als Datenzeichen interpretiert, im Textpuffer abgespeichert und bei Beendigung des Programms EDIT (Kommando E) in die SUBMIT-Datei gebracht. Wenn das Steuerzeichen CTRL-E für ISIS II als Teil der Kommandozeile eingegeben wird, also entweder von einer SUBMIT-Datei oder von der Tastatur des Konsolgeräts stammt, dann wird das Steuerzeichen CTRL-E zur Kontrolle wiederholt, aber nicht im Eingabepuffer abgespeichert.

Steuerzeichen CTRL-S und CTRL-Q

Die Ausgabe auf der Konsole kann durch Eingabe des Steuerzeichens CTRL-S angehalten werden. Dies ist besonders bei Datensichtgeräten nützlich, um die Ausgabe auf dem Bildschirm zur Überprüfung anhalten zu können. Mit Eingabe des Steuerzeichens CTRL-Q kann die Ausgabe wieder fortgesetzt werden.

2.2.3. Testschalter

Der Testschalter (debug toggle) ist ein interner Software-Systemschalter, der anzeigt, ob beim Auftreten eines Fehlers die Steuerung an ISIS II (Stellung = Ø) oder an den Monitor (Stellung = 1) zurückkehren soll. Das DEBUG-Kommando und der LOAD-Systemaufruf mit RETSW = 2 setzen den Testschalter auf 1. Wenn der Testschalter auf 1 steht, läuft ISIS II im Testmodus. Die folgenden Maßnahmen setzen den Testschalter auf Ø (bewirken, daß ISIS II den Testmodus verläßt):

Drücken des Unterbrechungsschalters 1 während eines Programmlaufs
Ausführen eines EXIT-Systemaufrufs
Ausführen eines LOAD-Systemaufrufs mit RETSW = 1
Ausführen eines Monitor-Kommandos G8.

2.2.4. Unterbrechungen

Die Unterbrechungsschalter Ø, 1 und 2 am Bedienfeld des SME sind für Systemzwecke reserviert. Die anderen Unterbrechungen sind für den Anwender vorgesehen (siehe »SME-Bedienungsanleitung« und »SME-Hardware-Beschreibung«). Mit dem Unterbrechungsschalter 1 kann zu jeder Zeit (nicht aber aus dem MONITOR!) in das ISIS-II-Betriebssystem zurückgekehrt werden. Wenn dieser Schalter gedrückt wird, geschieht folgendes:
alle eröffneten Dateien werden in ihrem gegenwärtigen Zustand geschlossen.
wenn notwendig, wird die ursprüngliche Konsole wieder zugewiesen.
der Testschalter wird auf Ø rückgesetzt.
ISIS II wird wieder von der Diskette geladen und erwartet eine neue Eingabe.
Durch Drücken des Unterbrechungsschalters Ø kann eine Rückkehr zum SME-Monitor erzwungen werden. Das Programm kann durch Eingabe des Monitor-Kommandos G an der Unterbrechungsstelle fortgesetzt werden. Wenn z. B. gewünscht wird, daß die Ausgabe von Daten auf dem Datensichtgerät unterbrochen werden soll, ist der Unterbrechungsschalter Ø zu drücken. Durch Eingabe des G-Kommandos wird das Programm an der Unterbrechungsstelle fortgesetzt.

Wenn der Programmablauf beendet ist, kann durch Eingabe des Monitor-Kommandos G8 die Steuerung wieder an ISIS II übergeben werden, wobei alle eröffneten Dateien geschlossen werden, der Testschalter rückgesetzt wird, ISIS II geladen und gestartet wird und auf eine Eingabe wartet.

Das Drücken des Unterbrechungsschalters 1 ist im Prinzip gleichwertig, ist aber hier unwirksam, weil der Monitor über Maskierung alle Unterbrechungen außer Ø sperrt.

Zusammenfassung:
1. Übergabe der Steuerung von ISIS II an den SME-Monitor:
Unterbrechungsschalter Ø drücken oder
DEBUG-Kommando eingeben oder
Systemaufruf LOAD mit RETSW = 2 ausführen.

2. Übergabe der Steuerung vom SME-Monitor an ISIS II:
Monitorkommando G8 eingeben oder
Systemaufruf EXIT ausführen oder
Systemaufruf LOAD mit RETSW = 1 ausführen.

2.2.5. Fehlerbehandlung

Fehler, die von ISIS II während des Programmablaufes erkannt werden, sind in behebbare und nicht behebbare eingeteilt. Die Fehler sind durchnummeriert und in Kapitel 15 aufgelistet und beschrieben. Wurde ein behebbarer Fehler von ISIS II erkannt, wird die Fehlernummer dem Anwenderprogramm übergeben, das dann die Fehlerbehandlung durchführen kann.

Ein nichtbehebbarer Fehler führt zu folgender Meldung:
ERROR nnn, USER PC mmmm

Die Ausgabe der Fehlermeldung erfolgt über die ursprüngliche Konsole, wobei nnn die Fehlernummer und mmmm den Befehlszählerstand beim Auftreten eines Fehlers identifiziert. Anschließend wird die Steuerung ISIS II bzw. dem SME-Monitor übergeben, abhängig von der Stellung des Testschalters.

Hat der erkannte Fehler die Nummer 24 (Disketten-Ein-/Ausgabe-Fehler), dann wird über die Konsole folgende zusätzliche Meldung ausgegeben:

FDCC = ØØnn

Dabei kennzeichnet »nn« den Typ des aufgetretenen Ein-/Ausgabefehlers (siehe Kapitel 15).

Falls Fehler 24 aufgetreten ist, dürfte es zweckmäßig sein, die Diskette, die den Fehler verursacht hat, neu zu formatieren. Das FORMAT-Kommando unter Verwendung des A-Schalters (siehe Kapitel 4) formatiert die Diskette neu und kopiert alle Dateien bis zum Erreichen der Datei, die den Fehler 24 verursacht hat, auf eine andere Diskette. Die restlichen Dateien kann man dann mit je einem COPY-Kommando kopieren. Man kann auch die fehlerhafte Datei löschen und dann mit einem FORMAT-Kommando eine Kopie der Diskette herstellen.

Falls der Testschalter auf Ø steht, wenn ein nichtbehebbarer Fehler auftritt, geschieht folgendes:

Alle eröffneten Dateien werden in ihrem gegenwärtigen Zustand geschlossen, einschließlich

:CI: und :CO:
Die ursprüngliche Konsole wird wieder als :CI: und :CO: eröffnet.
ISIS II wird von der Diskette geladen und gestartet und erwartet eine Eingabe.

Falls der Testschalter auf 1 steht, wenn ein nicht behebbarer Fehler auftritt, geschieht folgendes:

Alle eröffneten Dateien bleiben eröffnet.
Die Steuerung geht an den Monitor über.
Der Monitor gibt das Aufforderungszeichen Punkt (.) für ein Monitor-Kommando aus.

Jetzt kann mit den Monitor-Kommandos die Fehlersuche durch Abfragen der Register- und Speicherinhalte begonnen werden, das Programm sollte aber nicht nochmals mit dem G-Kommando gestartet werden. Mit dem Kommando G8 kann man wieder in ISIS II zurückkehren.

Auf keinen Fall darf zu diesem Zeitpunkt ISIS II durch einen Urladevorgang gestartet werden, weil dann die eröffneten Dateien nicht geschlossen würden.

Hinweis

Es ist zwar unmöglich, Anwenderprogramme in den ISIS-II-residenten Bereich zu laden, doch ist dieser Bereich nicht gegen Überschreiben durch das Anwenderprogramm geschützt.

Sollte ein Anwenderprogramm Teile von ISIS II überschreiben, so können dadurch auch nachfolgende Systemaufrufe beeinträchtigt sein. In der Folge könnten durch fehlerhafte Ein-/Ausgabeoperationen Bereiche auf der Diskette zerstört werden.

Diese Gefahr ist vor allem bei ungetesteten Programmen gegeben.

3. Dateiverwaltung

Eine Datei ist die Summe von logisch zusammengehörenden Daten, die mit einem Namen versehen eine Einheit bilden und vom System als Einheit von und zu den Peripheriegeräten übertragen werden können. Die Daten in der Datei sind nach bestimmten Kriterien organisiert, wobei der Anfang und das Ende der Datei eindeutig identifiziert ist.

3.1. Dateinamen

Jede Datei ist dem Betriebssystem unter ihrem Dateinamen bekannt, bzw. muß dem System bekannt gemacht werden.

In ISIS-II-Dateinamen sind Groß- und Kleinbuchstaben sowie Ziffern erlaubt.

Eine Diskettendatei hat einen dreiteiligen Dateinamen mit folgender Syntax (die geschweiften Klammern bedeuten, daß der Eintrag wahlweise ist):

{:gerät:}name{.erw}

:gerät: bedeutet :FØ:, :F1:, :F2: oder :F3: und zeigt an, in welches Laufwerk die Diskette eingeschoben ist. Fehlt diese Angabe, so wird :FØ: angenommen. Mit :gerät: können aber auch Peripheriegeräte angesprochen werden.

name besteht aus 1 bis 6 Buchstaben oder Ziffern und muß immer angegeben werden.

.erw besteht aus 1 bis 3 Buchstaben oder Ziffern. Fehlt die Angabe, wird der Dateiname einfach durch name identifiziert.

Beispiel: COUNT. SRC (Primärdatei)
COUNT. OBJ (Maschinencodedatei)
COUNT. LST (Protokolldatei)

Peripheriegeräten (mit Ausnahme der Flopy-Disk-Speicher) werden ein oder mehrere Dateinamen zugeordnet, die nur aus dem Element :gerät: bestehen.
name und .erw sind für diese Geräte irrelevant und werden deshalb, wenn sie eingegeben werden, ignoriert.

Im folgenden sind die Zuordnungen für :gerät: zu der SME-Standardperipherie aufgeführt:

Die Dateinamen der Standardgeräte lauten:
:TI: Blattschreiber-Tastatur als Eingabedatei
:TO: Blattschreiber-Schreibwerk als Ausgabedatei
:TP: Blattschreiber-Lochstreifenstanzer als Ausgabedatei
:TR: Blattschreiber-Lochstreifenleser als Eingabedatei
:VI: Datensichtgerät-Tastatur als Eingabedatei
:VO: Datensichtgerät-Bildschirm als Ausgabedatei
:HP: Schneller Lochstreifenstanzer als Ausgabedatei
:HR: Schneller Lochstreifenleser als Eingabedatei
:LP: Matrixdrucker als Ausgabedatei

Es gibt drei besondere ISIS-II-Dateien, die keinem physikalischen Gerät zugeordnet sind, sondern abhängig von der Zuordnung (mit dem Monitor-Kommando A oder mit dem Monitor-Systemaufruf IOSET oder mit dem ISIS-II-Systemaufruf CONSOLE) verschiedene »Geräte« repräsentieren.

Es sind dies:

:BB: Leerdaten-Datei als Ein- oder Ausgabedatei
:CI: Konsolgerät-Eingabedatei
:CO: Konsolgerät-Ausgabedatei

Falls man nichtstandardmäßige Geräte mit Hilfe der Monitorroutine IODEF definiert (und dazu eigene Gerätesteuerprogramme liefert, siehe SME-Bedienungsanleitung), so muß man diese in ISIS-II-Kommandos und -Aufrufen mit folgenden Dateinamen aufrufen:

:R1: Lochstreifenleser 1
:R2: Lochstreifenleser 2
:P1: Lochstreifenstanzer 1
:P2: Lochstreifenstanzer 2
:L1: Drucker
:I1: Eingabe-Konsole
:O1: Ausgabe-Konsole

3.2. Eröffnen und Schließen von Dateien

Bevor ISIS II auf eine Datei zugreifen kann, muß diese eröffnet werden. Beim OPEN werden Informationen über die zu eröffnende Datei in Systemtabellen abgelegt und, falls notwendig, E/A-Puffer zugewiesen. Wird eine Datei geschlossen, werden die Einträge in den Systemtabellen, die diese Datei betreffen, gelöscht. Für den Fall, daß eine Diskettendatei geschlossen wird, wird die für diese Datei notwendige Information in den Etikettenbereich*) der Diskette geschrieben.

Wird mit einem ISIS-II-Kommando über die Konsole auf eine Datei zugegriffen, sorgt ISIS II für OPEN und CLOSE. Greift ein Programm mit einem Systemaufruf auf eine Anwenderdatei zu, muß das Programm selbst für das Eröffnen und Schließen sorgen.

3.3. Konsolgerät-Ein-/Ausgabedateien

Unter ISIS II sind die Dateien :CI: und :CO: in Wirklichkeit nur Pseudonyme für die Geräte, die als Konsolgerät-Eingabe und -Ausgabe dienen (Systemdateien). Die Datei :CI: ist immer die Herkunft der System-Kommandos. Die Datei :CO: erhält die Ausgabe auf Bediengerät, z. B. die Kontrollausgabe eines Kommandos. Diese zwei Dateien sind immer offen, es ist daher zwar nicht sinnvoll, aber kein Programmfehler, diese Dateien mit einem Systemaufruf OPEN eröffnen zu wollen. Weder :CI: noch :CO: zählen zu den 6 Dateien von ISIS II, die simultan eröffnet sein dürfen.

Beim Urladen des Systems (Monitor oder ISIS II) wird die Konsoleingabe- und die Konsolausgabedatei den Geräten :VI: oder :TI: und :TO: zugewiesen, abhängig davon, auf welchem Gerät während des Urladevorgangs die Leertaste gedrückt wurde (Primärzuweisung).

Durch Programme, die einen CONSOL-Systemaufruf beinhalten, können die Systemdateien :CI: und :CO: anderen Dateien (Geräten) zugewiesen werden.

Wenn das Konsoleingabegerät :CI: das Dateiende erkennt, werden :CI: und :CO: geschlossen. ISIS II wird urgeladen und die ursprüngliche Gerätezuweisung der Konsole wiederhergestellt.

*) Etikettenbereich beinhaltet Inhaltsverzeichnis und Dateibeschreibung.

3.4. Leerdatendatei

Die Leerdatendatei :BB: (bucket byte) entspricht keinem physikalischen Gerät. Sie kann als Pseudo-Ausgabedatei verwendet werden. Wenn z. B. das im Entwurf befindliche Anwenderprogramm vorsieht, daß der Anwender ein Kennwort eingibt, kann man :BB: dem Bedienungs-Ausgabegerät zuweisen, damit das Kennwort nicht auf dem Bediengerät als Echo ausgegeben wird (siehe Kapitel 11 unter Zeilenaufbereitung). Die Leerdatendatei kann auch als Eingabe verwendet werden, falls eine Datei mit der Länge Null benötigt wird (Günstig beim Test von Anwenderprogrammen, die E/A-Dateien verarbeiten, wenn die Dateien im Teststadium noch nicht existieren!!).

Die Leerdatendatei :BB: muß wie jede andere Datei eröffnet werden.

Es können mehrere Leerdatendateien eröffnet sein. Jede von Ihnen zählt aber dann als eine der maximal 6 Dateien, die von ISIS II simultan geöffnet sein dürfen (!).

3.5. Umsetzen von Dateien

Mit Hilfe von ISIS-II-Kommandos können Dateien in andere Dateien übertragen oder von einem Gerät auf ein anderes Gerät umgesetzt werden. Wenn beispielsweise eine Datei, die in Form eines Lochstreifens vorliegt, in eine Diskettendatei übertragen werden soll, ist der Lochstreifen in den Lochstreifenleser einzulegen und das ISIS-II-Kommando COPY wie folgt über die Konsole einzugeben:

```
COPY :HR: TP PROG.SRC
```

Wenn der Inhalt einer Diskettendatei über das Konsolausgabegerät :CO: ausgegeben werden soll, ist dies durch folgendes COPY-Kommando möglich:

```
COPY FILE.LST TO :CO:
```

Das Gleiche wird erreicht, wenn ein Anwenderprogramm, wie in Kapitel 13 oder 14 als Musterbeispiel gezeigt, geschrieben wird.

Nach der Übersetzung und der Konvertierung in den Binärcode würde das Programm über die Konsole den Inhalt der angegebenen Datei ausgeben:

```
TYPE FILE.LST
```

In Kapitel 5 können Sie über das Umsetzen von Dateien durch den Texteditor nachlesen.

3.6. Dateitypen

Obwohl alle Daten in Dateien aus Folgen binärer Ziffern (Nullen und Einsen) bestehen, werden diese Bits entsprechend dem Dateityp auf verschiedene Weise interpretiert. Der Texteditor erzeugt Dateien mit ASCII-codiertem Text (siehe Vorwort mit Hinweis auf ASCII). Die Daten in Primärprogrammen werden von ASM80 oder PL/M80 als ASCII-codierter Text interpretiert. Wenn ASM80 und PL/M80 die Primärprogramme übersetzen, erzeugen sie Objektprogramme, die binäre Maschinenbefehle enthalten und zusätzlich Informationen, die von den Dienstprogrammen LINK, LOCATE und LOAD verwendet werden.

Die Art der in einer Datei enthaltenen Daten bestimmt, was mit dieser Datei geschehen kann. Eine Datei mit ASCII-Zeichen kann kauf dem Matrixdrucker ausgegeben werden. Wird jedoch versucht, z. B. eine Datei mit Binärcode auf dem Matrixdrucker auszugeben, würde unverständlicher Text gedruckt.

Objektprogramme können erst ausgeführt werden, wenn ihnen absolute Adressen durch LOCATE zugewiesen worden sind. Selbst dann ist es möglich, daß sie nicht ordnungsgemäß ablaufen, falls sie sich auf Externadressen beziehen, die nicht im Objektprogramm vorhanden sind (siehe Kapitel 6).

Es ist sinnvoll, die Datenart der Datei durch eine geeignete Erweiterung des Datennamens zu kennzeichnen. Primärprogramme könnten z. B. die Erweiterung .SRC haben (oder .ASY oder .PLM, um die verwendete Sprache anzuzeigen). Der von den Übersetzern erzeugte Objektcode könnte den Zusatz .OBJ haben; der Übersetzer verwendet, wenn man keinen Namen für die Ausgabedatei vorgibt, den Namen der Eingabedatei mit dem Zusatz .OBJ. Außer den Übersetzern weisen auch viele ISIS-II-Programme den durch sie angesprochenen Dateien eine spezielle Erweiterung zu, falls man sie nicht selbst vorgibt. Das Programm SUBMIT nimmt z. B. selbständig so eine Erweiterung bei der Eingabedatei an. Dies erspart Zeit bei der Eingabe der Kommandos, kann aber zu Mißverständnissen führen, wenn man sich nicht darüber klar ist, was vom Programm gemacht wird. Andere ISIS-II-Programme erzeugen temporäre Arbeitsdateien mit dem Zusatz .TMP. Einige Beispiele dafür sind

EDIT.TMP, LOCATE.TMP, LINK.TMP und LIB.TMP.

3.7. Diskettenformat

Es gibt Floppy-Disk-Speicher für normale und für doppelte Schreibdichte. Die nachfolgenden Angaben gelten für normale Schreibdichte. Eine Diskette verfügt über 77 Spuren. Jede Spur enthält 26 Blöcke (oder Sektoren) mit je 128 Bytes (insgesamt 2002 Blöcke). In ISIS II können Disketten entweder Systemdisketten oder Nichtsystemdisketten sein. Eine Systemdiskette enthält die ISIS-II-Systemsoftware, die zum Betrieb des Systems notwendig ist. Die von der Systemsoftware und anderen Dateien belegte Blockanzahl kann durch Verwendung des DIR-Kommandos ermittelt werden (siehe Kapitel 4). Nichtsystemdisketten enthalten nur den zur Verwaltung des Dateninhaltsverzeichnisses erforderlichen Teil der ISIS-II-Systemsoftware und können daher nicht ohne eine im Laufwerk Ø eingelegte Systemdiskette verwendet werden. Dafür steht aber nahezu die gesamte Speicherkapazität der Diskette für Anwenderprogramme zur Verfügung. Disketten müssen vor ihrer Verwendung im System formatiert werden (siehe FORMAT-Kommando in Kapitel 4).

ISIS II behandelt eine Datei als Folge von Bytes; es gibt also nicht so etwas wie einen »physikalischen Satz«. Jedoch wird der Speicherplatz einer Datei in kompletten Blöcken zugewiesen, selbst wenn dabei der letzte Block in der Datei nur teilweise gefüllt wird. Es wird also nie ein Block gemeinsam von zwei Dateien verwendet. Wenn eine Datei gelöscht wird, werden die von ihr belegten Blöcke für eine Neuzuweisung durch ISIS II freigegeben.

3.7.1. Disketteninhaltsverzeichnis

Jede Diskette besitzt ein Inhaltsverzeichnis mit Platz für 200 Einträge. Dies bedeutet, daß eine Diskette 200 Dateien enthalten könnte, falls dadurch nicht die Speicher-Kapazität der Diskette überschritten wird.

Ein Inhaltsverzeichniseintrag enthält Datei-Informationen, die vom DIR-Kommando verwendet werden, um folgende vier Einträge auszugeben:

Dateiname,
Anzahl der Blöcke, die der Datei zugewiesen sind,
Anzahl der Bytes in der Datei (Länge),
Attribute.

Außerdem gibt das DIR-Kommando auch an, wieviele der insgesamt 2002 verfügbaren Blöcke belegt sind.

Länge

Die Anzahl der Datenbytes in einer Datei ist ihre Länge. Diese Anzahl schließt nicht die Zeigerblöcke ein. Die Länge vergrößert sich, wenn eine Datei gerade geschrieben wird, und kann auch durch den SEEK-Systemaufruf mit MODE = 3 und durch den OPEN-Systemaufruf mit ACCESS = 2 beeinflußt werden (siehe Kapitel 12, Systemaufrufe).

Attribute

Es gibt vier Attribute, die zu jeder Diskettendatei gehören. Diese können durch das ATTRIB-Kommando (siehe Kapitel 4) oder durch den ATTRIB-Systemaufruf (siehe Kapitel 12) gesetzt oder rückgesetzt (ein- oder ausgeschaltet) werden.

Die Attribute sind:

Unsichtbar	(INVISIBLE)
Schreibschutz	(WRITE-PROTECT)
Format	(FORMAT)
System	(SYSTEM)

Wenn eine Datei erzeugt wird, werden alle Attribute rückgesetzt. Dies bedeutet, daß die Datei keines der oben genannten Attribute besitzt. Dies gilt auch für eine mit dem COPY-Kommando erzeugte Datei.

Unsichtbar — Dateien, bei denen das Attribut »unsichtbar« gesetzt ist, werden nicht vom DIR-Kommando aufgelistet, wenn nicht der I-Schalter verwendet wird. Alle Systemdateien besitzen das Attribut »unsichtbar«. Es ist ratsam, Dateien mit dem Attribut »unsichtbar« auch »schreibgeschützt« zu machen.

Schreibschutz — Dateien, bei denen das Attribut »Schreibschutz« gesetzt ist, können nicht als Ausgabedateien eröffnet werden (siehe OPEN-Systemaufruf, Kapitel 12), sie können auch nicht gelöscht oder umbenannt werden. Schreibgeschützte Dateien können durch das FORMAT-Kommando überschrieben werden.

Format — Was für schreibgeschützte Dateien gilt, gilt auch für Dateien mit dem »Format«-Attribut. Beim Formatieren von Disketten mit dem FORMAT-Kommando ohne Schalter A oder S werden nur die Dateien mit dem Format-Attribut auf der neuen Diskette erzeugt. Darum sollte das bei den vier ISIS-II-Dateien ISIS.DIR, ISIS.MAP, ISIS. TO und ISIS.LAB gesetzte Format-Attribut niemals rückgesetzt werden. Keine Diskette kann ohne diese vier Dateien ordnungsgemäß arbeiten. Das Format-Attribut sollte auch keiner anderen Datei zugewiesen werden.

System — Wenn der Schalter S beim FORMAT-Kommando angegeben ist, werden nur die Dateien, bei denen das Systemattribut gesetzt ist, auf die Diskette im Laufwerk 1 kopiert.

3.8. Programmsteuerung der Dateien

Programme können durch Verwendung der ISIS-II-Systemaufrufe sowohl Informationen von einer Eingabedatei erhalten als auch Informationen an eine Ausgabedatei übertragen. Die Systemaufrufe ermöglichen es, Dateien dynamisch zu erzeugen, Datenblöcke variabler Länge zu übertragen, Zeilen speziell aufzubereiten und bei Eingabe über Tastatur die Eingabe zur Kontrolle wiederholt zu bekommen. Das ist in Kapitel 11 und 12 erklärt.

4. ISIS-II-Kommandos

Die ISIS-II-Kommandos sind Bestandteil des ISIS-II-Betriebssystems und sind nach dem Laden des Betriebssystems dem Benutzer zugänglich. Die gewünschten auszuführenden Funktionen sind über die Konsole dem System mitzuteilen. Vor Ausführung der Kommandos werden diese auf formale Richtigkeit geprüft.

Gerätezuordnung

Nach dem Laden des Betriebssystems muß die Konsole immer ein Bedienungsblattschreiber oder ein Datensichtgerät sein. Die symbolische Gerätebezeichnung für die Konsole lautet:

für Eingabe :CI: (Console Input)
für Ausgabe :CO: (Console Output)

Später können per Programm mit dem Systemaufruf CONSOLE andere Geräte für CI oder CO zugewiesen werden. ISIS II nimmt ferner Kommandos von einer Diskettendatei an.

Die Kommandos sind aufgeteilt nach

Dateiverwaltung

DIR	Ausgeben des Disketteninhaltsverzeichnisses einer Diskette
COPY	Kopieren einer Datei
DELETE	Löschen einer Diskettendatei
RENAME	Umbenennen einer Diskettendatei
ATTRIB	Ändern von Attributen einer Diskettendatei

Initialisierung einer Diskette

FORMAT	Einrichten einer neuen Diskette

Umwandlung von Programmen

BINOBJ	Umwandeln eines Programms vom absoluten Binärcode des ISIS I zum absoluten Objektcode des ISIS II.
HEXOBJ	Umwandeln eines Programms vom hexadezimalen Format des ISIS I zum absoluten Objektcode des ISIS II
OBJHEX	Umwandeln vom absoluten Objektcode des ISIS II zum hexadezimal-Format des ISIS I.

Programmausführung

DEBUG	Ausführen eines Programms im Testmodus
Dateiname	Ein ausführbares Programm in einer Diskettendatei kann ausgeführt werden, indem man den Dateinamen als Kommando eingibt. (ISIS-II-Kommandos sind, ausgenommen DEBUG, in Wirklichkeit Dateinamen von ausführbaren Programmen auf der Systemdiskette)
SUBMIT	Ausführen einer Kommandofolge im Nichtdialogbetrieb

4.1. Korrekturmöglichkeiten bei der Kommandoeingabe

Das Tastenfeld der Konsole sieht einige Möglichkeiten vor, eingegebene Kommandos zu korrigieren. Werden Korrekturen eingegeben, muß die CTRL-Taste gleichzeitig mit dem Steuerzeichen gedrückt werden.

4.1.1. Rubout

Löschen des zuletzt eingegebenen Zeichens. Das gelöschte Zeichen wird zur Kontrolle nochmals ausgegeben. Sollen mehrere Zeichen gelöscht werden, ist dies durch wiederholtes Drükken dieser Taste möglich. Sind alle Zeichen dieser Zeile gelöscht, wird dies durch ein Klingelzeichen mitgeteilt.

4.1.2. CTRL-R

Wird diese Taste gedrückt, wird das augenblicklich gültige Kommando ausgegeben. Dies ist vorteilhaft, wenn durch mehrere Korrekturen die Übersichtlichkeit des eingegebenen Textes beeinträchtigt ist.

4.1.3. CTRL-X

Die zuletzt eingebene Zeile wird gelöscht. ISIS antwortet mit dem Zeichen #, sowie mit Wagenrücklauf und Zeilenvorschub.

4.2. Blanko-Dateinamen

Einige Systemkommandos erlauben Dateinamen mit einem Blankoteil zu spezifizieren. Ein Stern (*) oder ein Fragezeichen (?) ersetzt einige oder alle Zeichen eines Namens oder einer Erweiterung. Die Kommandos wirken sich dann auf alle Dateien aus, deren Namen unter Auslassung der ersetzten Zeichen zur Übereinstimmung gebracht werden können.

Zum Beispiel bedeutet:

name.*	daß nach der Datei »name« mit beliebigem Zusatz gesucht werden soll
*.zus	daß nach allen Dateinamen mit dem Zusatz »zus« gesucht werden soll
.	daß nach allen Dateien mit einem beliebigen Zusatz gesucht werden soll.

Der Stern kann auch als Blankoteil den Rest eines Namens oder Zusatzes festlegen. Zum Beispiel bedeutet:

AB*.HEX	daß nach allen Dateien gesucht wird, die AB als erste zwei Zeichen ihres Namens und HEX als Zusatz aufweisen. Folgende Dateinamen stimmen überein: ABC.HEX, ABXYZ.HEX, AB.HEX

Das Fragezeichen legt ein einzelnes Zeichen als Blankoteil fest. Zum Beispiel bedeutet

A?B.HEX	daß alle Dateinamen gesucht werden, mit A als erstes und B als drittes Zeichen eines aus drei Zeichen bestehenden Namens und Hex als Zusatz,
A??.*	daß alle Dateinamen mit A als erstem Zeichen eines aus drei Zeichen bestehenden Namens und mit einem beliebigen Zusatz gesucht werden.

4.3. Kommandosyntax

ISIS-II-Kommandos haben im allgemeinen folgendes Format:

‹Kommando›{parameterliste}

Darin ist ›Kommando‹ der Name eines in diesem Kapitel beschriebenen ISIS-II-Systemkommandos. Die Parameter in der ›parameterliste‹ werden durch Leerzeichen getrennt. Einige der Parameter erfordern Zusatzinformationen, die dem Parameter folgen müssen und in runde Klammern einzuschließen sind. Das bei ISIS-I-Kommandos teilweise erforderliche Dollarzeichen ($) wird von ISIS II ignoriert und wie ein Leerzeichen behandelt.

Erklärung der verwendeten Symbole:

{ } geschweifte Klammern bedeuten wahlfreie Angabe

[ ] einer der in eckigen Klammern stehenden Einträge ist obligatorisch

( ) runde Klammern müssen eingegeben werden

XXX Großbuchstaben müssen genau in der gezeigten Weise eingegeben werden, da sie das Kennwort darstellen

xxx Kleinbuchstaben müssen entsprechend den Erläuterungen im Text durch Parameter ersetzt werden

:xx: bedeutet eines der möglichen Peripheriegeräte.

Das Ende des Kommandos wird durch Drücken der RETURN-Taste mitgeteilt.

4.3.1. ATTRIB-Kommando (Ändern der Attribute einer Diskettendatei)

Allgemeines

Das Kommando ATTRIB ändert die Attribute einer Diskettendatei. Wird in dem Kommando ATTRIB eine Datei angegeben, die auf der Diskette nicht vorhanden oder keine Diskettendatei ist, wird folgende Meldung über die Konsole ausgegeben:

dateiname, NO SUCH FILE

Hierbei ist ›dateiname‹ der im Kommando festgelegte Name. Anschließend wird die Steuerung ISIS II übergeben.

Werden für dasselbe Attribut zwei verschiedene Werte angegeben, besitzt das zuletzt eingegebene Gültigkeit.

Format

ATTRIB {:Fx:}datei [{IØ / I1} {WØ / W1} {FØ / F1} {SØ / S1}] {Q}

Es bedeuten

ATTRIB	Name des Kommandos
:Fx:	Diskette im Laufwerk 1–5. Fehlt der Parameter, wird die Datei auf der Diskette im Laufwerk Ø angesprochen
datei	Name der Diskettendatei, deren Attribute geändert werden sollen

IØ	Attribut ›Unsichtbar‹ rücksetzen. Diese Datei wird beim Aufruf des DIR-Kommandos aufgelistet.
I1	Attribut ›Unsichtbar‹ setzen. Diese Datei wird beim Aufruf des Kommandos DIR nicht aufgelistet, es sei denn, der Parameter I ist beim DIR-Kommando angegeben
WØ	Attribut ›Schreibschutz‹ rücksetzen. Diese Datei wird dann als ungeschützte Anwenderdatei behandelt.
W1	Attribut ›Schreibschutz‹ setzen. Diese Datei kann weder für Ausgabe oder Änderungen eröffnet, noch gelöscht oder umbenannt werden.
FØ	Attribut ›Format‹ rücksetzen. Es erfolgt keine automatische Kopierung beim Aufruf des Kommandos FORMAT (außer mit dem Parameter A)
F1	Attribut ›Format‹ setzen. Diese Datei kann durch das Kommando FORMAT auf die Diskette im Laufwerk 1 kopiert werden. Bei den 4 ISIS-Dateien ISIS.DIR, ISIS.MAP, ISIS.TO und ISIS.LAB ist das Attribut ›Format‹ gesetzt und darf nie zurückgesetzt werden, da keine Diskette im Betriebssystem ISIS II ohne diese 4 Dateien richtig arbeiten kann.
SØ	Attribut ›System‹ rücksetzen. Es erfolgt keine automatische Kopierung beim Aufruf des Kommandos FORMAT (außer mit dem Parameter A)
S1	Attribut ›System‹ setzen. Diese Datei wird beim Aufruf des Kommandos FORMAT auf die Diskette im Laufwerk x kopiert, vorausgesetzt, der Parameter S ist im FORMAT-Kommando angegeben.
Q	Operation im Abfragemodus durchführen. Bevor die Attribute einer Datei geändert werden, wird folgende Meldung ausgegeben: dateiname, MODIFY ATTRIBUTES? Wird Y oder y eingegeben, wird das Attribut geändert. Jedes andere eingegebene Zeichen bewirkt, daß die Attribute der festgelegten Datei ungeändert bleiben.

Beispiel 1
Dieses Beispiel ändert das Attribut ›Schreibschutz‹ einer Dateigruppe:

```
-ATTRIB PROGA.* W1
PROGA.SRC. ATTRIBUTES MODIFIED
PROGA.OBJ. ATTRIBUTES MODIFIED
-
```

Beispiel 2
Dieses Beispiel setzt das Attribut ›System‹ für das Programm TYPE (das Programmbeispiel in den Anhängen A und B) auf 1 (siehe Kommando FORMAT).

```
-ATTRIB TYPE S1
TYPE, ATTRIBUTES MODIFIED
-
```

4.3.2. BINOBJ-Kommando (Umwandeln von ISIS I Binärformat in einen ISIS II absoluten Objektmodul)

Allgemeines
Das Kommando BINOBJ wandelt das absolute Binärformat früherer ISIS-Versionen in das Objektmodulformat von ISIS II um.

Format
BINOBJ {:Fx:}bindatei TO {:Fx:}absdatei

Es bedeuten

BINOBJ	Name des Kommandos
:Fx:	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die Datei auf der Diskette im Laufwerk Ø angesprochen.
bindatei	Datei, die den Modul im absoluten Binärformat von ISIS I (ISIS-Versionen 1.Ø bis 1.2) enthält.
absdatei	Datei, die den absoluten Objektmodul enthalten soll, der unter ISIS II ablauffähig ist. Die Startadresse (die Adresse des ersten auszuführenden Befehls) wird der ›bindatei‹ entnommen. Der mit dem Objektmodul gespeicherte Modulname wird dem Namensteil des Dateinamens ›absdatei‹ entnommen. Der durch BINOBJ erzeugte Objektmodul enthält **keine Symboltabelle**.

Beispiel
Das folgende Beispiel wandelt ein ausführbares Programm vom Format der ISIS-Version 1.2 (und früher) in einen ausführbaren absoluten Objektmodul um:

```
-BINOBJ PROGA.BIN TO PROGA
-
```

4.3.3. COPY-Kommando (Kopieren einer Datei)

Allgemeines

Mit dem COPY-Kommando wird eine vorhandene Datei in eine neue Datei kopiert. Wird in dem Kommando COPY als neue Datei ein Dateiname angegeben, der schon auf der angesprochenen Diskette existiert, wird folgende Meldung über die Konsole ausgegeben:

neudatei, ALREADY EXISTS, DELETE?

Werden daraufhin ein Y oder y und ein Wagenrücklauf eingegeben, löscht das Kommando COPY die vorhandene Datei, bevor die Kopie erzeugt wird.

Wurde ein anderes Zeichen als Y oder y eingegeben, wird das Kommando nicht ausgeführt und die Steuerung ISIS II übergeben.

Format 1

Es werden nur Diskettendateien kopiert
COPY {:Fx:}altdatei1{, altdatei2, ...} TO {:Fx:}neudatei{U}

Es bedeuten

COPY	Name des Kommandos
:Fx: (Eingabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die ›altdatei‹ von der Diskette im Laufwerk Ø gelesen.
altdatei 1	Name der zu kopierenden Datei. Die Datei muß bei diesem Format eine Diskettendatei sein. Der Inhalt von ›altdatei‹ wird nicht verändert.
altdatei 2 . . . altdatei n	Name(n) der Datei(n), die in der Reihenfolge wie angegeben an die ›altdatei1‹ in der ›neudatei‹ gekettet werden soll(en)
:Fx: (Ausgabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die ›neudatei‹ auf die Diskette im Laufwerk Ø geschrieben.
neudatei	Name der zu erstellenden Datei, die den Inhalt von ›altdatei1‹ und, wenn angegeben, ›altdatei2‹, ... n, aufnimmt. Alle Attribute werden bei der ›neudatei‹ rückgesetzt.
U	Unabhängig davon, ob die ›neudatei‹ schon existiert oder nicht, wird die ›neudatei‹ zur Aktualisierung eröffnet. Die bestehende Datei wird dabei nicht gelöscht, sondern die zu kopierende Datei in die bestehende einkopiert, wobei die bestehende soweit überschrieben wird, wie die einzukopierende lang ist. Ist die einzukopierende Datei länger als die bestehende, wird ›neudatei‹ auf die Länge der einzukopierenden erweitert. Die Attribute der bestehenden Datei werden nicht verändert.

Format 2

Es wird eine Datei von einem Standardperipheriegerät in eine Diskettendatei kopiert.
COPY :xx: TO {:Fx:}neudatei {U}

Es bedeuten

COPY	Name des Kommandos
:xx: (Eingabe)	symbolischer Gerätename. Von diesem Gerät wird die zu kopierende Datei eingelesen
:Fx: (Ausgabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die ›neudatei‹ auf die Diskette im Laufwerk Ø geschrieben.
neudatei	siehe Format 1
U	siehe Format 1

Format 3

Eine Diskettendatei wird auf ein Standardperipheriegerät ausgegeben

COPY {:Fx:}altdatei1{,altdatei2, ...}TO :xx:

Es bedeuten

COPY	Name des Kommandos
:Fx: (Eingabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die ›altdatei‹ von der Diskette im Laufwerk Ø gelesen
altdatei 1	siehe Format 1
altdatei 2 . . . altdatei n	siehe Format 1
:xx: (Ausgabe)	symbolischer Gerätename Auf dieses Gerät wird die zu kopierende Datei ausgegeben.

Format 4

Eine Datei wird von einem Standardperipheriegerät auf ein anderes Standardperipheriegerät kopiert

COPY :xx: TO :xx:

Es bedeuten

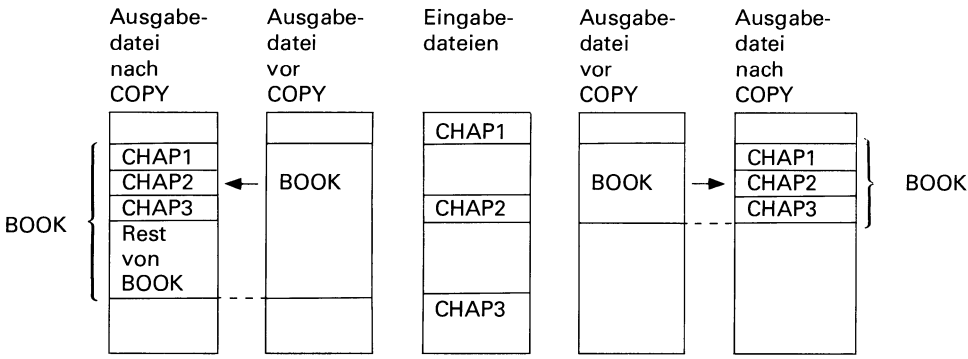
COPY	Name des Kommandos
:xx:	symbolischer Gerätename des Eingabe- bzw. des Ausgabegerätes.

Beispiel 1
In diesem Beispiel werden 3 Eingabedateien in eine schon bestehende Ausgabedatei kopiert, wobei deren Inhalt zerstört wird. Die Eingabedateien und die Ausgabedatei sind Diskettendateien, wobei die Diskette im Laufwerk Ø liegt.

```
-COPY CHAP1,CHAP2,CHAP3 TO BOOK
BOOK, FILE ALREADY EXISTS, DELETE?
Y
-
```

Beispiel 2
Wird im Beispiel 1 noch der U Parameter angegeben, könnte die Dateianordnung wie folgt aussehen

```
-COPY CHAP? TO BOOK U
```



Beispiel 3

In diesem Beispiel wird eine Datei auf der Diskette im Laufwerk 3 über den Matrixdrucker ausgegeben

```
-COPY :F3:BOOK TO :LP:
-
```

Beispiel 4

In diesem Beispiel wird eine Datei auf der Diskette im Laufwerk Ø auf die Diskette im Laufwerk 1 kopiert

```
-COPY CHAP.BIN TO :F1:NEWPR.BIN
```

Beispiel 5

In diesem Beispiel wird eine Lochstreifendatei über das Konsolausgabegerät ausgegeben.

```
-COPY :HR: TO :CO:
```

4.3.4. DEBUG-Kommando (Ausführen eines Programms im Testmodus)

Allgemeines

Das Kommando DEBUG lädt ein ablauffähiges Programm von der Diskette in den Speicher und übergibt die Steuerung dem SME-Monitor. Dieser gibt die Startadresse des Programms über die Konsole aus und wartet dann auf die Eingabe eines Monitor-Kommandos. Das Programm kann mit dem Monitor-Kommando G gestartet werden. Im G-Kommando können eine Startadresse (Einsprungadresse) und bis zu zwei Unterbrechungsadressen angegeben werden. Wenn die Programmausführung an einer Adresse unterbrochen wird, können die Speicher- und Registerinhalte mit Monitor-Kommandos geprüft und gegebenenfalls geändert werden. Danach kann das Programm mit einem weiteren G-Kommando fortgesetzt werden. Ist während der Programmausführung ein Fehler aufgetreten, kann der Versuch, das Pro-

gramm mit einem G-Kommando fortzusetzen, schwerwiegende Folgen haben. Es ist z.B. möglich, daß ISIS II überschrieben wird oder es können unerwünschte Schreib-/Leseoperationen auf der Diskette durchgeführt werden.

Die SME-Monitor-Kommandos sind in der SME-Bedienungsanleitung beschrieben. Zur Rückgabe der Steuerung an ISIS, d.h. Rücksetzen des Testschalters, gibt es 4 Möglichkeiten:

Monitor-Kommando G8 eingeben
in dem zu testenden Programm den Systemaufruf EXIT ausführen
in dem zu testenden Programm den Systemaufruf LOAD mit dem Parameter RETSW = 1 ausführen
Unterbrechungsschalter 1 während des Programmlaufs drücken

Format

DEBUG {:Fx:} progname {parameter}

Es bedeuten

DEBUG	Name des Kommandos
:Fx: (Eingabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die Datei von der Diskette im Laufwerk Ø geladen, wenn ein ›progname‹ angegeben ist.
progname	Name des zu ladenden ablauffähigen Programms. Das Programm muß ein absoluter Objektmodul sein. Wird kein ›progname‹ (einschließlich :Fx:) angegeben, wird die Steuerung dem SME-Monitor übergeben und kein Programm geladen
parameter	Parameter sind nur anzugeben, wenn sie vom Programm benötigt werden, denn das Programm muß diese selbst einlesen. (Siehe auch Abschnitt 4.3.7 »Ausführen eines Programms in einer Diskettendatei«.)

Beispiel 1
In diesem Beispiel wird gezeigt, wie das Musterprogramm in den Kapiteln 13 und 14 von der Diskette im Laufwerk :FØ: geladen und die Steuerung an den Monitor übergeben wird.

```
-DEBUG TYPE FILE.TXT
#3100 H
. G
```
Die Datei FILE.TXT wird auf der Konsole ausgegeben.
```
-
```

Beispiel 2
In diesem Beispiel wird dasselbe Programm unter der Steuerung des Monitors ausgeführt, doch wird die Steuerung mit dem Kommando G8, anstatt durch die Ausführung des Systemaufrufs EXIT an ISIS II zurückgegeben.

```
-DEBUG TYPE FILE.TXT
#3100 H
. G - 3120
```

Beim Erreichen des Unterbrechungspunktes 312Ø können Monitorkommandos verwendet werden, um Register und Speicher zu prüfen.

-G8
ISIS II, Vx.x

Beispiel 3
Mit diesem Beispiel wird die Steuerung dem Monitor übergeben und es wird kein Programm geladen

-DEBUG
#ØØØ8

-

4.3.5. DELETE-Kommando (Löschen einer Diskettendatei)

Allgemeines

Das Kommando DELETE löscht eine oder mehrere Dateien auf der Diskette und stellt den von ihr belegten Platz ISIS II für Neuzuweisung zur Verfügung.

Wird mit dem Kommando DELETE versucht, eine Datei zu löschen, deren Attribute ›Schreibschutz‹ oder ›Format‹ gesetzt sind, wird folgende Meldung über die Konsole ausgegeben:

datei, WRITE PROTECTED

und die Steuerung ISIS II übergeben.

Wird in dem Kommando DELETE eine Datei angegeben, die auf der Diskette nicht vorhanden ist, wird folgende Meldung über die Konsole ausgegeben:

dateiname, NO SUCH FILE

und die Steuerung ISIS II übergeben.

Format

DELETE {:Fx:}datei1,{{:Fx:}datei2, {:Fx:}datei3 ...}{Q}

Es bedeuten

DELETE	Name des Kommandos
:Fx:	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die Datei auf der Diskette im Laufwerk Ø gelöscht
datei 1	Name der zu löschenden Datei. Blanko-Dateinamen können verwendet werden, um Dateigruppen zu löschen. Die ordnungsgemäße Löschung wird über die Konsole bestätigt.
:Fx:	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird (werden) die Datei(en) auf der Diskette im Laufwerk Ø gelöscht.

datei 2 . . . datei n	Name(n) der zu löschenden Datei(en)
Q	Operation im Abfragemodus durchführen. Bevor eine Datei gelöscht wird, wird folgende Meldung ausgegeben: DELETE? Wird Y oder y eingegeben, wird die Datei gelöscht; jedes andere eingegebene Zeichen bewirkt, daß die Datei nicht gelöscht wird.

Beispiel 1
In diesem Beispiel werden 3 Dateien auf der Diskette im Laufwerk Ø gelöscht.

```
-DELETE CHAP1.TXT,CHAP2.TXT,CHAP3.TXT
CHAP1.TXT    DELETED
CHAP2.TXT    DELETED
CHAP3.TXT    DELETED
```

Dasselbe Ergebnis erzielt man durch Eingabe von:

```
-DELETE CHAP?.*
CHAP1.TXT    DELETED
CHAP2.TXT    DELETED
CHAP3.TXT    DELETED
```

Beispiel 2
In diesem Beispiel wird versucht, eine Datei, die mit dem Attribut »Schreibschutz« versehen ist, zu löschen:

```
-DELETE SYSTEM.DIR
SYSTEM.DIR, WRITE PROTECTED

-
```

Beispiel 3
In diesem Beispiel wird versucht, eine Datei auf der Diskette im Laufwerk 1 zu löschen, die dort nicht vorhanden ist:

```
-DELETE :F1:TYPE.SRC
:F1: TYPE.SRC, NO SUCH FILE
```

4.3.6. DIR-Kommando (Auflisten des Inhaltsverzeichnisses einer Diskette)

Allgemeines
Mit dem DIR-Kommando wird das Inhaltsverzeichnis einer Diskette über die Konsole ausgegeben.

Format

DIR {TO listdatei} {FOR datei} {:Fx:} {I} {F}

Es bedeuten

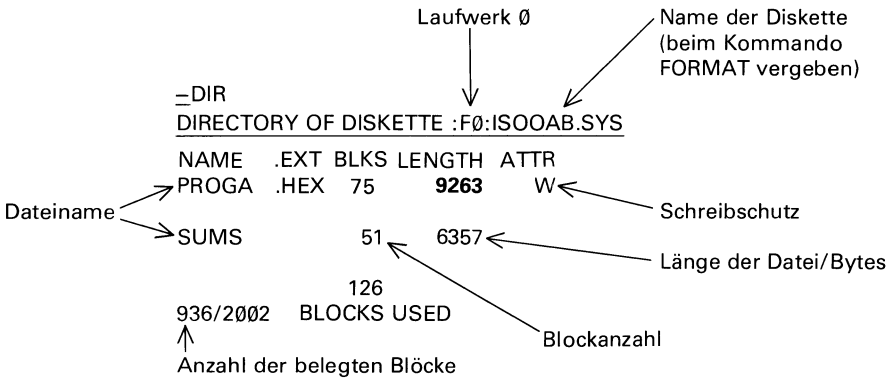
DIR	Name des Kommandos
TO listdatei	Datei, auf der das Inhaltsverzeichnis ausgegeben werden soll. Fehlt dieser Parameter, wird das Inhaltsverzeichnis auf dem Konsolgerät ausgegeben
FOR datei	Datei oder Dateigruppe (durch Blanko-Dateinamen festgelegt), deren Inhaltsverzeichniseinträge aufgelistet werden sollen. Fehlt der Eintrag **FOR datei,** wird das ganze Inhaltsverzeichnis aufgelistet.
:Fx:	Ausgabe des Inhaltsverzeichnisses der Diskette in einem der Laufwerke :F1: bis :F5:. Fehlt dieser Parameter, wird das Inhaltsverzeichnis der Diskette in :FØ: ausgegeben. Werden mehrere Laufwerknummern angegeben, wird nur die letzte wirksam. Die Laufwerknummer hebt auch jede Gerätefestlegung in FOR datei auf.
I	Es werden alle Dateinamen, einschließlich der mit dem Attribut ›Unsichtbar‹, ausgegeben. Fehlt der Parameter, werden nur die Dateinamen aufgeführt, bei denen das Attribut ›Unsichtbar‹ nicht gesetzt ist.
F	Es wird nur der Dateiname mit seiner Erweiterung ausgegeben.

Ausgabeformat

DIRECTORY OF DISKETTE: Laufwerk: Name der Diskette
NAME .EXT BLKS LENGTH ATTR
Darin ist ›NAME.EXT‹ das Etikett der Diskette, die durch das **Format-Kommando** zugewiesen worden ist. Es hat dieselbe Syntax wie ein Dateiname (Kapitel 3). Am Ende der Liste wird die belegte Anzahl von Blöcken ausgegeben.

Beispiel 1

Auflisten aller Dateien, deren Attribute ›Unsichtbar‹ nicht gesetzt sind.



Beispiel 2
Ausgabe wie Beispiel 1, jedoch in Kurzform

```
-DIR F
DIRECTORY OF DISKETTE :FØ:ISOOAB.SYS
PROGA HEX
SUMS
936/2ØØ2 BLOCKS USED
```

Beispiel 3
Ausgabe aller ISIS-II-Systemdateien an den Matrixdrucker

```
-DIR FOR ISIS.* TO :LP:
```

Beispiel 4
Ausgabe des Inhaltsverzeichnisses der Diskette im Laufwerk 1 in Kurzform

```
-DIR 1 F
DIRECTORY OF DISKETTE :F1:ISIS.V1Ø
TYPE.M8Ø
TYPE.HEX
TYPE
1337/2ØØ2 BLOCKS USED
```

4.3.7. dateiname-Kommando (Ausführen eines Programms in einer Diskettendatei)

Die ISIS-II-Kommandoliste kann beliebig erweitert werden durch Anwenderprogramme in absolutem ausführbarem Format, wobei der Name der Anwenderdatei als Kommando benutzt wird. Das Programm wird geladen, und die Steuerung geht an die Startadresse dieses Programms über. Das Programm kann nun den Rest der Kommandozeile (die programmspezifischen Parameter) von :CI: oder das gesamte Kommando lesen, wenn vor dem Lesebefehl der Systemaufruf RESCAN gegeben wird. Die Kommandozeile darf eine Gesamtlänge von 122 Zeichen nicht überschreiten. Wenn das geladene Programm den Rest der Zeile nicht liest, wird diese vor der Ausführung des nächsten Kommandos oder Systemaufrufs gelöscht.

(Anmerkung: :CI: ist immer eine zeilenaufbereitete Datei)

Da auch ein ISIS-Kommando ein ablauffähiges Programm (eine Datei) auf der Diskette darstellt, kann dessen Name geändert werden.

Als Beispiel soll das Kommando DELETE in DEL umbenannt werden. Vorher ist der »Schreibschutz« rückzusetzen und dann mit dem RENAME-Kommando die Datei umzubenennen. Anschließend sollte das Attribut »Schreibschutz« wieder gesetzt werden.

4.3.8. FORMAT-Kommando (Initiieren einer neuen Diskette)

Allgemeines
Das Kommando FORMAT initiiert eine neue Diskette im Laufwerk 1, 2, 3, 4 oder 5.

Achtung!
Werden nur die Dateien, bei denen das Attribut »Format« gesetzt ist, mit dem FORMAT-Kommando übertragen, ist das Ergebnis keine Systemdiskette. Sie kann deshalb nur im Laufwerk 1

bis 5 eingesetzt werden. Systemdisketten sollten nicht durch Übertragen von Systemdateien mit dem Kommando COPY erzeugt werden. Systemdisketten sind immer mit dem Kommando FORMAT und dem Parameter S zu erzeugen.

Format

FORMAT {:Fx:}etikett {A / S}

Es bedeuten

FORMAT	Name des Kommandos
:Fx:	zu formatierende Diskette im Laufwerk 1 bis 5. Die Angabe :FØ: ist nicht möglich. Fehlt der Parameter, wird die Diskette im Laufwerk :F1: formatiert.
etikett	Name der Diskette. Der Aufbau von ›etikett‹ ist derselbe wie der eines Dateinamens: name.erw wobei bis zu 6 Zeichen für ›name‹ und bis zu 3 Zeichen für die Erweiterung ›erw‹ zugelassen sind
A	Alle Dateien von :FØ: auf :Fx: kopieren. Die neue Diskette enthält alle Dateien der alten Diskette und ist deshalb eine Systemdiskette.
S	Alle Dateien, bei denen das Attribut ›System‹ oder ›Format‹ gesetzt ist, von :FØ: auf :Fx: kopieren. Die neue Diskette ist deshalb eine Systemdiskette.
	Wird kein Parameter angegeben, werden nur diejenigen Dateien auf die Diskette in :Fx: übertragen, bei denen das Attribut ›Format‹ gesetzt ist.

Beispiel 1
In diesem Beispiel wird eine neue Diskette im Laufwerk 1 eingerichtet, sie erhält den Namen LIB.V1. Auf der neuen Diskette werden nur die 4 Formatdateien (ISIS.DIR, ISIS.MAP, ISIS.TO, ISIS.LAB) erzeugt.

```
-FORMAT LIB.V1
NON SYSTEM DISKETTE

-
```

Beispiel 2
In diesem Beispiel wird eine neue Systemdiskette erstellt.

```
-FORMAT ISØØAA.DUP S
COPYING SYSTEM FILES
ISIS.BIN
ISIS.CLI
ATTRIB
COPY
DELETE
DIR
HEX OBJ
RENAME
```

ASM8Ø
EDIT
ICE8Ø
FORMAT

=

Beispiel 3
In diesem Beispiel werden alle Dateien der Systemdiskette sichergestellt.

=FORMAT ISØØA.BAK A
COPYING ALL FILES

(ISIS antwortet wie in Beispiel 2 und gibt zusätzlich die Namen der Dateien aus, die nicht Systemdateien sind).

=

Erzeugen einer Sicherungskopie für eine Nichtsystemdiskette

Sicherungskopien von Nichtsystemdisketten (unter Verwendung von FORMAT mit dem A-Parameter) werden erzeugt, indem man FORMAT im Testmodus ablaufen läßt. Dazu verwendet man folgendes Verfahren:

1. Unter der Voraussetzung, daß sich eine Systemdiskette im Laufwerk Ø befindet, ist folgendes DEBUG-Kommando einzugeben:

DEBUG FORMAT etikett A

2. Wenn das Aufforderungszeichen (.) des Monitors an der Konsole erscheint, ist die Systemdiskette vom Laufwerk Ø zu entnehmen und die zu kopierende Nichtsystemdiskette einzuschieben. In das Laufwerk 1 ist eine leere Diskette einzuschieben.

3. Eingabe des Monitor-G-Kommandos

4. Ist die Formatierung vollständig, hält das System an, ohne ein Aufforderungszeichen an die Konsole auszugeben, da keine Systemdiskette im Laufwerk Ø ist. Die Nichtsystemdiskette ist durch eine Systemdiskette im Laufwerk Ø zu ersetzen und das System urzuladen, indem man die Tasten Urladen und Rücksetzen am Bedienungsfeld des SME betätigt.

4.3.9. HEXOBJ-Kommando
(Umwandeln von ISIS I Hexadezimalformat in einen ISIS II absoluten Objektmodul)

Allgemeines
Das Kommando HEXOBJ wandelt den Objektcode im Hexadezimalformat in einen absoluten Objektmodul um, der kompatibel mit ISIS II ist. Das Hexadezimalformat wird erzeugt von Cross-Übersetzerprogrammen, die auf DVA-Anlagen laufen, und von Assemblern der früheren ISIS-Versionen.

Format
HEXOBJ {:Fx:}hexdatei TO {:Fx:}absdatei START(adr)

Es bedeuten

HEXOBJ	Name des Kommandos
:Fx: (Eingabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die ›hexdatei‹ von der Diskette im Laufwerk Ø gelesen.
hexdatei	Datei mit dem hexadezimalen Maschinencode. Dieses Format ist im Anhang D der SME-Bedienungsanleitung beschrieben.
:Fx: (Ausgabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird ›absdatei‹ auf die Diskette im Laufwerk Ø geschrieben.
absdatei	Ausgabedatei, die den absoluten Objektmodul enthält und unter ISIS II geladen werden kann. Der mit dem Objektmodul gespeicherte Modulname ist derselbe wie der Namensteil des Dateinamens von **›absdatei‹**. Der durch HEXOBJ erzeugte absolute Objektmodul enthält **keine** Symboltabelle.
START (adr)	wird verwendet, um eine Startadresse (die Adresse des ersten auszuführenden Befehls) in den absoluten Objektmodul einzubringen. Die Adresse kann festgelegt werden durch eine hexadezimale, dezimale, oktale oder binäre Zahl, gefolgt von einem Buchstaben, der die BASIS kennzeichnet. Der Buchstabe H steht für hexadezimal, O oder Q für oktal, B für binär und D oder leer für dezimal. Entsprechend wird die Adresse $3ØØØ_{(16)}$ als START (3ØØØH) festgelegt. Wenn START(adr) fehlt, wird die Startadresse der Datei dem Hexadezimalformat entnommen. Diese wird bestimmt durch die END-Anweisung bei Assemblerprogrammen oder bei PL/M-Programmen durch die numerische Marke vor der ersten Anweisung oder durch die $H-Steuerung beim Durchlauf 2 des Übersetzers. Falls auf keinem der beiden Wege eine Startadresse festgelegt ist, wird Ø angenommen. Dies liegt aber im ISIS-II-Bereich und bewirkt einen Fehler 15, wenn sie geladen und gestartet werden soll.

Beispiel 1
Umwandeln einer Objektdatei in Hexadezimalformat, auf Lochstreifen in einen absoluten Objektmodul auf Diskette.

```
-HEXOBJ :HR: TO SUMS
-
```

Beispiel 2
Umwandeln einer Datei im Hexadezimalformat in das absolute Objektformat und Festlegen einer Startadresse

```
-HEXOBJ PRIME.HEX TO PRIME.OBJ START(32ØØH)
```

4.3.10. OBJHEX-Kommando (Umwandeln von einem ISIS II absoluten Objektmodul in ISIS I Hexadezimalformat)

Allgemeines
Das Kommando OBJHEX wandelt einen absoluten Objektmodul in das Hexadezimalformat um.

Format
OBJHEX {:Fx:}absdatei TO {:Fx:}hexdatei

Es bedeuten

OBJHEX	Name des Kommandos
:Fx: (Eingabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die ›hexdatei‹ von der Diskette im Laufwerk Ø gelesen.
absdatei	Datei, die den absoluten Objektmodul enthält
:Fx: (Ausgabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird ›absdatei‹ auf die Diskette im Laufwerk Ø geschrieben.
hexdatei	Datei, die den hexadezimalen Objektcode enthalten soll, der vom ISIS-II-Format umgewandelt worden ist. Die Startadresse (die Adresse des ersten auszuführenden Befehls) wird der Datei **›absdatei‹** entnommen. Der durch OBJHEX erzeugte hexadezimale Objektcode enthält **keine** Symboltabelle.

4.3.11. RENAME-Kommando (Ändern des Namens einer Diskettendatei)

Allgemeines

Mit dem RENAME-Kommando wird der Dateiname einer Diskettendatei geändert. Von der Änderung ist nur das Inhaltsverzeichnis betroffen.

Wird in dem Kommando RENAME versucht, eine Datei umzubenennen, deren Attribute ›Schreibschutz‹ oder ›Format‹ gesetzt sind, wird folgende Meldung über die Konsole ausgegeben:

neuname, ALREADY EXISTS, DELETE?

Werden daraufhin Y oder y und Wagenrücklauf eingegeben, wird der Name im Inhaltsverzeichnis gelöscht und der neue Name zugeordnet. Wird ein anderes Zeichen eingegeben, wird das Kommando nicht ausgeführt und die Steuerung ISIS übergeben,

Format
RENAME {:Fx:}altname TO neuname

Es bedeuten

RENAME	Name des Kommandos
:Fx: (Eingabe/ Ausgabe)	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird das Inhaltsverzeichnis von der Diskette im Laufwerk Ø gelesen und anschließend wieder auf die Diskette im Laufwerk Ø geschrieben.
altname	Name der Datei, die umbenannt werden soll.
neuname	Name, der der Datei neu zugeordnet werden soll. Der Geräteteil des Dateinamens muß derselbe sein wie von ›altname‹. Werden verschiedene Geräte angegeben, wird ein Fehler gemeldet.

Beispiel 1
In diesem Beispiel wird der Name der Datei CHAP1 auf der Diskette im Laufwerk Ø in CHAP.ONE geändert.

```
-RENAME CHAP1 TO CHAP.ONE
```

Beispiel 2
In diesem Beispiel wird versucht, den Dateinamen SYSTEM.DIR, der mit dem Attribut ›Schreibschutz‹ versehen ist, zu ändern.

```
-RENAME SYSTEM.DIR TO DIRECT.SYS
SYSTEM.DIR, WRITE PROTECTED

-
```

Beispiel 3
In diesem Beispiel ist ›neuname‹ der Name einer vorhandenen Datei.

```
-RENAME :F1:COSOL.BAK TO :F1:CONSOL.TXT
CONSOL.TXT, ALREADY EXISTS, DELETE?
Y
-
```

4.3.12. SUBMIT-Kommando
(Bereitstellen von Kommandos für Nichtdialogverarbeitungs-Prozeduren)

Allgemeines
Das Kommando SUBMIT bewirkt, daß ISIS II seine Anweisungen vorübergehend von einer Diskettendatei, statt von der Konsole, erwartet. Dazu muß auf einer Diskette eine Prozedurdatei bereitgestellt werden. Diese kann beim Aufruf mit aktuellen Parametern versorgt werden.

Format
SUBMIT {:Fx:}dateiname.Erweiterung {(param Ø, ... param 9)}

Es bedeuten

SUBMIT	Name des Kommandos
:Fx:	Diskette im Laufwerk 1 bis 5. Fehlt der Parameter, wird die Prozedurdatei von der Diskette im Laufwerk Ø gelesen.
dateiname	Diskettendateiname, in der die definierte Kommandofolge (Prozedur) steht. Die Prozedur kann formale Parameter enthalten.
.Erweiterung	entsprechend den Konventionen zum Aufbau von Dateinamen. Fehlt die Angabe, sucht SUBMIT nach der Erweiterung .CSD.
(param Ø... param 9)	Aktuelle Parameter, die die formalen Parameter in der Prozedurdefinition ersetzt. Die maximale Parameterzahl ist 1Ø. Wenn ein Parameter in der Liste weggelassen wird, muß seine Position durch ein Komma angezeigt werden. Ein Parameter ist eine Zeichenfolge mit maximal 31 Zeichen. Falls ein Parameter ein Komma, ein Leerzeichen oder eine schließende Klammer enthält, muß der Parameter in Apostrophe eingeschlossen werden. Um einen Apostroph innerhalb eines in Apostrophen stehenden Parameters zu verwenden, sind an seiner Stelle zwei Apostrophe zu setzen.

Bemerkungen:
SUBMIT kopiert die Kommandofolgedefinition Prozedur in eine zweite Datei (Kommandodatei) und ersetzt darin die formalen Parameter durch die aktuellen Parameter. Die Kommandodatei hat denselben Namen wie die Prozedurdatei, jedoch mit der Erweiterung .CS.

SUBMIT ändert die Zuweisung vom Konsoleingabegerät zu der von SUBMIT erzeugten Kommandodatei und gibt die Steuerung an ISIS II zurück, das dann die Kommandos in der Kommandodatei ausführt. Das letzte Kommando in der Kommandodatei ist ein spezielles SUBMIT-Kommando, wodurch das Konsoleingabegerät wieder seine frühere Zuweisung zurückerhält und die Kommandodatei gelöscht wird.

Formale Parameter werden in der Kommandofolgedefinition durch die zwei Zeichen % n festgelegt, wobei n eine Ziffer von Ø bis 9 ist. Sie können an beliebiger Stelle in der Kommandofolgedefinition erscheinen. Soll ein %-Zeichen in einer Kommandofolgedefinition nicht als Formalparameter gelten, so ist das Zeichen CTRL-P (^P) diesem %-Zeichen vorzustellen.

Jedes Programm, das seine Kommandos von :CI: liest, kann im Nichtdialogbetrieb ausgeführt werden. Die Kommandofolgedefinitionsdatei (Prozedurdatei) kann auch Kommandos und Parameter für die Anwenderprogramme enthalten. In einer Kommandofolgedefinitionsdatei kann wieder ein SUBMIT-Kommando verwendet werden, wodurch eine weitere Kommandodatei erzeugt wird. Dieses Schachteln von SUBMIT-Kommandos kann wiederholt werden.

Ein CTRL-E (^E) in einer SUBMIT-Datei schaltet die Konsoleingabe von der Kommandodatei auf die ursprüngliche Konsole um und erlaubt dadurch eine Dialogverarbeitung. Ein CTRL-E, das an der ursprünglichen Konsole eingegeben wird, gibt die Steuerung an die Kommandodatei zurück. Falls die Steuerung nicht an die Kommandodatei zurückgegeben wird oder ein Fehler auftritt, nachdem eine Kommandofolge die Verarbeitung begonnen hat, geht die Steuerung an ISIS II zurück. Dabei wird die .CS-Datei nicht gelöscht.

SUBMIT eröffnet eine Datei und benötigt dazu zwei Puffer. Dies bedeutet, daß jedes unter SUBMIT laufende Programm diese zusätzlichen Puffer zulassen muß (vgl. Kapitel 11).

Beispiel 1
Die folgenden Beispiele zeigen eine PL/M-Übersetzung, ausgeführt auf einem System mit vier Laufwerken und im Nichtdialogbetrieb. Das PL/M-80-Kommando hat nur 3 Einträge, die sich ändern können. Die Verwendung von SUBMIT zur Kommandoeingabe automatisiert den Vorgang und erspart Eingaben von der Konsole.

Die Kommandofolgedefinition steht in der Datei PL/M8Ø.CSD. Für eine Erklärung der Steueranweisungen im PL/M-80-Kommando wird auf die im Vorwort angeführte Bedienungsanleitung verwiesen. Die Datei PL/M8Ø.CSD enthält folgendes:

```
PLM8Ø%Ø.%1 DEBUG XREF PRINT(:F3:%Ø.LST) &
DATE (%2) TITLE ('%Ø.%1 COMPILATION')
```

Diese Kommandofolgedefinition enthält 3 Parameter, gekennzeichnet durch %Ø, %1 und %2.

Das zum Start der Übersetzung benutzte SUBMIT-Kommando lautet:

```
SUBMIT PLM8Ø (:F1:PROGA,SRC, '9SEPT 76')
```

Die Kommandofolge, die von SUBMIT erzeugt und dann ausgeführt wird, lautet:

```
PLM8Ø :F1:PROGA.SRC DEBUG XREF PRINT (:F3:PROGA.LST) &
DATE ('9 SEPT 76') TITLE ('PROGA.SRC COMPILATION')
SUBMIT RESTORE PLM8Ø.CS (:VI:)
```

Beispiel 2
Dieses Beispiel zeigt eine PL/M-Übersetzung mit einem LINK- und einem LOCATE-Lauf, ausgeführt von einer SUBMIT-Datei auf einem System mit 2 Laufwerken. Ein CTRL-E (^E) ist in die Kommandofolgedefinition hinter der PL/M-Übersetzung eingetragen worden, um die Übersetzung auf Fehlerfreiheit zu überprüfen. Anschließend wird durch Eingabe von CTRL-E die Verarbeitung fortgesetzt.

Die Datei CMPLNK.CSD im Laufwerk 1 enthält die folgende Kommandofolgedefinition:

```
PLM8Ø %Ø.%1 DEBUG XREF DATE (%2) &
TITLE ('%Ø.%1 COMPILATION')
^E
LINK %Ø.OBJ,SYSTEM.LIB TO %Ø.SAT &
PRINT (%Ø.LNK) MAP
LOCATE %Ø.SAT PRINT (%Ø.LOC) MAP
```

Das SUBMIT-Kommando, das eingegeben wird, um PROGA.SRC zu übersetzen, zu binden und absolut zu adressieren lautet:

```
SUBMIT :F1:CMPLNK (:F1:PROGA.SRC, '3 OCT 76')
```

Die tatsächlich ausgeführte Kommandofolge wird auf dem Konsolausgabegerät zur Kontrolle angezeigt.

```
-PLM8Ø :F1:PROGA.SRC DEBUG XREF DATE (3 OCT 76) &
TITLE ('PROGA.SRC COMPILATION')
^E^E
-LINK :F1:PROGA.OBJ,SYSTEM.LIB TO :F1:PROGA.SAT &
PRINT (:F1:PROGA.LNK) MAP
-LOCATE :F1:PROGA.SAT PRINT (:F1:PROGA.LOC) MAP
```

5. ISIS-II-Texteditor

5.1. Einführung

Mit dem Texteditor kann der Anwender Textdateien im ASCII-Code erstellen und aufbereiten. Der Texteditor ist zeichenorientiert, es können aber auch Zeilen angesprochen werden. Ein oder mehrere Zeichen können in einer Textzeile eingefügt, gelöscht oder geändert werden. Einfügungen und Löschungen können auch in mehr als einer Textzeile durchgeführt werden. Die Einfügungen oder Löschungen können an beliebiger Stelle gemacht werden: am Anfang des Textes, am Ende des Textes, am Anfang einer Zeile oder an irgendeiner Stelle des Textes. Zeilennummern oder andere unwesentliche Zusatzinformationen werden nicht benötigt.

5.1.1. Allgemeiner Ablauf der Textaufbereitung

Der übliche Vorgang zum Erzeugen einer neuen Textdatei ist, den Texteditor aufzurufen, Text über das Konsolgerät einzugeben, die jeweils gewünschten Textaufbereitungen durchzuführen und dann die Textdatei zur Speicherung auf eine Diskette auszugeben. Um eine vorhandene Datei aufzubereiten, ist der Texteditor aufzurufen, der Text von einer Diskettendatei einzugeben, der Text aufzubereiten und die aufbereitete Version auf eine Diskettendatei auszugeben.

5.1.2. Speicherbedarf, Arbeitsspeicher und Textpuffer

Der Texteditor belegt etwa 7400 Bytes des RAM-Speichers, oberhalb der residenten ISIS-II-Routinen, die 12 KBytes benötigen. Der ganze restliche RAM-Speicher steht dem Editor als Arbeitsspeicher zur Verfügung. Der Textpuffer ist der Teil des Arbeitsspeichers, der vom Texteditor laufend verwendet wird, um den Text zu speichern. Die Größe des Textpuffers ändert sich; er wird größer, wenn Text eingegeben wird, und wird kleiner, wenn Text gelöscht wird. Bei leerem Puffer decken sich Pufferanfang und -ende.

5.1.3. Pufferzeiger

Mit dem Pufferzeiger wird die Stelle im Textpuffer definiert, an der die Aufbereitung geschehen soll. Der Pufferzeiger kann wie folgt positioniert werden:

1. vor das erste Zeichen im Textpuffer (Pufferanfang)
2. unmittelbar hinter das letzte Zeichen im Textpuffer (Pufferende)
3. zwischen zwei benachbarte Zeichen im Textpuffer.

Der Pufferzeiger wird niemals direkt auf ein einzelnes Zeichen positioniert, sondern davor oder dahinter. Ein gegebener Text wird im Puffer **unmittelbar vor dem Pufferzeiger** abgelegt. Man kann den Pufferzeiger auf eine beliebige Stelle innerhalb des Puffers positionieren. Jedes Kommando, das den Pufferzeiger jenseits der Textpuffergrenzen zu verschieben sucht, wird beendet, sobald der Pufferzeiger die Puffergrenze erreicht hat, selbst wenn das angegebene Kommando noch nicht abgeschlossen ist.

Der Pufferzeiger kann zeichen- oder zeilenweise verschoben werden. Eine Textzeile im Textpuffer ist eine Zeichenfolge, die einen Zeilenvorschub als letztes Zeichen hat. Das nächste Zeichen im Textpuffer, das unmittelbar auf den Zeilenvorschub folgt, steht in der nächsten Zeile. Werden im Text keine Zeilenvorschubzeichen verwendet, wird der gesamte Text als eine einzige Zeile betrachtet. Zeilenvorschübe werden automatisch nach einem Wagenrücklauf, jedesmal beim Betätigen der Wagenrücklauftaste eingefügt.

5.2. Aufruf des Texteditors

Der Texteditor steht in einer Datei mit dem Namen EDIT auf der ISIS-II-Systemdiskette. Er wird durch Eingeben des Dateinamens als ISIS-Kommando aufgerufen. Der Texteditor kann nicht von einer SUBMIT-Datei her ausgeführt werden. Das EDIT-Kommando hat folgende Syntax:

EDIT dateiname1 {TO dateiname2}

›dateiname 1‹ ist entweder der Name einer neuen, zu erzeugenden und aufzubereitenden Datei oder der Name einer vorhandenen Datei, die aufbereitet werden soll. Wenn ›dateiname1‹ keine Diskettendatei ist, muß sie eine Eingabedatei sein. Außerdem muß dann ›dateiname2‹ angegeben werden, und dies muß eine Ausgabedatei sein.

Beispiel
EDIT :HR: TO :F1:ASSY.SRC

Dieses Kommando in Verbindung mit einem darauffolgenden A-Kommando liest vom schnellen Lochstreifenleser in den Textpuffer ein. Ist die Aufbereitungssitzung beendet (E-Kommando), wird der Inhalt des Textpuffers in die Diskettendatei ASSY.SRC auf Laufwerk 1 geschrieben.

Wenn ›dateiname1‹ eine Diskettendatei ist und ›dateiname2‹ nicht angegeben ist, hängt die vom Texteditor ergriffene Maßnahme davon ab, ob ›dateiname1‹ eine neue oder eine bereits vorhandene Datei ist. Wenn ›dateiname1‹ neu ist, wird sie als Ausgabedatei erzeugt. Wenn sie vorhanden und nicht schreibgeschützt ist, wird sie zur Eingabe eröffnet. Eine Sonderdatei mit dem Namen EDIT.TMP wird zur (temporären) Ausgabe erzeugt. Ist die Aufbereitungssitzung mit dem EXIT-Kommando beendet worden, werden die Dateien geschlossen und folgende Maßnahmen durchgeführt.

a) ›dateiname1‹ wird umbenannt. Der Hauptname bleibt derselbe, jedoch mit der Erweiterung .BAK

b) EDIT.TMP wird in ›dateiname1‹ umbenannt.

Beispiel
Eine Aufbereitungssitzung ist begonnen worden mit:

EDIT FILE.TXT

Am Ende der Sitzung stehen die Dateien FILE.BAK und FILE.TXT auf der Diskette in Laufwerk Ø. Die Datei FILE.BAK ist eine Sicherungsversion, die den Zustand der Textdatei vor der Aufbereitungssitzung darstellt.

›dateiname2‹ ist, falls angegeben, der Name der Ausgabedatei, in die das Ergebnis der Aufbereitungssitzung geschrieben wird. Wenn ›dateiname2‹ eine Diskettendatei ist, hängt die Maßnahme des Texteditors davon ab, ob ›dateiname2‹ eine neue oder eine vorhandene Datei ist. Wenn sie eine neue Datei ist, erzeugt sie der Texteditor zur Ausgabe. Wenn ›dateiname2‹ bereits vorhanden und nicht schreibgeschützt ist, wird sie zur Ausgabe eröffnet. Dies bewirkt, daß der frühere Inhalt verloren geht. Falls ›dateiname1‹ und ›dateiname2‹ angegeben und die Namen dieselben sind, wird keine Sicherungsdatei erzeugt.

Beispiel
EDIT :F1:Q3.SRC TO :F1:Q3.SRC

In diesem Falle wird für die Datei Q3.SRC auf Laufwerk 1 keine Sicherungsdatei erzeugt. Dasselbe gilt für das folgende Beispiel.

EDIT :F1:Q3.BAK
Diese Möglichkeit ist von Nutzen, wenn man große Dateien aufbereitet und der Diskettenplatz beschränkt ist.

Der Dateiname EDIT.TMP sollte für die Anwendung des Texteditors reserviert werden, weil jede Datei mit dem Namen EDIT.TMP, die auf derselben Diskette wie die Eingabedatei steht, vom Texteditor überschrieben wird. Zu beachten ist, daß beim Abbruch einer Aufbereitungssitzung die temporäre Ausgabedatei EDIT.TMP auf der Diskette stehen bleibt.

Beispiel
Eröffnen und Aufbereiten einer neuen Datei mit dem Namen PROGA.SRC auf Laufwerk 1. Man gibt an der Konsole nach dem ISIS-II-Aufforderungszeichen (–) folgendes ein:

–EDIT PROGA.SRC

Das System antwortet dann:

ISIS-II TEXT EDITOR, Vn.n
NEW FILE
*

Die Versionsnummer des Texteditors wird als »Vn.n« ausgegeben. »NEW FILE« erscheint, um die Eröffnung einer neuen Datei zu bestätigen. Am Ende gibt der Texteditor einen Stern (*) als Aufforderungszeichen zur weiteren Eingabe aus.

Beispiel
Aufbereitung einer bereits vorhandenen Datei namens GEN5.TXT auf Laufwerk 1. Eingabe an der Konsole:

–EDIT :F1:GEN5.TXT

Das System antwortet:

ISIS-II TEXT EDITOR, Vn.n
*

Die Meldung NEW FILE wird diesmal nicht ausgegeben, weil eine schon vorhandene Datei aufbereitet werden soll.

5.3. Editorkommandos und Kommandosyntax

Der Texteditor meldet seine Bereitschaft, Kommandos anzunehmen, durch das Aufforderungszeichen Stern (*). Kommandos können mit Groß- oder Kleinbuchstaben eingegeben werden; z. B. kann das Einfügungskommando 'I' oder 'i' lauten.

Kommandos können einzeln oder als Kommandofolge eingegeben werden. Die Kommandos in einer Kommandofolge werden in der Reihenfolge ausgeführt, in der sie eingegeben worden sind. Eine Kommandofolge wird durch zwei aufeinanderfolgende ESC-Zeichen (oder ALT MOD-Zeichen bei einigen Konsolgeräten) beendet. Die ESC-Zeichen werden auf dem Konsolgerät als Dollarzeichen ($) ausgegeben. Zeichenfolgen innerhalb von Kommandos müssen jedoch mit nur einem ESC-Zeichen beendet werden. Ein Ersetzungskommando enthält eine neue Zeichenfolge, die an Stelle einer alten Zeichenfolge in eine Datei eingefügt werden soll.

Beispiele
*SDAS BEDIENUNGSGERÄT$DAS BEDIENGERÄT $$
ersetzt DAS BEDIENUNGSGERÄT durch DAS BEDIENGERÄT,

*$ALTEN TEXT$NEUEN TEXT$$
ersetzt ALTEN TEXT durch NEUEN TEXT.

Kommandofolgen werden von der oberen Grenze des verfügbaren RAM-Speichers nach abwärts gespeichert. Das erste Kommandozeichen wird in der höchsten verfügbaren Stelle gespeichert, die nachfolgenden Zeichen darunter. Wenn ein Kommando den Textpuffer zu überschreiben versucht, ertönt eine Klingel (oder ein Pfeifton bei einigen Konsolgeräten). Dann ist die Taste »Rubout« zu drücken, um das eingegebene Kommando zu löschen. Anschließend schreibt man einen Teil des Puffers mit dem W-Kommando aus, um Platz für weitere Aufbereitung freizubekommen.

5.3.1. Wagenrücklauf und Zeilenvorschub

Wenn man Text eingibt, erzeugt der Texteditor automatisch einen Zeilenvorschub nach jedem Wagenrücklauf. Deshalb muß man zum Löschen eines Wagenrücklaufs zwei Zeichen entfernen, nämlich zusätzlich den Zeilenvorschub.

5.3.2. Unterbrechungskommandos (CTRL-C)

Eine Kommandofolge kann während der Eingabe ungültig gemacht werden, sofern die Kommandoendekennung noch nicht eingegeben worden ist. Wenn es nötig ist, eine ganze Kommandofolge ohne Rücksicht auf ihren Inhalt zu löschen, entfernt ein CTRL-C die ganze Kommandofolge und veranlaßt den Texteditor, ein Aufforderungszeichen für ein neues Kommando auszugeben.

Nachdem die Kommandoendekennung eingegeben worden ist und die Kommandos in Ausführung sind, kann der Ablauf durch CTRL-C beendet werden. CTRL-C wird nur während eines Ausgabekommandos wirksam. Wenn CTRL-C während einer langen Kommandofolge oder einer Kommandowiederholung (siehe weiter unten) eingegeben wurde, ist der aktuelle Stand der Verarbeitung nicht immer leicht feststellbar.

5.3.3. Löschen von Tippfehlern (Rubout und CTRL-X)

Alle Tippfehler bei der Eingabe können durch Anschlagen der Taste Rubout entfernt werden, je einmal für jedes zu entfernende Zeichen. Sobald ein Zeichen, beginnend bei dem zuletzt eingegebenen, gelöscht ist, wird es vom Texteditor zur Kontrolle auf dem Bediengerät ausgegeben (»geechot«). Die Eingabe von CTRL-X bewirkt, daß alle Zeichen vom Kommando aus bis zum letzten Wagenrücklauf-Zeichenvorschub gelöscht werden.

5.3.4. Setzen von Tabulatorzeichen (CTRL-I)

Der Texteditor erzeugt ein horizontales Tabulatorzeichen, wenn CTRL-I eingegeben (oder bei einigen Konsolgeräten die TAB-Taste gedrückt) wird. Ein CTRL-I wird im Textpuffer als einzelnes Zeichen gespeichert. Das System nimmt dieses Zeichen zur Erzeugung einer genügenden Anzahl von Leerzeichen, um das nächste Zeichen auf die nächste Tabulatorstelle zu positionieren. Tabulatorzeichen sind auf jeder achten Zeichenstelle der Textzeilen gesetzt (Ø, 8, 16, 24 usw.).

5.4. Beschreibungen der Kommandos

5.4.1. Die Grundkommandos I und T

Die Kommandos I und T erlauben, Text in den Puffer einzufügen und den Text auf dem Konsolgerät auszugeben, also zur Überprüfung wiederholen zu lassen.

I-Text in Textpuffer einfügen

Das Kommando I wird verwendet, um vom Bediengerät aus Text in den Textpuffer einzufügen. Der neue Text wird unmittelbar vor dem Pufferzeiger eingefügt. Das Kommando hat folgendes Format:

*Itext$$

Dabei kann **text** auch Wagenrückläufe enthalten. Nachdem der Texteditor den Buchstaben I als Kommando erkannt hat, nimmt er die anschließende Eingabe als Text an, bis er auf ein ESC- oder ALT MODE-Zeichen trifft.

Beispiel
ISEIN ODER NICHTSEIN ‹WR›
DAS IST HIER DIE FRAGE ‹WR›
$$
In diesem Beispiel werden zwei Zeilen vor dem Pufferzeiger eingefügt.

T-Textpuffer auf Konsolgerät ausgeben

Um den eingegebenen Text zur Überprüfung der korrekten Eingabe wieder ausgegeben zu bekommen, verwendet man das Kommando T. Das Kommando hat folgende Syntax:

*nT$$

Dabei ist ›n‹ eine ganze Dezimalzahl von –65535 bis +65534. Wenn ›n‹ positiv ist, wird der Text ausgegeben von der augenblicklichen Position des Pufferzeigers vorwärts bis zum n-ten Wagenrücklauf. Wenn ›n‹ negativ ist, beginnt die Ausgabe bei ›n‹ Zeilen vor dem Anfang der laufenden Zeile (der Zeile, in der der Pufferzeiger steht) und setzt sich fort, bis der Pufferzeiger erreicht ist. Wenn n = Ø ist, wird vom Anfang der laufenden Zeile bis zum Pufferzeiger ausgegeben. Wenn kein Wert für ›n‹ festgelegt ist, nimmt der Texteditor als Standardwert 1 an. Dies bewirkt, daß vom Pufferzeiger bis zum Ende der laufenden Zeile ausgegeben wird.

5.4.2. Positionieren des Pufferzeigers

Die folgenden vier Kommandos des Texteditors, B, Z, L und C verschieben alle den Pufferzeiger.

B verschiebt den Pufferzeiger zum Anfang des Textpuffers.
Z verschiebt den Pufferzeiger zum Ende des Textpuffers.
L verschiebt den Pufferzeiger eine festgelegte Anzahl von Zeilen vorwärts oder rückwärts.
C verschiebt den Pufferzeiger eine festgelegte Anzahl von Zeichen vorwärts oder rückwärts.

B-Pufferzeiger zum Textpufferanfang verschieben

Das Kommando B, das den Pufferzeiger zum Anfang des Textpuffers verschiebt, hat mehrere Anwendungen, z.B.:

Setzen eines Bezugspunktes für das Zählen von Textzeilen oder für das Suchen nach einem Wort oder einer Wortverbindung.

Definieren eines Anfangspunktes, wenn der gesamte Inhalt des Textpuffers angegeben werden soll.
Einfügen von Text am Anfang des Textpuffers, also vor dem bereits im Puffer vorhandenen Text.

Z-Pufferzeiger zum Textpufferende verschieben

Das Kommando Z verschiebt den Pufferzeiger zum Ende des Textpuffers, also unmittelbar hinter das letzte Zeichen im Puffer. Es wird hauptsächlich verwendet, um Text am Ende des Puffers anzufügen.

L-Pufferzeiger zum Zeilenanfang verschieben

Das Kommando L verschiebt den Pufferzeiger um eine festgelegte Anzahl von Zeilen vorwärts oder rückwärts. Das Kommando hat folgendes Format:

*nL$$

Dabei ist ›n‹ eine ganze Dezimalzahl von –65535 bis +65534. Ein positiver Wert von ›n‹ verschiebt den Pufferzeiger auf den Anfang der n-ten Zelle hinter der laufenden Zeile. Ein negativer Wert von ›n‹ verschiebt den Pufferzeiger zurück auf den Anfang der n-ten Zeile vor der laufenden Zeile. Wenn der Argumentwert – 1 oder nur – ist, wird der Pufferzeiger auf den Anfang der Zeile verschoben, die der laufenden Zeile vorangeht. Wenn n = Ø ist, wird der Pufferzeiger auf den Anfang der laufenden Zeile zurückverschoben.

C-Pufferzeiger um ein oder mehr Zeichen verschieben

Das Kommando C verschiebt den Pufferzeiger um eine festgelegte Anzahl von Zeichenstellen. Das Kommando hat folgendes Format:

*nC$$

Dabei ist ›n‹ eine ganze Dezimalzahl von –65535 bis +65534.

Wenn ›n‹ positiv ist, wird der Pufferzeiger um n Zeichen vorwärts verschoben. Wenn ›n‹ negativ ist, wird der Pufferzeiger um n Zeichen rückwärts verschoben. Wenn n weggelassen ist, wird als Standardwert 1 angenommen. Ein Wert n = Ø hat keine Auswirkung auf den Pufferzeiger.

Es ist unpraktisch, mit dem Kommando C den Pufferzeiger über große Textblöcke zu verschieben. Es wird vorteilhaft für das Verschieben des Pufferzeigers innerhalb einer Textzeile verwendet. Für das Verschieben des Pufferzeigers über Zeilen oder Abschnitte sollte eines der Kommandos L oder F verwendet werden.

5.4.3. Beispiele für Aufbereitungen mit I, T, B, Z und L

Die folgenden Beispiele verwenden fünf der bisher vorgestellten Aufbereitungskommandos, nämlich I, T, B, Z und L.

1. Angenommen, man hat Daten mit Hilfe des Kommandos I in den Textpuffer eingegeben und wünscht, den gesamten Textpuffer auszugeben. Dann kann die nachstehende Kommandofolge benutzt werden

*B5ØØT$$ Das Kommando B verschiebt den Pufferzeiger zum Anfang des Textpuffers. Das Kommando 5ØØT gibt den gesamten Inhalt des Textpuffers aus, sofern er 5ØØ oder weniger Zeilen enthält. Andernfalls werden die ersten 5ØØ Zeilen ausgegeben.

2. Man kann die nachstehende Kommandofolge verwenden, um die laufenden Textzeilen auszugeben:

*ØTT$$ Der Teil ØT der Kommandofolge gibt vom Zeilenanfang bis zum Pufferzeiger aus. Das anschließende Kommando T gibt vom Pufferzeiger bis zum Zeilenende aus.

Die laufende Textzeile kann auch mit Hilfe der nachstehenden Kommandofolge ausgegeben werden:

*ØLT$$ Der Teil ØL der Kammandofolge verschiebt den Pufferzeiger zum Zeilenanfang. Das anschließende Kommando T gibt vom Pufferzeiger bis zum Zeilenende aus.

3. Man kann den Pufferzeiger 5 Textzeilen zurückverschieben und die Zeile, in der der Pufferzeiger dann positioniert ist, durch nachstehende Kommandofolge ausgeben lassen:

*–5LT$$ Der Pufferzeiger wird auf den Anfang der fünften Zeile vor der laufenden Zeile zurückverschoben. Diese neue Zeile wird dadurch zur laufenden Zeile. Das Kommando T bewirkt, daß die Zeile ausgegeben wird.

5.4.4. Löschen von Text

Die Kommandos K und D werden verwendet, um Text zu löschen.

K-Textzeilen löschen

Das Kommando K löscht Textzeilen. Das Kommando hat folgende Syntax:

*nK$$

Dabei ist ›n‹ eine ganze Dezimalzahl von –65535 bis +65534.
Wenn ›n‹ positiv ist, wird der Text von der laufenden Position des Pufferzeigers vorwärts bis zum n-ten Wagenrücklauf Zeilenvorschub (einschließlich) gelöscht.

Wenn ›n‹ negativ ist, beginnt das Löschen bei n Zeilen vor dem Anfang der laufenden Zeile (der Zeile, in der der Pufferzeiger steht) und setzt sich fort, bis der Pufferzeiger erreicht ist. Wenn n = Ø ist, wird vom Anfang der laufenden Zeile bis zum Pufferzeiger gelöscht. Wenn kein Wert für ›n‹ festgelegt ist, nimmt der Texteditor als Standardwert 1 an. Dies bewirkt, daß vom Pufferzeiger bis zum Ende der laufenden Zeile gelöscht wird. Das Kommando wird dann beendet.

D-Zeichen im Textpuffer löschen

Das Kommando D wird verwendet, um eine festgelegte Anzahl von Zeichen im Textpuffer zu löschen. Das Kommando hat folgende Syntax:

*nD$$

Dabei ist ›n‹ eine ganze Dezimalzahl von –65535 bis +65534.

Ein positiver Wert von ›n‹ bewirkt, daß n Zeichen hinter dem Pufferzeiger gelöscht werden. Ein negativer Wert von ›n‹ bewirkt, daß n Zeichen vor dem Pufferzeiger gelöscht werden. Wenn ›n‹ weggelassen ist, wird als Standardwert 1 angenommen. Ein Kommando ØD hat keine Auswirkung.

5.4.5. Beispiele für Aufbereitungen mit B, T, L, C, K und D

Die folgenden Beispiele verwenden die bisher beschriebenen Kommandos:

1. Angenommen, man will die ersten 20 Textzeilen löschen.

*B2ØK5T$$ Diese Kommandofolge bewirkt, daß der Pufferzeiger zum Anfang des Textpuffers verschoben wird, 20 Zeilen gelöscht und die anschließenden 5 Zeilen auf dem Bediengerät ausgegeben werden. Die Kommandoendekennung wird an das Ende der Kommandofolge gesetzt, die einzelnen Kommandos benötigen keine eigenen Endekennungen.

2. Betrachtet man die folgende Textzeile, dann kann das Wort MULTUIPLIZIERT dadurch korrigiert werden, daß man den überflüssigen Buchstaben U mit dem Kommando D löscht:

;DIESE ROUTINE MULTUIPLIZIERT ZWEI 16-BIT-ZAHLEN

Da der Pufferzeiger am Anfang der Textzeile steht, muß um 19 Zeichenpositionen zur Stelle unmittelbar vor U verschoben werden. Die nachstehende Kommandofolge führt dies aus:

*19CD$$

Dies ist allerdings ein umständlicher Weg, ein Zeichen zu löschen. Er ist hier nur als Beispiel eingeflochten worden, um die Verwendung des Kommandos C zu zeigen. Ein besser geeignetes Kommando, um diese Operation durchzuführen, ist das später beschriebene Kommando S.

3. Der Textpuffer enthalte folgenden Text:

DIES IST ZEILE 1
DIES IST ZEILE 2
DIES IST ZEILE 3
DIES IST ZEILE 4
DIES IST ZEILE 5
DIES IST ZEILE 6
DIES IST ZEILE 7
DIES IST ZEILE 8
DIES IST ZEILE 9
DIES IST ZEILE 1Ø

Ein Wagenrücklauf und ein Zeilenvorschub beenden jede Zeile, sind aber hier nicht dargestellt. Die Ausgaben des Rechners sind zur besseren Kennzeichnung unterstrichen. Der Pufferzeiger befindet sich in Zeile 6 zwischen I und S des Wortes IST.

Dieses Beispiel zeigt die Wirkungsweise des Kommandos T und das Löschen mehrerer Zeilen mit den Kommandos L und K.

---

*ØT$$ gibt vom Anfang der laufenden Zeile (Zeile 6) bis zum Pufferzeiger aus.
DIES I

---

*T$$ gibt vom Pufferzeiger bis zum Zeilenende aus.

ST ZEILE 6

---

*ØTT$$ gibt die ganze Zeile aus, ohne den Pufferzeiger zu verschieben.

```
DIES IST ZEILE 6
```

*-5T$$ gibt 5 Zeilen vor der laufenden Zeile und die laufende Zeile bis zum Pufferzeiger aus.

```
DIES IST ZEILE 1
DIES IST ZEILE 2
DIES IST ZEILE 3
DIES IST ZEILE 4
DIES IST ZEILE 5
DIES I
```

*5T$$ gibt 5 Zeilen einschließlich des Teils der laufenden Zeile ab der Position des Pufferzeigers aus. In diesem Falle werden 5 Zeilen ausgegeben.

```
ST ZEILE 6
DIES IST ZEILE 7
DIES IST ZEILE 8
DIES IST ZEILE 9
DIES IST ZEILE 1Ø
```

Wenn nun das Kommando 6T wäre, würde die 6. Zeile nicht ausgegeben werden, weil keine 6. Zeile hinter der Zeile 6 innerhalb der Grenzen des Textpuffers vorhanden ist. Die 5 Zeilen würden wie im Beispiel links ausgegeben, dann würde das Kommando beendet.

*-5T5T$$ gibt alle 10 Zeilen des Textpuffers einschließlich der fünf Zeilen vor der Zeile mit dem Pufferzeiger, der Zeile mit dem Pufferzeiger selbst (vom Zeilenanfang bis zum Pufferzeiger), dem Rest der laufenden Zeile (vom Pufferzeiger bis zum Zeilenende) und der restlichen vier Zeilen aus.

```
DIES IST ZEILE 1
DIES IST ZEILE 2
DIES IST ZEILE 3
DIES IST ZEILE 4
DIES IST ZEILE 5
DIES IST ZEILE 6
DIES IST ZEILE 7
DIES IST ZEILE 8
DIES IST ZEILE 9
DIES IST ZEILE 1Ø
```

Angenommen, man möchte jetzt die Zeilen 3, 4, 5 und 6 löschen. Dann muß zuerst der Pufferzeiger entweder vor oder hinter die zu löschenden Zeilen positioniert werden. Man kann den Pufferzeiger vor die Zeilen stellen und ein Kommando K mit positivem Argument verwenden. Da sich der Pufferzeiger in Zeile 6 befindet, muß er vor die Zeile 3 gestellt werden; dann können die 4 Zeilen 3, 4, 5 und 6 gelöscht werden. Das Kommando sieht folgendermaßen aus:

*–3L4K$$ Das Kommando –3L verschiebt den Pufferzeiger zum Anfang der Zeile 3. Das Kommando 4K löscht die Zeilen 3, 4, 5 und 6.

Man könnte auch den Pufferzeiger zu der Zeile hinter der zu löschenden Zeile verschieben und ein Kommando K mit negativem Argument verwenden.

*L–4K$$ Das Kommando L verschiebt den Pufferzeiger zum Anfang der Zeile 7. Das Kommando –4K löscht die Zeilen 6, 5, 4 und 3.

Man kann sich durch Ausgabe des gesamten Inhalts des Textpuffers von der Löschung überzeugen.

*B1ØT$$ Der Pufferzeiger wird zum Anfang des Textpuffers verschoben. Das Kommando 10T versucht, 10 Textzeilen auszugeben, es sind aber nur 6 übriggeblieben. Das Kommando endet, wenn alle 6 Zeilen ausgegeben sind.

```
DIES IST ZEILE 1
DIES IST ZEILE 2
DIES IST ZEILE 7
DIES IST ZEILE 8
DIES IST ZEILE 9
DIES IST ZEILE 1Ø
```

5.4.6. Suchkommandos

Zwei Kommandos des Texteditors stehen zur Verfügung, um nach bestimmten Zeichenfolgen zu suchen. Das Kommando F veranlaßt den Editor, die erste Zeichenfolge zu suchen, die mit der im Befehl angegebenen Zeichenfolge übereinstimmt. Das Kommando S sucht ebenfalls eine angegebene Zeichenfolge und ersetzt sie durch eine zweite angegebene Zeichenfolge. Beide Kommandos beginnen mit der Suche beim Pufferzeiger und suchen vorwärts, bis die Suche erfolgreich ist oder bis das Ende des Textpuffers erreicht ist.

F – Zeichenfolge suchen

Das Kommando F sucht eine Zeichenfolge, die bis zu 16 Zeichen lang sein kann. Die Zeichen ESC, ALT MODE und CTRL-C werden nicht als Text angesehen, weil sie Steuerfunktionen ausüben und nicht in den Satz der Textzeichen eingeschlossen werden können. Das Kommando F hat folgende Syntax:

*Fx . . .x$$.

Dabei ist ›x . . .x‹ die im Kommando F anzugebende Zeichenfolge mit maximal 16 Zeichen, einschließlich der nichtabdruckbaren Zeichen.

Das Kommando F bewirkt, daß der Texteditor nach dem ersten Auftreten der Zeichenfolge sucht, die mit der im Kommando festgelegten Zeichenfolge übereinstimmt. Es müssen alle Zeichen übereinstimmen, einschließlich der abdruckbaren und nichtabdruckbaren Zeichen (wie Wagenrücklauf und Zeilenvorschub). Die Suche beginnt bei der augenblicklichen Position des Pufferzeigers und wird fortgesetzt, bis entweder eine Übereinstimmung gefunden oder das Ende des Textpuffers erreicht wird.

Sobald die Übereinstimmung gefunden ist, beendet der Texteditor das Kommando und läßt den Pufferzeiger unmittelbar hinter dem letzten Zeichen stehen. Dann wird ein Aufforderungszeichen für das nächste Kommando ausgegeben.

Wenn keine Übereinstimmung gefunden wird und das Ende des Textpuffers erreicht ist, gibt der Texteditor folgende Meldung aus:

CANNOT FIND ›XXX‹
* BREAK *

Dabei stellt ›XXX‹ die im Kommando angegebene Zeichenfolge dar. In diesem Falle wird der Pufferzeiger auf den Anfang des Textpuffers zurückgesetzt.

Wenn die festgelegte Zeichenfolge länger als 16 Zeichen ist, werden nur die ersten 16 Zeichen verwendet.

Wenn das Kommando F ein Teil einer Kommandofolge ist, beendet ein einzelnes ESC-Zeichen die Zeichenfolge, um zusätzliche Kommandos anfügen zu können. Falls keine weiteren Kommandos angehängt werden, kann die Zeichen- bzw. die Kommandofolge mit ESC ESC beendet werden. Die beiden ESC werden am Konsolgerät als $ $ ausgegeben.

Es ist wichtig, sich daran zu erinnern, daß die Zeichenfolge abgeschlossen werden muß, bevor weitere Kommandos angefügt werden können. Andernfalls werden die zusätzlichen Kommandos als Teil der Zeichenfolge behandelt, z. B. leitet das Kommando

*FDIVIDEOLT$$

eine Suche nach der Zeichenfolge DIVIDEOLT ein, statt nach der beabsichtigten Zeichenfolge DIVIDE. Das richtige Format mit einem einzelnen ESC nach dem Text lautet folgendermaßen:

*FDIVIDE$OLT$$

Es ist angebracht, sich durch Ausgeben der Zeile zu überzeugen, ob die Suche erfolgreich gewesen ist. Manchmal kann die gesuchte Zeichenfolge an mehreren unerwarteten Stellen vor der gesuchten Zeile auftreten. Wird z. B. nach der Marke DIV: gesucht und tritt die Zeichenfolge DIV mehrmals auf (z. B. in DIVIDE, DIV1, DIV2 und DIV3), dann kann die Angabe DIV als zu suchende Zeichenfolge zu falschen Resultaten führen. Es ist also eine eindeutige Zeichenkombination erforderlich. In diesem Falle wäre es besser gewesen, nach DIV: zu suchen und dann das Suchergebnis durch Ausgeben der Zeile zu überprüfen.

Wenn ein Wagenrücklauf in der zu suchenden Zeichenfolge eines Kommandos F auftritt, fügt der Texteditor automatisch dahinter einen Zeilenvorschub ein. Dadurch sucht das Kommando

*FEND.
NEXT$$

nach der Zeichenfolge END.›WR‹ ›ZV‹ NEXT

S – Zeichenfolge ersetzen

Das Kommando S sucht eine Zeichenfolge und ersetzt sie durch eine andere. Das Ersetzen geschieht nur, wenn die Suche erfolgreich gewesen ist. Das Kommando hat folgende Syntax:

Salttext$neutext$$

Dabei wird ›neutext‹ an Stelle von ›alttext‹ eingesetzt, falls dieser gefunden wird. Beide Zeichenfolgen müssen mit einem ESC-Zeichen (als $ ausgegeben) abgeschlossen werden. Die erste Zeichenfolge ›alttext‹ ist die zu suchende Zeichenfolge. Es werden nur die ersten 16 Zei-

chen verwendet. Die zweite Zeichenfolge ›neutext‹ ist die zu ersetzende Zeichenfolge. Sie kann eine beliebige Anzahl von Zeichen enthalten (ausgenommen die Zeichen ESC, ALT MODE und CTRL-C). Die zu ersetzende Zeichenfolge muß mit einem ESC- oder ALT MODE-Zeichen abgeschlossen werden.

Die Zeichenfolge ›alttext‹ wird nach erfolgreicher Suche gelöscht, falls ›neutext‹ im Kommando weggelassen wurde. Das Kommando S kann also dazu benutzt werden, ausgewählte Zeichenfolgen mit bis zu 16 Zeichen zu löschen.

Beim Suchen mit Hilfe des Kommandos S muß die zu suchende Zeichenfolge ganz genau mit der im Kommando S festgelegten Zeichenfolge übereinstimmen, einschließlich Groß- und Kleinschreibung, Zeichensteuerung, Wagenrücklauf- und Zeilenvorschubzeichen. Nach Abschluß des Kommandos S steht der Pufferzeiger hinter dem letzten ersetzten Zeichen.

5.4.7. Beispiele für die Anwendung der Suchkommandos F und S

1. Angenommen, man möchte die Zeichenfolge OFF in einer Programmzeile löschen, die folgendes enthält:

PARAM: CALL BACKOFF; BACK IS THE RETN
Um den Pufferzeiger vor die zu löschende Zeichenfolge zu positionieren, wird ein Kommando F verwendet. Eine Kommandofolge, die derartiges durchführt, lautet:

*BFBACK$3DØTT$$

Diese Kommandofolge verschiebt den Pufferzeiger zum Anfang des Pufferspeichers und beginnt dann die Suche nach der Zeichenfolge BACK. Beim ersten Auftreten dieser Zeichenfolge wird der Pufferzeiger hinter das K von BACK positioniert. Die nächsten drei Buchstaben (OFF) werden gelöscht und schließlich wird die ganze Zeile ausgegeben. Die zwei ESC-Zeichen (wiederholt als $$) beenden die Kommandofolge.

Nach der Kommandoausführung erscheint die Textzeile wie folgt:

PARAM: CALL BACK; BACK IS THE RETN

Zu beachten ist, daß man genau dasselbe Resultat erzielt hätte durch die Kommandofolge

*BSBACKOFF$BACK$ØTT$$

Hier wird die Zeichenfolge BACKOFF durch BACK ersetzt, was dieselbe Wirkung wie das Löschen von OFF hat.

Falls man unsicher ist, ob die gesuchte Zeichenfolge bereits in einer früheren Programmzeile auftritt, sucht man einfach die Zeichenfolge mit dem Kommando F und gibt die gefundene Zeile aus, bevor man ersetzt oder löscht, um sicherzugehen, daß man die gewünschte Textzeile vor sich hat.

2. Angenommen, man möchte eine Zeichenfolge suchen und löschen. Dies kann durch Kombination der Kommandos F und D geschehen.
Z.B. ist die Marke PARAM: in der folgenden Zeile zu löschen:

PARAM: CALL SUB1

Dazu wäre nachstehende Kommandofolge nötig:

*BFPARAM:$–6D$$

Die Zeichenfolge PARAM wird mit dem Kommando F gesucht. Jetzt steht der Pufferzeiger hinter dem Doppelpunkt. Anschließend werden die davorstehenden 6 Zeichen gelöscht. Die nachstehende Kommandofolge mit dem Kommando S würde dasselbe Ergebnis bringen:

*BSPARAM:$$

3. Der zu korrigierende Fehler bestehe aus einem falsch geschriebenen Wort. Die nachstehende Kommandofolge wird verwendet, um das falsche Wort zu suchen und durch das richtige zu ersetzen. Nach der Korrektur wird eine Ausgabe verlangt, um die Operation überprüfen zu können.

*SINITAIL$INITIAL$ØTT$$

Der Texteditor führt die Ersetzung aus und gibt die korrigierte Zeile aus. Am Schluß der Operation ist der Pufferzeiger zwischen dem L von INITIAL und dem anschließenden Leerzeichen positioniert. Die korrigierte Zeile wird wie folgt ausgegeben:

LXI H,Ø ; INITIAL VALUE FOR REMAINDER

5.4.8. Ein- und Ausgabekommandos

Die nächsten fünf Kommandos bewirken die Eingabe von und die Ausgabe auf Dateien, für die eine Textaufbereitung ausgeführt wird.

A – Text am Pufferende anhängen

Das Kommando A wird verwendet, um Text von der Eingabedatei in den Textpuffer zu übertragen. Das Kommando A hängt den Text hinten an den bereits im Textpuffer stehenden Text an. Das Kommando A ist üblicherweise das erste verwendete Kommando, wenn man eine bereits vorhandene Datei aufbereiten will, da man ja den Text von der Eingabedatei in den Textpuffer bringen muß, um ihn aufzubereiten. Das Kommando A bewirkt das Einlesen von Text, bis eine der folgenden Bedingungen erfüllt ist:
Es sind schon fünfzig Textzeilen in den Textpuffer eingelesen worden.
Der Arbeitsspeicher ist voll. Der Arbeitsspeicher wird nur bis zu den 240 Bytes unter dem oberen Ende des RAM-Speichers gefüllt.
Ein Seitenvorschubzeichen (CTRL-L) ist gelesen worden. Das Seitenvorschubzeichen wird in den Textpuffer gebracht.
Das Dateiende ist erreicht.
Ein Dateiendezeichen (CTR-Z) ist gelesen worden. Das Dateiendezeichen wird nicht in den Textpuffer gebracht.
Wenn man mehr als 50 Textzeilen einzugeben hat, muß man das Kommando A so oft wie nötig wiederholen. Jedes Kommando A wird beendet, wenn eine der obigen Bedingungen erfüllt ist.

Beispiel
Um 150 Textzeilen von der vorhandenen Diskettendatei FLOW.SRC auf Laufwerk 1 in den SME-Arbeitsspeicher einzulesen, ist folgendes zu tun:

–EDIT :F1:FLOW.SRC
ISIS-II TEXT EDITOR, Vn.n
*M
24.791 – CHARACTERS AVAILABLE IN WORK SPACE
*AAA$$

Da viel Arbeitsspeicher zur Verfügung steht (in diesem Falle 24791 Bytes) und keine Seitenvorschübe in der Datei vorhanden sind, bewirkt jedes Kommando A, daß 50 Textzeilen eingegeben werden (es sei denn, daß ein Seitenvorschub oder das Dateiende auftritt). Insgesamt werden 150 Textzeilen an den Textpuffer angehängt.

W – Text in Ausgabedatei schreiben

Das Kommando W wird verwendet, um eine festgelegte Anzahl von Zeilen vom Textpuffer auf die im Kommando EDIT festgelegte Ausgabedatei zu schreiben. Der Text wird immer ab dem Anfang des Textpuffers genommen, ohne Rücksicht auf die augenblickliche Position des Pufferzeigers. Sobald eine Textzeile auf die Ausgabedatei geschrieben ist, wird sie im Textpuffer gelöscht. Der restliche Text wird zum Anfang des Textpuffers verschoben. Das Kommando hat folgende Syntax:

*nW$$

Dabei ist ›n‹ eine ganze Dezimalzahl von −65535 bis +65534. Negative Argumentwerte werden als positive Werte betrachtet. Der Wert n = Ø bewirkt, daß keine Zeilen auf die Ausgabedatei geschrieben werden.

E – Beenden des Editors nach Ausgabe des Puffers an die Ausgabedatei

Das Kommando E wird verwendet, um sowohl den ganzen Inhalt des Textpuffers als auch den Rest der Eingabedatei auf die Ausgabedatei zu schreiben und dann nach ISIS II zurückzukehren. Der Text wird auf die im Kommando EDIT festgelegte Ausgabedatei geschrieben. Das Kommando hat folgende Syntax:

*E$$

Der Textpuffer ist nach dem Kommando E völlig leer und die Steuerung geht an ISIS II zurück.

Q – Sitzung beenden

Das Kommando Q kann eingegeben werden, um den Verlauf einer Aufbereitungssitzung ohne irgendwelche Ausgabe zu beenden. Das Kommando hat folgende Syntax:

*Q$$

Die Ausgabedatei enthält alles, was vor dem Kommando Q darauf geschrieben worden ist. In gewissen Fällen bleibt die Sonderdatei EDIT.TMP auf der Diskette stehen.

Zu beachten ist, daß der Dateiname EDIT.TMP am besten für den Gebrauch durch den Texteditor reserviert wird, weil jede Datei mit dem Namen EDIT.TMP, die auf derselben Diskette wie die Eingabedatei steht, vom Texteditor überschrieben wird.

M – Speicherplatz

Das Kommando M errechnet die Größe des Speicherplatzes, der zur Verwendung durch den Texteditor übrigbleibt, und gibt diesen Wert auf dem Konsolgerät aus. Das Kommando hat folgende Syntax:

*M$$

Das System antwortet mit einer Meldung:
n – CHARACTER(S) AVAILABLE IN WORKSPACE

Dabei ist ›n‹ eine ganze Dezimalzahl.

Hinweis:
Wenn man eine große Datei aufbereitet und dafür zu wenig freien Arbeitsspeicher hat, kann man den ersten Teil dieser Datei mit Hilfe des Kommandos W auf die Ausgabedatei schreiben. Dadurch gewinnt man zusätzlichen Speicherplatz für den Textpuffer des Texteditors.

5.5. Kommandowiederholungen

Ein Kommando oder eine Kommandofolge kann beliebig oft wiederholt werden, indem man sie in spitze Klammern (‹....›) einschließt und davor eine Zahl schreibt, die festlegt, wie oft die Wiederholung durchgeführt werden soll. Das Kommando hat folgende Syntax:

n‹Kommando oder Kommandofolge›$$

Dabei ist n die Anzahl, wie oft das in spitzen Klammern stehende Kommando ausgeführt werden soll.

Hat man zum Beispiel beim Schreiben eines Programms die Marke DIVID zehnmal in der Primärprogrammdatei verwendet und möchte die Marke auf DIV kürzen, kann man ein wiederholtes S-Kommando wie folgt benutzen:

*B1Ø‹SDIVIDDIV›$$

Das B-Kommando verschiebt den Pufferzeiger an den Anfang des Textpuffers. Das S-Kommando wird zehnmal wiederholt; es sucht nach der Zeichenfolge DIVID und ersetzt sie, falls gefunden, jedesmal durch die neue Zeichenfolge DIV.

Kommandowiederholungen können bis achtmal geschachtelt werden. Jeder Versuch, mehr als achtmal zu schachteln, wird abgewiesen. Auf dem Bediengerät wird die Fehlermeldung ausgegeben:

ITERATION STACK FAULT

Zum Beispiel sei angenommen, daß der Textpuffer die Sinuswerte für die Winkel 0 bis 90 Grad mit der Schrittweite 1 Grad enthält. Die Sinuswerte seien mit einem Wert je Zeile angeordnet und mit 15 Ziffern Genauigkeit wie folgt angegeben:

```
Ø.ØØØØØØØØØØØØØØØ
Ø.Ø17452406437284
Ø.Ø34899496702501
Ø.Ø52335956242944
    ''
    ''
Ø.999390827Ø19096
Ø.999847695156391
1.ØØØØØØØØØØØØØØØ
```

Zur Verbesserung der Lesbarkeit ist es zweckmäßig, durch Einfügen von Leerzeichen jede Zahl in Fünfergruppen aufzuteilen, so daß jede wie folgt aussieht:

```
Ø.ØØØØØ ØØØØØ ØØØØØ
Ø.Ø1745 24Ø64 37284
Ø.Ø3489 94967 Ø25Ø1
    ''
    ''
```

Ø.99939 Ø827Ø 19Ø96
Ø.99984 76951 56391
1.ØØØØØ ØØØØØ ØØØØØ

Dies kann durch Schachtelung von zwei wiederholten Kommandos erreicht werden. Die eine Stufe überläuft die ersten zwei Zeichen und fügt dann ein Leerzeichen nach jedem fünften Zeichen ein. Die andere Stufe durchläuft alle 91 Zeilen in der Datei. Das Kommando lautet:

*B91‹2C2‹5CI $›L›$$

5.6. Fehlermeldungen des Texteditors

Der Texteditor gibt Meldungen auf dem Bediengerät des Systems aus, um den Anwender über bestimmte Zustandsbedingungen zu unterrichten. Es gibt folgende drei Fehlermeldungen:

n ILLEGAL IN THIS CONTEXT

Das n stellt das unzulässige alphanumerische Zeichen dar, das falsch eingegeben worden ist.

CANNOT FIND xxx

Die Zeichen xxx stellen die Zeichenfolge dar, die der Texteditor während eines Kommandos F oder S nicht hat finden können. Diese Meldung gibt außerdem *BREAK* aus, um anzuzeigen, daß das Kommando abgebrochen worden ist.

ITERATION STACK FAULT

Diese Meldung ist für den Anwender ein Hinweis, daß wiederholte Kommandos mehr als achtmal geschachtelt worden sind. Das Kommando wird abgebrochen.

5.7. Aufbereiten eines Musterprogramms

Im folgenden Musterprogramm sind einige Korrekturen und Änderungen, die handschriftlich eingetragen wurden. Dabei werden die zur Richtigstellung des Programms erforderlichen Aufbereitungskommandos gezeigt, zusammen mit ausführlichen Kommentaren, und schließlich das berichtigte Programm selbst.

```
;
; REENTRANT DIVIDE ROUTINE
;
DIV:
 MOV     A,D          ; NEGATE THE DIVISOR
 CMA
 MOV     D,A
 MOV     A,E
 CMA
 MOV     E,A
 INX     D
 LXI     H,0          ; INITIAL VALUE FOR REMAINDER
 MVI     A,17         ; INITIALIZE LOOP COUNTER
DIVO:
 PUSH    H            ; SAVE REMAINDER
 DAD     D            ; SUBTRACT DIVISOR (ADD NEGATIVE)
 JNC     DIV1         ; UNDER FLOW, RESTORE HL
 XTHL
DIV1:
 POP     H
 PUSH    PSW          ; SAVE LOOP COUNTER
 MOV     A,C          ; 4 REGISTER LEFT SHIFT
 RAL                  ; WITH CARRY
 MOV     C,A          ; CY -> C -> B -> L -> H
 MOV     A,B
 RAL
 MOV     B,A
 MOV     A,L
 RAL
 MOV     L,A
 MOV     A,H
 RAL
 MOV     H,A
 POP     PSW          ; RESTORE LOOP COUNTER
 DCR     A            ; DECREMENT IT
 JNZ     DIVO         ; KEEP LOOPING
DIV2:
 ORA     A            ; POST-DIVIDE CLEAN UP
 MOV     A,H          ; SHIFT REMAINDER RIGHT AND
 MOV     D,A          ;   RETURN IN DE
 MOV     A,L
 MOV     E,A
 RET
 END
```

BC = DIVIDEND/QUOTIENT
HL = TEMPORARY
DE = DIVISOR/REMAINDER

RAR
RAR

Die drei Kommentare können mit nachstehender Kommandofolge in den Text eingefügt werden:

```
*3LI; BC = DIVIDEND/QUOTIENT
; HL = TEMPORARY
; DE = DIVISOR/REMAINDER
;
$$
```

Der als nächster zu korrigierende Fehler besteht aus einem falschgeschriebenen Wort. Das folgende Kommando dient dazu, das falsche Wort zu suchen und es durch das richtige zu ersetzen. Sobald es verbessert ist, soll die Zeile ausgegeben werden, um den Vorgang überprüfen zu können.

```
*SINITAIL$INITIAL$ØTT$$
```

Der Texteditor antwortet dadurch, daß er das Ersetzen ausführt und die korrigierte Zeile ausgibt. Nach Operationsende steht der Pufferzeiger an der Stelle zwischen dem L von INITIAL und dem anschließenden Leerzeichen. Die korrigierte Zeile wird wie folgt ausgegeben:

```
LXI   H,Ø   ; INITIAL VALUE FOR REMAINDER
```

Die nächste Aufgabe ist es, alle Vorkommen der Marke DIVØ zu finden und sie durch die Zeichenfolge DVØ zu ersetzen. Die Zahl 1Ø wird gewählt, weil bekannt ist, daß die Marke DIVØ weniger als zehnmal vorkommt. Bei jedem Vorkommen, für das sich eine Übereinstimmung ergibt, wird die Marke ersetzt und die Zeile ausgegeben. Die Kommandofolge sieht folgendermaßen aus:

```
*1Ø‹SDIVØ$DVØ$ØTT›$$
```

Der Texteditor antwortet folgendermaßen:

```
DVØ:
JNZ     DVØ     ;KEEP LOOPING
CANNOT FIND DIVØ
*BREAK*
```

In ähnlicher Weise kann man überall DIV1 durch DV1 ersetzen.

Als nächstes ist das Register PSW im Primärprogramm falsch mit PCW bezeichnet worden. Es wird ein Suchvorgang nach der Zeichenfolge PCW durchgeführt. Jedesmal, wenn diese gefunden wird, wird sie durch die Zeichenfolge PSW ersetzt. Die Kommandofolge lautet:

```
*B1Ø‹SPCW$PSW$ØTT›$$
```

Der Vorgang ist dem oben beschriebenen ähnlich. Der Texteditor gibt dann folgendes aus:

```
PUSH    PSW     ; SAVE LOOP COUNTER
POP     PSW     ; RESTORE LOOP COUNTER
CANNOT FIND PCW
*BREAK*
```

Es sind jetzt alle Vorkommen von DIVØ, DIV1 und PCW gefunden und verbessert worden. Der Pufferzeiger steht am Ende des Textpuffers. Die nächste Aufgabe lautet, die Marke DIV2 zu löschen (die in keiner der Anweisungen aufgerufen wird) und die Kommentare zum linken Rand hin zu verschieben. Außerdem sind Kommentarzeilen vor und hinter dem Text

POST DIVIDE CLEAN UP zu erzeugen. Zusätzlich soll zwischen die Wörter POST und DIVIDE ein Bindestrich geschrieben werden. Die Kommandofolge sieht folgendermaßen aus:

```
*BFKEEP LOOPING$LI;
$9DFPOST$DI–$LI; SHIFT REMAINDER RIGHT AND RETURN IN DE
;
$–4T$$
```

Das erste Kommando B stellt den Pufferzeiger auf den Anfang des Textpuffers zurück. Da die Zeichenfolge KEEP LOOPING in der vorangegangenen Zeile steht, ist es am bequemsten, das F-Kommando zum Finden der gewünschten Zeile zu benutzen.

Das nächste Kommando L verschiebt den Pufferzeiger zur nächsten Zeile, wo das Semikolon und ein Wagenrücklauf/Zeilenvorschub eingefügt werden. Dann werden 9 Zeichen gelöscht, wobei die Marke und 4 Tabulatorzeichen entfernt werden, was den Kommentar zum linken Rand hin verschiebt. Jetzt wird die Zeichenfolge POST gesucht und schnell gefunden, weil sie unmittelbar hinter dem Pufferzeiger steht. Dies positioniert den Pufferzeiger hinter das T von POST. Ein Zeichen (das Leerzeichen) wird gelöscht und durch das Zeichen Bindestrich (-) ersetzt. Das L-Kommando verschiebt den Pufferzeiger zur nächsten Zeile, wo eine weitere Kommentarzeile, ein Wagenrücklauf/Zeilenvorschub eingefügt werden. Das Kommando -4T wird verwendet, um den ganzen Vorgang zu überprüfen. Der Texteditor gibt folgendes aus

```
;
; POST-DIVIDE CLEAN UP
; SHIFT REMAINDER RIGHT AND RETURN IN DE
;
```

Die nächste Aufgabe lautet, zwei RAR-Kommandos einzufügen. Das Kommando, die Zeile zu finden und das erste RAR einzufügen, sieht so aus:

```
*2‹FMOV$›ØLI     RAR
$-2T$$
```

Von der augenblicklichen Stellung des Pufferzeigers aus, anschließend an die letzte Operation, werden zwei Suchvorgänge nach der Zeichenfolge MOV durchgeführt. Zu beachten ist, daß durch dieses Kommando unvorhersehbare Ergebnisse verursacht würden, wenn die Stellung des Pufferzeigers nicht bekannt wäre. Da man jedoch noch die letzte Kommandofolge in Erinnerung hat und sich ausrechnen kann, daß der Pufferzeiger an der Stelle zwischen dem Zeilenvorzeichen der Zeile mit dem Semikolon und dem Tabulatorzeichen vor dem O von ORA stehengeblieben ist, ist die Position der nächsten zwei MOV-Zeichenfolgen bekannt. Nachdem das zweite MOV ermittelt worden ist, steht der Pufferzeiger hinter dem V des zweiten MOV. Der Pufferzeiger wird dann zum Anfang der laufenden Zeile verschoben und es wird eine Zeile eingefügt, die aus ‹tabulatorzeichen›, RAR und ‹Wagenrücklauf, Zeilenvorschub› besteht.

Das Kommando -2T wird verwendet, um sich durch Ausgabe der vorangegangenen und der laufenden Zeile von der Einfügung zu überzeugen. Der Texteditor gibt folgendes aus:

```
MOV     A,H
RAR
```

Durch die letzte Kommandofolge wird der Pufferzeiger um drei Zeilen vorwärts verschoben und dann die Zeichenfolge ‹tabulatorzeichen›, RAR und ‹Wagenrücklauf, Zeilenvorschub› ein-

gefügt. Die laufende und die vorangehende Zeile werden ausgegeben. Die Kommandofolge sieht folgendermaßen aus:

```
*3LI     RAR
$-2T$$
```

Der Texteditor gibt daraufhin folgendes aus:

```
MOV      A,L
RAR
```

Das Programm ist nun vollständig. Eine korrigierte Fassung des Textes folgt.

```
;
; REENTRANT DIVIDE ROUTINE
;
; BC = DIVIDEND/QUOTIENT
; HL = TEMPORARY
; DE = DIVISOR/REMAINDER
;
DIV:
 MOV     A,D          ; NEGATE THE DIVISOR
 CMA
 MOV     D,A
 MOV     A,E
 CMA
 MOV     E,A
 INX     D
 LXI     H,0          ; INITIAL VALUE FOR REMAINDER
 MVI     A,17         ; INITIALIZE LOOP COUNTER
DVO:
 PUSH    H            ; SAVE REMAINDER
 DAD     D            ; SUBTRACT DIVISOR (ADD NEGATIVE)
 JNC     DV1          ; UNDER FLOW, RESTORE HL
 XTHL
DV1:
 POP     H
 PUSH    PSW          ; SAVE LOOP COUNTER
 MOV     A,C          ; 4 REGISTER LEFT SHIFT
 RAL                  ; WITH CARRY
 MOV     C,A          ; CY -> C -> B -> L -> H
 MOV     A,B
 RAL
 MOV     B,A
 MOV     A,L
 RAL
 MOV     L,A
 MOV     A,H
 RAL
 MOV     H,A
 POP     PSW          ; RESTORE LOOP COUNTER
 DCR     A            ; DECREMENT IT
 JNZ     DVO          ; KEEP LOOPING
;
; POST-DIVIDE CLEAN UP
; SHIFT REMAINDER RIGHT AND RETURN IN DE
;
 ORA     A
 MOV     A,H
 RAR
 MOV     D,A
 MOV     A,L
 RAR
 MOV     E,A
 RET
 END
```

5.8. Zusammenstellung der Kommandos des Texteditors

Die folgende Aufstellung ist eine alphabetische Liste von Kommandos des Texteditors:

Kommando	Beschreibung
A	Text an das Ende des Textpuffers anhängen
B	Pufferzeiger an den Anfang des Textpuffers verschieben.
± nC	Pufferzeiger um n Zeichen vorwärts (+) oder rückwärts (−) verschieben.
± nD	n Zeichen vor (−) oder hinter (+) dem Pufferzeiger löschen.
E	Inhalt des Textpuffers und Rest der Eingabedatei auf die Ausgabedatei schreiben und nach ISIS II zurückkehren.
Ftext$	Die erste Zeichenfolge ›text‹ hinter dem Pufferzeiger suchen und dann den Pufferzeiger dahinter positionieren.
Itext$	Text vor der aktuellen Position des Pufferzeigers in den Textpuffer einfügen.
± nK	n Zeilen vor (−) oder hinter (+) dem Pufferzeiger löschen.
± nL	Pufferzeiger n Zeilen vorwärts (+) oder rückwärts (−) verschieben.
M	Größe des noch zur Verfügung stehenden Arbeitsspeicherplatzes errechnen und ausgeben.
Q	Sitzung beenden, ohne den Textpuffer auf die Ausgabedatei zu schreiben.
Salttext$neutext$	Die erste Zeichenfolge ›alttext‹ vom aktuellen Pufferzeiger an suchen und durch ›neutext‹ ersetzen.
± nT	n Zeilen vor (−) oder hinter (+) dem Pufferzeiger auf dem Konsolgerät ausgeben.
nW	n Zeilen ab Textpufferanfang auf die Ausgabedatei schreiben und die geschriebenen Zeilen im Textpuffer löschen.
Z	Pufferzeiger zum Ende des Textpuffers verschieben.

6. Einführung in das Binden und Entrelativieren

(d. h. umwandeln von relativ-adressierten Objektmoduln in absolut-adressierte)

Wesentlich erhöhter Bedienungskomfort im Betriebssystem ISIS II ergibt sich aus drei neu hinzugefügten Programmen, LIB, LINK und LOCATE; im Zusammenhang damit hat jetzt der Objekt-Code ein anderes Format, es gibt einen neuen **residenten** Compiler, PL/M 80 und einen neuen residenten Assembler, ASM80. Der im System ISIS II von PL/M 80 oder ASM80 erzeugte Objekt-Code ist relativ adressiert. In ISIS I erzeugter Code muß deshalb umgewandelt werden, um im ISIS II weiter verwendet werden zu können (siehe HEXOBJ- und BINOBJ-Kommandos in Kapitel 4).

Das LOCATE-Programm wandelt relativ-adressierten Code in absolut-adressierten um. Dieser kann dann von ISIS II geladen werden, um das Programm ablaufen zu lassen und zu testen oder von UPM, um PROMs zu programmieren oder einen Streifen für die ROM-Maskierung zu erstellen oder auch vom ICE80, um ein System mit dem Emulations- und Test-Adapter zu testen.

Das LINK-Programm fügt Dateien, die die Einzelteile eines Programms im Objekt-Code enthalten, zu einer Datei zusammen, die einen einzigen Objektmodul enthält. Dabei werden die relativen Adressen entsprechend angepaßt.

Das LIB-Programm erzeugt und wartet Bibliotheken aus Objektmoduln, die als Bausteine zum Erzeugen neuer Programme mit Hilfe von LINK verwendet werden können.

Die zahlreichen Gründe für das Relativieren eines Programms und für die modulare Programmierung werden in den folgenden zwei Hauptpunkten, Speicherzuteilung und modulare Programmentwicklung, erörtert.

6.1. Speicherstruktur

Der Hauptspeicher eines Mikroprozessoranwendersystems ist im allgemeinen nicht einheitlich zusammengesetzt. Der Speicher ist üblicherweise für die spezifische Anwendung maßgeschneidert, d. h. mit RAM-Speichern für variable Daten und mit ROM- oder PROM-Speichern zum Speichern von Programmbefehlen und Konstanten ausgestattet. Die Speicherbausteine können auch so eingebaut sein, daß manchen Adressen gar kein entsprechender physikalischer Speicher entspricht.

Dieser uneinheitliche Speicheraufbau (zusammen mit später erklärten Programmierüberlegungen) verlangt, daß das Programm so zugeschnitten wird, daß die absoluten Adressen der aktuellen ROM- und RAM-Speicherstellen festgelegt werden, die für Befehle und Daten benutzt werden.

In einem frühen Entwicklungsstadium genügt es, beim Programmtest im SME darauf zu achten, daß durch das Adressieren des Programms keine residenten ISIS-II-Routinen überschrieben werden. Der Speicher im SME ist einfach eine Folge von RAM-Speicherstellen von Ø bis 32 K, 48 K oder 64 K. Wenn jedoch die Programmentwicklung fortschreitet und das Programm unter dem ICE-80-Programm in einer Kombination von Anwendungs- und SME-Speicher oder ganz im Speicher des Anwendungsprototyps läuft, muß man genau auf die Adressierung achten. Der Befehlscode könnte etwa in einem PROM-Speicher stehen und bei der Speicherstelle Ø starten, variable Daten könnten in der ersten Speicherstelle eines RAM-Speicherblocks beginnen.

Das Programm, das zur Kompatibilität mit der residenten ISIS-II-Software im SME als Basisadresse 4000 H gehabt hat und dessen variable Daten unmittelbar hinter dem Befehlscode gestanden haben, benötigt neue Adressen, um den Speicheranforderungen in der endgültigen Anwendung zu entsprechen. In einem System, das nur mit absolutem Code umgeht, müßte man das Primärprogramm ändern, um die neuen Adressen festzulegen und erneut übersetzen, um einen Objekt-Code mit richtigen Adressen zu erhalten. Im ISIS II erzeugt man einfach ein neues absolutes Speicherabbild aus dem relativ-adressierten Objektmodul, indem man das LOCATE-Programm anwendet. Dieses Programm weist neue Adressen zu, um die Befehle im ROM- oder PROM-Speicher, die variablen Daten im RAM-Speicher und den Stack in einem anderen RAM-Bereich unterzubringen.

6.1.1. Programmsegmente

Mit LOCATE können dem Befehlsteil, den Daten und dem Stack gesonderte Speicherbereiche zugeordnet werden, und zwar auf Grund dessen, daß schon die Sprachenübersetzer (ASM80 und PL/M 80) einen relativierbaren Objektmodul (das übersetzte Programm) in Segmente aufteilen. Die Segmente sind:

CODE	(Befehlsfolge)
DATA	(Daten)
STACK	(Stack)
MEMORY	(freier Speicher)

Jedes Segment hat die relative Anfangsadresse Null. Für jedes Segment kann eine Basisadresse festgelegt werden, wenn ihm sein Platz zugewiesen wird. Die Basisadresse ist die aktuelle Adresse der ersten Speicherstelle, die das Segment belegen wird. LOCATE addiert die Basisadresse zu jeder relativen Adresse und paßt die Aufrufadressen entsprechend an.

CODE-Segment

Das Codesegment ist der für ROM bestimmte Programmteil. Er enthält Maschinenbefehle und Programmkonstanten. Dieses Segment ist selbstverständlich auch im RAM ablauffähig. In der Assemblersprache sind es die Anweisungen, die auf eine CSEG-Anweisung folgen.

DATA-Segment

Das Datensegment ist der Programmteil, für den ein RAM-Speicher erforderlich ist. Dieses Segment enthält variable Daten und Speicher für Ein-/Ausgabepuffer. In der Assemblersprache enthält es die Anweisungen, die auf eine DSEG-Anweisung folgen.

STACK-Segment

Das Stacksegment muß im RAM liegen. Üblicherweise besitzt nur ein Modul, das Hauptprogramm, eine Bezugnahme auf den Stack. Die Segmentlänge wird bei PL/M-Programmen durch den Übersetzer, bei Assemblerprogrammen durch die STKLN-Anweisung bestimmt. Man kann die Länge auch erst dann festlegen, wenn das Programm durch LOCATE absolut adressiert wird. Mit Hilfe von STACK (einem reservierten Wort in der Assemblersprache) und STACKPTR (einem Schlüsselwort in PL/M) wird auf den Stack bezuggenommen.

MEMORY-Segment

Das Segment des freien Speichers ist der RAM-Speicher im SME, der nicht durch Code-, Daten- und Stacksegmente belegt ist. Aufrufe an das Speichersegment werden mit Hilfe von

MEMORY (einem reservierten Wort in der Assemblersprache bzw. einem Schlüsselwort in PL/M) ausgeführt. Obwohl die Sprachenübersetzer für jeden relativ-adressierten Objektmodul ein Speichersegment erzeugen, ist seine Länge erst dann bekannt, wenn LOCATE einen absoluten Modul erzeugt hat. LOCATE errechnet die Länge des verfügbaren RAM-Speichers, indem es die Monitorroutine MEMCK und die Basisadresse des freien Speichersegments verwendet.

Absoluter Code

Außer den Code-, Daten-, Stack- und freien Speichersegmenten kann ein relativ-adressierter Objektmodul Code oder Daten mit bereits zugewiesenen absoluten Adressen enthalten. Es gibt drei Arten, wie dies geschehen kann. Die ASEG-Anweisung in Assemblersprache bewirkt, daß die darauffolgenden Anweisungen (bis zum Auftreten einer Anweisung CSEG oder DSEG) absolute Adressen erhalten. Absolute Moduln, die von LOCATE erzeugt sind, können mit relativ-adressierten Moduln gebunden werden. PL/M-Variable, die mit dem AT-Attribut deklariert sind, erzeugen absoluten Code.

Es ist möglich, ein in sich geschlossenes Programm in Assemblersprache unter Verwendung der ASEG-Anweisung zu schreiben. Dies ergibt einen absoluten Modul, der sofort nach der Assemblierung ausgeführt werden kann, weil die Assemblerausgabe ein Speicherabbild ist. Dieses Vorgehen ist nur in solchen Fällen möglich, die keine Veränderung der zugehörigen Speicheradressen erfordern.

6.2. Modulare Programmentwicklung

Die meisten Programme sind zu lang oder zu komplex, um sie in einem Stück zu programmieren (monolithische Vorgehensweise). Man kann die Arbeit vereinfachen, indem man zuerst eine Hauptroutine erstellt, die einzelne Funktionen in Unterprogrammen aufruft. Die Parameterübergabe an diese Unterprogramme kann während des Schreibens der Hauptroutine bestimmt werden. Die wirkliche Programmierung der Unterprogramme kann jedoch bis später aufgeschoben werden. Das endgültige komplette Programm wird durch das LINK-Programm aus der Hauptroutine und den Unterprogrammen zusammengebaut. Die Vorteile dieses Vorgehens beim Programmaufbau werden in den folgenden Paragraphen zusammengefaßt.

6.2.1. Schnelle Programmentwicklung

Ein Programm kann schneller entwickelt werden, wenn die modulare Vorgehensweise verwendet wird, weil kleine Moduln leichter zu durchschauen und einfacher zu programmieren sind. Das Aufteilen eines Programms in Funktionsmoduln erleichtert es, Einzelteile den Mitarbeitern eines Teams zu übergeben. Bei speziellen gut definierten Aufgaben kann sich ein Entwickler auf das Programmieren und Testen eines speziellen Moduls konzentrieren. Er kann die vom Unterprogramm benötigte Eingabe simulieren und sie durch Überprüfen der Ausgabe bestätigen. Wenn die Moduln ausgetestet sind, können sie unter Verwendung des LIB-Programms in eine Bibliotheksdatei gebracht werden. Wenn alle Moduln fertiggestellt sind, wird das LINK-Programm verwendet, um die Moduln zu einem kompletten Programm zusammenzufügen, dem dann durch LOCATE die absoluten Speicheradressen zugewiesen werden.

6.2.2. Verschiedene Programmiersprachen

Die Moduln, die das endgültige Programm bilden, brauchen nicht aus derselben Sprache übersetzt worden zu sein. PL/M oder Assemblersprache kann verwendet werden, je nachdem,

welche davon am besten für die Lösung der Aufgabe geeignet ist. Relativ-adressierte Moduln, entweder vom PL/M-Compiler oder vom Assembler erzeugt, können Eingabe für das LINK-Programm sein, um ein Programm zu erstellen.

6.2.3 Mehrfachbenutzbare Unterprogramme

Wenn allgemeine Unterprogramme getestet und in eine Bibliothek eingebracht worden sind, können sie von anderen Programmen mitbenutzt werden. Dies bedeutet, daß ein Teil zukünftiger Programmierarbeit bereits getan ist. Da die Unterprogramme relative Adressen haben, können sie mit unterschiedlichen Programmen, die jedesmal unterschiedliche Adreßanforderungen haben, kombiniert werden.

6.2.4. Leichte Fehlerbereinigung und Programmänderung

Es ist leicht, die Stelle eines Fehlers einzukreisen, wenn ein Programm in Moduln aufgeteilt ist. Hat man den Modul ermittelt, der den Fehler enthält, kann man sich darauf konzentrieren, diesen Modul zu testen. Wenn ein Programm geändert werden muß, bedeutet dies meist nur, daß ein oder zwei Moduln geändert werden oder neue Moduln hinzugefügt werden müssen. Sind die neuen oder geänderten Moduln übersetzt und ausgetestet, kann ein neues absolutes Speicherabbild einfach unter Verwendung von LINK und LOCATE erzeugt werden, ohne daß das gesamte Primärprogramm neu übersetzt werden muß.

6.3. Technik des Relativierens und Bindens

Das Programm LINK fügt Moduln zusammen und paßt dabei die relativen Adressen an. Das Programm LOCATE wandelt relative Adressen in absolute Adressen um, und zwar aufgrund von Informationen, die von den Übersetzern ASM80 und PL/M 80 in die Objektmoduln eingebracht worden sind. Die folgenden Informationsarten im Objektmodul werden von LINK und LOCATE verarbeitet:

1. relative Adressen der Befehle und Daten
2. eine Liste von Adressen in Befehlen oder Daten, die sich auf eine Speicherstelle im selben Segment beziehen (Intrasegmentaufrufe)
3. eine Liste von Adressen in Befehlen oder Daten, die sich auf Speicherstellen in anderen Segmenten desselben Moduls beziehen (Intersegmentaufrufe)
4. eine Liste von Adressen in Befehlen oder Daten, die sich auf Speicherstellen beziehen, die nicht im selben Modul enthalten sind (Externaufrufe)
5. eine Liste der Symbole im Modul, die im Primärprogramm als PUBLIC oder EXTERNAL deklariert worden sind.

Man sollte die Bedeutung der Externaufrufe und der Deklaration von Symbolen als PUBLIC oder EXTERNAL kennen, um LINK und LOCATE erfolgreich anzuwenden. Das Verständnis der anderen Themen in diesem Kapitel ist nicht ganz so wichtig. Sie werden hier nur gebracht, um ein geschlossenes Bild über die Methoden der relativen Adressierung und des Bindens zu vermitteln.

6.3.1. Relative Adressen

Die relativen Adressen der Befehle und Daten in den Code- und Datensegmenten werden von ASM80 und PL/M 80 zugewiesen, wenn ein Primärprogrammodul übersetzt wird. Die Adressen werden von einem Adreßpegel errechnet, der zu Anfang bei Null startet und um die Anzahl

der Bytes in jedem Befehl erhöht wird. Die Adressen sind ›relativ‹ zum Anfang des Segments. LINK fügt die Eingabemoduln zu einem Ausgabemodul zusammen, indem es alle Codesegmente zu einem Codesegment und alle Datensegmente zu einem Datensegment zusammenfügt. Die relativen Adressen des ersten Segments bleiben ungeändert, LINK ändert aber die relativen Adressen des darauffolgenden Segments derart, daß sie sich auf den Anfang des neuen Segments beziehen. Allgemein heißt dies, daß die Länge des ersten Segments zu den relativen Adressen des zweiten addiert wird, die Gesamtlänge der ersten zwei Segmente zu den relativen Adressen des dritten Segments addiert wird, und so fort.

LOCATE erzeugt einen absoluten Modul aus einem relativ-adressierten, indem es die Basisadresse jedes Segments zu allen relativen Adressen in diesem Segment addiert, um absolute Adressen zu erhalten. Die Basisadresse jedes Segments kann im LOCATE-Kommando festgelegt werden oder ihre Zuweisung wird dem LOCATE-Programm überlassen.

6.3.2. Intrasegmentaufrufe

Zusätzlich zur Veränderung von Ladeadressen müssen auch die Adressen, auf die in Befehlen oder Datenelementen bezuggenommen wird, angepaßt werden. Falls sich die Adresse auf eine Stelle im selben Segment bezieht, wird dies ein Intrasegmentaufruf genannt.

Ein Sprungbefehl, der sich – um mehrere Befehle rückwärts – auf den Anfang einer Schleife bezieht, ist eine derartige Bezugnahme. Der vom Übersetzer in die Adreßtabelle eingetragene Wert ist einfach relative oder absolute Adresse der aufgerufenen Speicherstelle. Falls es sich um eine relative Adresse handelt, wird sie von LINK und LOCATE angepaßt, indem die Basisadresse des Segments addiert wird.

6.3.3. Intersegmentaufrufe

Wenn sich eine Adresse in einem Befehl auf eine Stelle in einem anderen Segment desselben Moduls bezieht, wird dies ein Intersegmentaufruf genannt. Ein Befehl im Codesegment, der sich auf den Namen einer Variablen im Datensegment bezieht, ist ein Beispiel.

Wenn LINK die Segmente zusammenfügt, um einen neuen Objektmodul zu erzeugen, werden die Intersegmentaufrufe geändert, um den Bezug auf die neuen relativen Adressen oder die absoluten Adressen der Stellen in den anderen Segmenten herzustellen.

LOCATE wandelt die relative Adresse eines Intersegmentaufrufs in eine absolute Adresse um, indem es sie zur Basisadresse des zum Aufruf gehörigen Segments addiert.

6.3.4. Externaufrufe und PUBLIC-Symbole

Wenn sich eine Adresse in einem Befehl auf ein Symbol bezieht, die nicht im selben Modul enthalten ist, so wird dies ein Externaufruf genannt. Diese Bezugnahme unterscheidet sich von den obigen, weil der Übersetzer nichts über die zu diesen Symbolen gehörige Adresse weiß. Deshalb muß man solche Symbole als EXTERNAL deklarieren. Sie werden dadurch als externe Symbole bekannt, was bedeutet, daß sie in anderen Moduln definiert sind. Die Befehle, die sich auf derartige Symbole beziehen, sind Externaufrufe.

Der Modul, der einen Externaufruf enthält, wird als »unbefriedigter« Modul bezeichnet oder es heißt, daß er einen ›unbefriedigten‹ Externaufruf enthält. LINK fügt diesen Modul mit dem Modul zusammen, der den richtigen ›Anschluß‹ besitzt. Dieser Anschluß ist ein PUBLIC-Symbol, das mit dem EXTERNAL-Symbol übereinstimmt. Ein PUBLIC-Symbol ist ein Symbol, das in dem Primärprogrammodul als PUBLIC deklariert und vom Übersetzer in den Objektcode mit seiner Adresse eingebracht worden ist.

Wenn LINK zwei Moduln ›verbindet‹, indem es ein EXTERNAL-Symbol mit einem PUBLIC-Symbol in Übereinstimmung bringt, wird der Wert des PUBLIC-Symbols (seine relative oder absolute Adresse) zum Adreßteil des Befehls, der sich auf das PUBLIC-Symbol bezieht. Dann entfernt LINK den Externaufruf. Er wird durch einen Intersegment- oder Intrasegmentaufruf ersetzt, falls das PUBLIC-Symbol eine relative Adresse hat. Hat das PUBLIC-Symbol eine absolute Adresse, wird der Externaufruf nicht ersetzt, weil keine weitere Adreßanpassung nötig ist.

Falls der von LINK erzeugte Modul noch unbefriedigte EXTERNAL-Namen enthält, gibt LINK über jeden eine Meldung aus. Dieser Modul muß erneut gebunden werden mit Moduln, die die übereinstimmenden PUBLIC-Symbole enthalten, um einen befriedigten Modul zu erhalten. Die von LINK gebrachten Meldungen über unbefriedigte EXTERNAL-Namen zeigen nicht an, daß ein Fehler vorliegt. In Zwischenstufen einer Entwicklung, bevor alle Moduln des Programms vollständig sind, muß man damit rechnen, daß LINK derartige Meldungen hervorbringt. Wenn man außerdem einen Namen als EXTERNAL deklariert hat, sich aber im Programm niemals darauf bezieht, so erzeugt Link eine Meldung über den unbefriedigten EXTERNAL-Namen, obwohl kein unbefriedigter Externaufruf vorliegt.

Dies bedeutet nur, daß man den Zustand des Objektcode in einer Datei so identifizieren kann. Wenn man die Speicherübersichten (MAP) der LINK- und LOCATE-Läufe aufhebt, kann man sich damit auf dem Laufenden halten. Man kann aber auch die ›Extension‹ des Dateinamens nutzen, um die Art des Inhalts zu kennzeichnen.

Wenn ein Modul keine Externaufrufe enthält, wird er als ›befriedigt‹ bezeichnet. Die PUBLIC-Symbole werden aus dem Objektmodul nicht entfernt, weil sie möglicherweise später benötigt werden, falls ein neuer Modul hinzugefügt wird, der einen Externaufruf für eins der PUBLIC-Symbole hat.

Falls LOCATE einen Externaufruf in einem von ihm verarbeiteten Modul findet, gibt es eine Warnungsmeldung aus, fährt aber fort, den absoluten Modul zu erzeugen. Der absolute Modul kann ausgeführt werden, vielleicht im Testmodus mit einem Unterbrechungspunkt, der so festgelegt ist, daß er die Verarbeitung anhält, bevor der Befehl erreicht ist, der den unbefriedigten Externaufruf enthält. Falls dieser Befehl ausgeführt wird, sind die Ergebnisse nicht vorauszusagen, weil die Adresse in dem Befehl falsch ist.

6.4. Bibliotheken

Bibliotheken erleichtern die Arbeit, Programme mit Hilfe des LINK-Programms aus Objektmoduln aufzubauen. Das Bibliotheksverwaltungsprogramm LIB erzeugt und wartet Dateien, die Objektmoduln enthalten. LIB erzeugt ein Inhaltsverzeichnis der Moduln in jeder Bibliotheksdatei, damit man sich über die enthaltenen Moduln informieren kann.

Das LINK-Programm behandelt Bibliotheksdateien in besonderer Art. Wenn man eine Bibliotheksdatei als Eingabe für LINK festlegt, nachdem man den einzuschließenden Hauptmodul festgelegt hat, dann durchsucht LINK die Bibliothek nach Moduln, die die passenden PUBLIC-Symbole besitzen, um die unbefriedigten Externaufrufe im ersten Modul in Übereinstimmung zu bringen. Falls ein Modul aus der Bibliothek eingefügt wird, aber selbst unbefriedigte Externaufrufe enthält, durchsucht LINK die Bibliothek erneut und versucht, den Modul mit den PUBLIC-Symbolen zu finden, die die neuen Externaufrufe befriedigen. Dieser Vorgang wird wiederholt, bis alle Externaufrufe befriedigt sind, für die in der Bibliothek überhaupt ein PUBLIC-Symbol existiert.

Das Bibliotheksverwaltungsprogramm kann eine Liste der Moduln in der Bibliotheksdatei ausgeben, einschließlich der PUBLIC-Symbole in jedem Modul. Man könnte nun etwa alle Objektmoduln für ein Programm in einer Bibliothek haben oder alle Moduln, die sich auf eine spezielle Funktion beziehen. Zum Beispiel wird zusammen mit ISIS II eine Systembibliothek namens SYSTEM.LIB geliefert, die den Objekt-Code enthält, um Programme mit den im Kapitel 12 beschriebenen ISIS-II-Systemroutinen zu binden. Diese Bibliothek ist in Kapitel 13 beschrieben.

7. LOCATE-Operation

Als Eingabe verlangt das ISIS-Programm LOCATE eine Datei, die einen relativ-adressierten Objektmodul enthält, und erzeugt eine Ausgabedatei, die denselben Objektmodul (Objektcode) enthält, jedoch mit absoluten Adressen versehen. Kapitel 6 erklärt das Konzept der Relativierung, dieses Kapitel beschreibt die Arbeitsweise des LOCATE-Programms.

7.1. LOCATE-Kommando

Das LOCATE-Programm wird mit dem LOCATE-Kommando aufgerufen, dessen Format den in Kapitel 4 angegebenen Konventionen entspricht. Das LOCATE-Kommando hat folgende Syntax:

LOCATE eingabedatei TO ausgabedatei {steuerangaben}

eingabedatei — ist der Name der Datei, die den relativ-adressierten Objektmodul enthält.

ausgabedatei — ist der Name der Datei, die den absolut-adressierten Objektmodul enthalten soll. Fehlt die Angabe TO ausgabedatei, benutzt LOCATE den Namen der Eingabedatei, aber ohne die »Extension«. Falls jedoch die Eingabedatei keine Extension hat, wird bei fehlender TO-Klausel eine Fehlermeldung ausgegeben und das Programm abgebrochen. Falls eine Datei mit demselben Namen, ‹ausgabedatei› bereits vorhanden ist, wird diese überschrieben.

steuerangaben — legen ein oder mehrere Schlüsselworte fest, die die Arbeitsweise von LOCATE steuern. Die Steuerangaben und ihre Standardwerte (von LOCATE angenommene Werte, wenn eine Steuerangabe fehlt) sind im Abschnitt weiter unten erklärt. Die folgende Liste faßt die Steuerangaben zusammen.

7.1.1. Zusammenfassung der Steuerangaben

Die Segmente betreffende Steuerangaben
ORDER (segkenn, segkenn, segkenn, segkenn)
Die Folge der Angaben segkenn muß eine beliebige Kombination der vier Segmentnamen CODE, DATA, STACK und MEMORY sein, die anzeigt, in welcher Reihenfolge die Segmente im Ausgabemodul angeordnet sein sollen.

CODE (adr)
DATA (adr)
STACK (adr)
MEMORY (adr)

Diese vier Steuerangaben legen die Basisadresse (adr) für das genannte Segment fest.

Steuerangaben zur Fehlersuche

Information über das von LOCATE erzeugte Programm.

PRINT (dateiname) — gibt Informationen über das erzeugte Programm (Symboltabelle und/oder Liste der absoluten Segmentadressen) in die Datei ›dateiname‹ statt zum Bediengerät aus.

MAP	verlangt, daß eine Liste der absoluten Segmentadressen in die Information über das erzeugte Programm aufgenommen wird.
SYMBOLS	verlangt, daß die Symbole mit den zugehörigen absoluten Adressen und die Namen der beteiligten Moduln in die Symboltabelle aufgenommen werden.
LINES	verlangt, daß die Zeilennummern aus dem PL/M-Primärprogramm und die Namen der beteiligten Moduln in die Symboltabelle aufgenommen werden.
PUBLICS	verlangt, daß die PUBLIC-Symbole in die Symboltabelle aufgenommen werden.

Weitere Steuerangaben

NAME (modname)	legt einen Namen für den Ausgabemodul fest (keinen Dateinamen)
STACKSIZE (Wert)	legt die Größe des Stacksegments fest (Wert ist die Anzahl der Bytes)
START (adr)	legt die Startadresse für die Programmausführung fest.
PURGE	entfernt die Zeilennummern die Symbole mit absoluten Adressen und die PUBLIC-Symbole aus dem Ausgabemodul.
RESTARTØ	bewirkt, daß an die Stellen Ø,1,2 ein Sprungbefehl zur Startadresse des Programms geschrieben wird.

7.1.2. Fortsetzungszeilen

Falls ein LOCATE-Kommando länger als eine Zeile auf dem Bediengerät ist (sie darf nicht länger als 122 Zeichen sein), kann man es fortsetzen, indem man ein Und-Zeichen (&) vor dem Wagenrücklauf eingibt. Das Und-Zeichen darf nicht innerhalb eines Dateinamens oder einer Steuerangabe erscheinen. LOCATE fordert zur Fortsetzung der Zeile durch Ausgabe von zwei Sternen (**) auf. Falls notwendig, können in derselben Art weitere Zeilen angeschlossen werden.

7.2. Reihenfolge und Basisadressen der Segmente

Falls es nicht durch die Steuerangabe ORDER oder durch Basisadressen anders vorgeschrieben ist, adressiert LOCATE die Segmente aus dem Eingabemodul in nachstehender Reihenfolge:
Codesegment, Stacksegment, Datensegment, Freier-Speichersegment.

LOCATE weist dem Codesegment eine Basisadresse zu, die gleich der ersten Adresse oberhalb von ISIS II ist, plus der Byteanzahl, die für 13 Ein-/Ausgabepuffer benötigt wird. In Version 2.0 ist die erste Adresse oberhalb von ISIS II gleich 3ØØØH. Dies bedeutet, daß die Basisadresse des Codesegments gleich 3ØØØH + 68ØH, d. h. 368ØH ist. Die übrigen Segmente werden an das erste verfügbare Byte hinter dem vorangegangenen Segment gelegt.

Reihenfolge und Basisadresse eines Segments können durch Steuerangaben, die als Schlüsselwörter im LOCATE-Kommando festgelegt sind, geändert werden. Die Steuerangabe OR-

DER kann verwendet werden, wenn man die Reihenfolge geändert haben möchte, sich aber nicht um die Basisadressen der Segmente kümmern will. Die Adressen werden zugewiesen, wie oben beschrieben, aber in der Reihenfolge der Segmente, wie sie in der Steuerangabe ORDER festgelegt ist. Es müssen alle vier Segmente festgelegt werden. Wenn man zum Beispiel die Reihenfolge des Code- und des Datensegments vertauschen will, ist folgende Steuerangabe erforderlich:

ORDER (DATA,CODE,STACK,MEMORY)

Falls man die Basisadressen der Segmente selbst festlegen möchte und dies nicht dem LOCATE-Programm überlassen will, kann man die Adressen in den Steuerangaben mit den Segmentnamen in folgender Weise festlegen:

CODE (4000H) DATA(5000H) STACK(523AH) MEMORY(8000H)

Die festgelegten Adressen können sich auf eine beliebige Basis beziehen, die durch einen Buchstaben hinter der Zahl gekennzeichnet wird:

D oder leer	dezimal
H	hexadezimal
O oder Q	oktal
B	binär

Die Reihenfolge der Segmente im Ausgabemodul kann auch durch Steuerangaben von Basisadressen geändert werden. Falls jedoch weniger als vier Steuerangaben mit Segmentnamen festgelegt werden, versucht LOCATE, die nicht festgelegten in der standardmäßigen Reihenfolge zu belegen. Dies kann zu einem Widerspruch führen, der dann in der Speicherübersicht (MAP) erscheint. Das folgende Beispiel dreht die Reihenfolge des Codesegments und des Datensegments um.

DATA(4000H) CODE(4800H) STACK(523AH)

7.3. Ausgaben von LOCATE zur Information über das erzeugte Programm

LOCATE erzeugt eine MAP und eine Symboltabelle, wenn die entsprechenden Steuerangaben festgelegt sind. Sie werden auf dem Bediengerät ausgegeben, falls in der Steuerangabe PRINT nicht anders festgelegt.

7.3.1. Liste der absoluten Adressen

Falls die Steuerangabe MAP festgelegt ist, erzeugt LOCATE eine Liste der absoluten Anfangsadressen der Segmente und Länge und Typ jedes Segments. Die Länge des Freien-Speichersegments wird von LOCATE errechnet aus der Differenz zwischen der höchsten Speicherstelle (angegeben durch die Monitorroutine MEMCK) und der Basisadresse des Freien-Speichersegments. Falls die Steuerangabe ORDER festlegt, daß ein weiteres Segment dem Freien-Speichersegment folgen soll, wird eine Konfliktmeldung ausgegeben.

Im folgenden Beispiel wird die gebundene Objekt-Datei :F1: TELE.LNK mit dem Dienstprogramm LOCATE in ein ladefähiges Modul (absolute Adressen) überführt.

```
ISIS-II OBJEKT LOCATER, V1.3
INVOKED BY:
-LOCATE :F1:TELE.LNK PRINT(:LP:) CODE(5000H) DATA(6000H)&
** STACK(6500H) MEMORY(7000H) MAP PUBLICS
SYMBOL TABLE OF MODULE TELETEXT,
READ FROM FILE :F1:TELE.LNK,
WRITTEN TO FILE :F1:TELE

VALUE TYPE SYMBOL

0040H PUB  ISIS
6BD8H PUB  EXIT
57CEH PUB  L5X7
6BADH PUB  AUS1
5632H PUB  ZOUT
6BC5H PUB  CLOSE
6C2FH PUB  WRITE
6C4EH PUB  QP0034
6C50H PUB  QP0035
6C0BH PUB  READ
6BE7H PUB  OPEN
5028H PUB  TEIN
607EH PUB  STATUS
6000H PUB  HILF1
6001H PUB  HILF2
60FFH PUB  AUSPU
6002H PUB  TEPU1
6080H PUB  ACTUALCOUNT
UNSATISFIED EXTERNAL REFERENCE AT 500FH
UNSATISFIED EXTERNAL REFERENCE AT 5012H
UNSATISFIED EXTERNAL REFERENCE AT 523DH

MEMORY MAP OF MODULE TELETEXT,
READ FROM FILE :F1:TELE.LNK,
WRITTEN TO FILE :F1:TELE,
START ADDRESS OF 5000H.

ADDRESS LENGTH  SEGMENT

5000H   1C62H   CODE
6000H   0129H   DATA  *CONFLICT*
6500H   0038H   STACK  *CONFLICT*
7000H   86C0H   MEMORY.
```

Aufgrund der Steuerangabe PUBLICS werden alle PUBLIC-Deklarationen in die Symboltabelle aufgenommen.

Mit der Fehlermeldung

UNSATISFIED EXTERNAL REFERENCE AT...

wird auf einen EXTERN-Aufruf eines Unterprogramms hingewiesen, zu dem die zugehörige PUBLIC-Deklaration fehlt.

In der MAP werden die Startadressen der verschiedenen Programm-Segmente, aus denen sich der Modul zusammensetzt, aufgeführt.

Der in der MAP aufgeführte Adreß-Konflikt tritt auf, weil bei den Angaben der Basisadressen des DATA- und des STACK-Segmentes die reale Größe des CODE-Segments zu klein angesetzt wurde.

So beginnt der Code des über die Bibliothek PL/M80.LIB eingebundenen Programms GPØØ35 bei der Speicheradresse 6C5ØH – also im vorgegebenen STACK-Bereich.

7.3.2. Symboltabelle

Drei Steuerangaben bestimmen die Ausgabe des Inhalts der Symboltabelle: SYMBOLS, LINES und PUBLICS.
Wenn SYMBOLS festgelegt ist, werden alle Symbole in der Symboltabelle mit ihren absoluten Adressen aufgelistet. Sie werden als Typ SYM gekennzeichnet.
Wenn PUBLICS festgelegt ist, werden alle im Primärprogramm als PUBLIC deklarierten Symbole in der Symboltabelle aufgelistet. Sie werden als Typ PUB gekennzeichnet. Die Steuerangabe LINES bezieht sich nur auf PL/M-Programme.
Wenn LINES festgelegt ist, werden die Zeilennummern aus dem Primärprogramm und die Namen der Eingabemoduln mit in die Symboltabellen-Ausgabe aufgenommen. Zeilennummerneinträge werden als Typ LIN und Modulnamen als Typ MOD gekennzeichnet.

Hinweis: Der Übersetzer PL/M80 und der Assembler ASM80 bringen keine Informationen über Symbole oder Zeilennummern in die Objektdatei ein, außer man gibt das Schlüsselwort DEBUG im Übersetzeraufruf-Kommando an. Dazu wird auf die im Vorwort genannte Bedienungsanleitung verwiesen.

7.4. Weitere Steuerangaben

Die Steuerangabe NAME legt einen Modulnamen fest, der dem Ausgabemodul zugewiesen wird. Wenn die Steuerangabe NAME fehlt, ist der Name des Ausgabemoduls derselbe wie der Modulname der Eingabedatei.

Die Steuerangabe STACKSIZE legt die Länge des Stacksegments im Ausgabemodul in Bytes fest. Fehlt diese Angabe, wird die Länge des Stacks aus dem Eingabemodul entnommen. Auf jeden Fall addiert LOCATE 12 Bytes zur Länge des Stacks des Eingabemoduls.

Die Steuerangabe START legt die Adresse des ersten auszuführenden Befehls fest. Fehlt diese Angabe, wird die Startadresse dem Eingabemodul entnommen.

Die Steuerangabe PURGE entfernt die Zeilennummern, die Symbole mit ihren zugehörigen absoluten Adressen und die PUBLIC-Symbole aus der Ausgabe zur Information über das erzeugte Programm. PUBLIC-Symbole, falls vorhanden, wurden gebraucht, um den absoluten

Modul mit anderen Moduln zu binden. Zeilennummern und lokale Symbole sind nützlich bei der Fehlersuche. Die Steuerangabe PURGE erlaubt es, die Größe eines vollständig ausgetesteten Moduls zu verringern, was sich als Einsparung von Diskettenplatz und Verkürzung der Zeit zum Laden des Programms auswirkt.

Die Steuerangabe RESTARTØ speichert eine Sprunganweisung in den Stellen Ø,1 und 2 des absoluten Moduls. Die Adresse in der Sprunganweisung ist die Startadresse des Programms (die Adresse des ersten auszuführenden Befehls), die aus dem Eingabemodul oder von der Steuerangabe START entnommen wird. Man pflegt die Steuerangabe RESTARTØ zu verwenden, wenn ein absoluter Modul im Prototyp eines Systems geladen wird, d.h. entweder als Einzelsystem oder als unter ICE 80 emuliertes System. Wenn das 8080-System das RESET-Signal als Eingabe hat, wird der Befehlszähler auf Null gesetzt und die Ausführung mit dem Sprung auf den Programmanfang begonnen. Ein mit der Steuerangabe RESTARTØ versehener absoluter Modul ist nicht kompatibel zu ISIS II, weil dieses nicht erlaubt, daß Anwendercode in die Stelle Ø geladen wird.

7.5. Fehlermeldungen

Die bei LOCATE auftretenden schwerwiegenden Fehler bewirken, daß eine der folgenden Meldungen an das Bedienungs-Ausgabegerät gegeben wird. LOCATE beendet die Verarbeitung und gibt die Steuerung an ISIS II zurück.

dateiname, PROGRAM EXCEEDS 64K
Diese Meldung zeigt an, daß der zu adressierende Modul den Speicherbereich, der maximal 64K hat, überschreitet.

dateiname, ILLEGAL STACK CONTENT RECORD
Diese Meldung bedeutet, daß versucht worden ist, dem Stacksegment einen Anfangswert zuzuweisen. Dies ist unzulässig.

dateiname, NO MODULE HEADER RECORD
Diese Meldung wird ausgegeben, wenn der Eingabemodul keinen Modul-Kopfsatz besitzt. Dies bedeutet, daß er kein korrektes relativ-adressiertes Objektmodulformat hat.

Bei weniger schwerwiegenden Fehlern gibt LOCATE Meldungen aus, die in die Map-Datei geschrieben und/oder an das Bedienungs-Ausgabegerät gegeben werden. LOCATE vollendet die Verarbeitung, bevor es die Steuerung an ISIS II zurückgibt.

INPAGE SEGMENT COERCED TO PAGE RELTYP
Wenn diese Meldung ausgegeben wird, hat ein Segment, das als innerhalb einer Seite (256 Bytes) plazierbar bezeichnet worden ist, auf mehrere Seiten ausgedehnt werden müssen. Für die Erklärung der Kennzeichnungen PAGE, INPAGE wird auf Kapitel 10 verwiesen.

MEMORY CONFLICT FROM xxxxH THROUGH xxxxH
Diese Meldung wird ausgegeben, wenn zwei oder mehr Programmsegmente in Bezug auf dieselben Speicherstellen im Konflikt stehen.

UNSATISFIED EXTERNAL REFERENCE AT xxxxH
Diese Meldung wird ausgegeben, wenn bei der genannten Adresse auf eine Speicherstelle Bezug genommen wird, die nicht in dem momentan adressierten Modul enthalten ist.

8. LINK-Operation

Das Programm LINK erlaubt es, Objektmoduln aus mehreren Eingabedateien zu einem Objektmodul in einer Ausgabedatei zu kombinieren. In dem Verfahren, Moduln zu kombinieren, paßt LINK alle Adressen an, so daß sie relativ sind zu den jeweiligen Segmentanfängen im neuen Ausgabemodul. LINK durchsucht ferner Bibliotheken nach Moduln, die die Externaufrufe in zu kombinierenden Moduln befriedigen, und fügt sie in den neuen Ausgabemodul ein. Falls irgendwelche unbefriedigten Externaufrufe im Ausgabemodul übrigbleiben, bringt LINK eine Meldung in die Übersichtsliste, die den Aufbau des neuen Moduls beschreibt.

Der Ausgabemodul muß vom LOCATE-Programm verarbeitet werden, bevor er ausgeführt werden kann. Das LOCATE-Programm weist dem Objektmodul absolute Speicheradressen zu. Der Ausgabemodul kann auch als Eingabe für LINK benutzt werden, um mit anderen Moduln zu einem neuen und erweiterten Ausgabemodul kombiniert zu werden.

8.1. LINK-Kommando

Das LINK-Programm wird zu seiner Ausführung durch das LINK-Kommando aufgerufen, dessen Format den in Kapitel 4 angegebenen Konventionen entspricht. Das LINK-Kommando hat folgende Syntax:

LINK eingabeliste TO ausgabedatei {steuerangaben}

eingabeliste — kann einer oder beide der folgenden Einträge sein:

dateiname {(modname, ...)}, ...
PUBLICS (dateiname, ...), ...

Im ersten Eintrag legt dateiname eine Datei fest, die einen Objektmodul enthält oder eine Datei, die eine Bibliothek von Objektmoduln enthält. Falls dateiname keine Bibliotheksdatei ist, wird der darin enthaltene Modul in den Ausgabemodul eingefügt. Falls dateiname eine Bibliotheksdatei ist und modnamen festgelegt sind, dann werden nur die festgelegten Moduln in den Ausgabemodul gebunden. Falls die Angabe modname,... fehlt und dateiname eine Bibliotheksdatei ist, werden nur die Moduln eingefügt, die Externaufrufe befriedigen in Moduln, die

1. davor in der Eingabeliste festgelegt sind oder
2. soeben aus derselben Bibliothek eingefügt worden sind, um 1. oder 2. zu befriedigen.

Der zweite Eintrag legt Moduln fest, bei denen nur ihre absoluten PUBLIC-Deklarationen in den Ausgabemodul einzufügen sind. Dies erlaubt, Moduln zu binden, ohne sie in der Ausgabedatei zu kombinieren, damit sie getrennt geladen werden können. Diese Möglichkeit ist im Kapitel 10 beschrieben.

ausgabedatei	legt die Datei fest, die den Objektmodul enthalten soll, der durch das Binden der Eingabemoduln entstanden ist.
steuerangaben	legt ein oder mehrere Schlüsselwörter fest, die die LINK-Operation steuern. Die Steuerangaben sind:
MAP	Diese Steuerangabe verlangt, daß eine Liste mit Informationen über die Ausgabedatei erzeugt wird. Diese wird an das Bediengerät oder an die in der Steuerangabe PRINT festgelegte Datei gegeben.

NAME(modname)	Diese Steuerangabe legt den Namen fest, der dem Ausgabemodul zugewiesen werden soll. Der Name kann aus bis zu 31 Zeichen bestehen, deren erstes ein Buchstabe sein muß. Als Buchstaben zählen auch die Sonderzeichen ? und @. Der Rest kann aus Buchstaben oder Ziffern bestehen. Falls kein NAME(modname) festgelegt ist, wird der Namensteil der Ausgabedatei als Modulname verwendet.
PRINT(dateiname)	Diese Steuerangabe legt die Datei fest, die die Liste mit Informationen über die Ausgabedatei enthalten soll. Fehlt die Angabe, wird diese an das Bediengerät gegeben.

Falls das LINK-Kommando länger als eine Zeile am Bediengerät ist (nicht länger als 122 Zeichen), kann man es fortsetzen durch Eingeben eines Und-Zeichens (&) vor dem Wagenrücklauf. Das Und-Zeichen darf nicht innerhalb eines Dateinamens oder einer Steuerangabe erscheinen. LINK fordert zur Fortsetzung der Zeile durch Ausgabe von zwei Sternen (**) auf. Falls notwendig, können in derselben Art weitere Zeilen angeschlossen werden.

8.2. Liste mit Informationen über den Ausgabemodul

Die von LINK erzeugte Liste mit Informationen über den Ausgabemodul (Map) enthält folgendes:

1. die LINK-Erkennungsmeldung (es sei denn, es wird auf :CO: ausgegeben)
2. das zum Aufruf von LINK verwendete Kommando (es sei denn, es wird auf :CO: ausgegeben)
3. die Segmentlängen im Ausgabemodul
4. die Adressen des absolut-adressierten Teils des Ausgabemoduls
5. die Namen der Eingabemoduln
6. unbefriedigte EXTERNAL-Namen
7. Fehlermeldungen

Das folgende Beispiel zeigt eine Bindeübersicht für den Ausgabemodul TELETEXT in der Datei TELE.LNK. Erzeugt wird TELETEXT mit:

a) 5 explizit in der Eingabeliste aufgeführten Objekt-Moduln
(Dabei liegen den Moduln ORG, TEIN, ZOUT, ZAUSW und MATRIX PL/M-Quellprogramme zugrunde. Der Modul AUS1 wurde mit dem Assembler ASM8O erzeugt).
b) der Ausgabe PUBLICS (:F1: ZAUSW.OBJ).
Mit diesem Eingabeliste-Eintrag wird erreicht, daß ausschließlich die absoluten PUBLIC-Deklarationen des Moduls ZAUSW in den Ausgabemodul eingefügt sind.
c) der Angabe der beiden Bibliotheksdateien SYSTEM.LIB und PL/M8Ø.LIB.
Dabei werden über die Systembibliothek SYSTEM.LIB die EXTERN-Aufrufe der Systemaufrufe CLOSE.EXIT usw., die in einigen Anwender-Programm-Moduln angegeben sind, ›befriedigt‹. Die PL/M-spezifische Bibliothek PL/M8Ø.LIB ist nur dann anzugeben, wenn die PL/M-Quellprogramme mit einer PL/M-Übersetzer-Version ≤3.Ø übersetzt wurden.

```
ISIS-II LINKER, V1.3 WAS INVOKED BY:
LINK :F1:ORG.OBJ,:F1:TEIN.OBJ,PUBLICS(:F1:ZAUSW.OBJ),&
:F1:ZOUT.OBJ,:F1:MATRIX.OBJ,:F1:AUS1.OBJ,SYSTEM.LIB,&
PLM80.LIB TO :F1:TELE.LNK NAME(TELETEXT) MAP PRINT(:LP:)

LINK MAP FOR :F1:TELE.LNK(TELETEXT)

SEGMENTS INCLUDED:
  CODE      LENGTH: 1C62H
  DATA      LENGTH:  129H
  STACK     LENGTH:   2CH

INPUT MODULES INCLUDED:
 :F1:ORG.OBJ(ORG)
 :F1:TEIN.OBJ(TEXTEINGABE)
 :F1:ZAUSW.OBJ(ZEICHENAUSWERTUNG) (PUBLICS)
 :F1:ZOUT.OBJ(ZEICHENAUSGABE)
 :F1:MATRIX.OBJ(MATRIX)
 :F1:AUS1.OBJ(AUS1)
 SYSTEM.LIB(CLOSE)
 SYSTEM.LIB (EXIT)
 SYSTEM.LIB(ISIS)
 SYSTEM.LIB(OPEN)
 SYSTEM.LIB(READ)
 SYSTEM.LIB(WRITE)
 PLM80.LIB(QP0034)

UNRESOLVED EXTERNAL NAMES:
MATADR
ZAUSW
SCAN
```

8.3. Reihenfolge der Moduln im Ausgabemodul

LINK kombiniert die Moduln aus der Eingabeliste durch Zusammenfügen aller Codesegmente aus der Eingabeliste zu einem Codesegment, aller Datensegmente zu einem Datensegment und aller Stacksegmente zu einem Stacksegment. Die Länge des Freien-Speichersegments wird zahlenmäßig von LOCATE ermittelt. Alle absoluten Segmente in den Eingabemoduln werden mit ungeänderten absoluten Adressen in den Ausgabemodul übertragen. Falls absolute Segmente hinsichtlich ihrer Adressen im Konflikt stehen, wird eine dahingehende Meldung in die Liste der Informationen zur Ausgabedatei gegeben.

Die Reihenfolge des Zusammenfügens entspricht der Reihenfolge, in der die Moduln in der Eingabeliste festgelegt sind. Der erste Modul ist der erste in der Reihenfolge des Zusammenfügens. Die Segmente des zweiten Moduls folgen den Segmenten des ersten Moduls unmittelbar anschließend.

Beispiel:
Das folgende LINK-Kommando und die Erklärung zeigen, wie Moduln kombiniert werden.
LINK A,B TO C
Modul A enthält Code-, Daten- und Stacksegment.

Modul B enthält Code- und Datensegment. Sie sind alle relativ-adressiert ohne Bezug auf Seiteneinteilung. Der sich ergebende Modul C' hat folgenden Aufbau

Codesegment	Datensegment	Stacksegment	Speichersegment
aus Modul A aus Modul B	aus Modul A aus Modul B	aus Modul A	Länge durch LOCATE ermittelt

Beim Binden der Eingabemoduln werden die Externaufrufe befriedigt. Wenn z. B. der Modul A einen Sprung in den Modul B enthält, ist dieser nicht mehr extern, sobald A und B zu C gebunden worden sind. Falls A und B einen Externaufruf für einen Modul in einer Bibliotheksdatei aufweist, muß die Bibliothek in der Eingabeliste hinter dem Modul festgelegt werden, der sich darauf bezieht. Angenommen, die Bibliotheksdatei RTNS.LIB enthalte einen Modul, der einen Externaufruf in A befriedigt. Dann könnte die Eingabeliste wie folgt festgelegt werden:
A, RTNS.LIB,B oder A,B, RTNS.LIB.

8.4. Fehlermeldungen

Schwerwiegende Fehler, die bei LINK auftreten, bewirken das Ausgeben einer Meldung am Bediengerät. LINK beendet die Verarbeitung und gibt die Steuerung an ISIS II zurück.

dateiname, BAD FIXUP RECORD
Fehlerhafter Fixup-Satz

dateiname, BAD RECORD SEQUENCE
Fehlerhafte Satzreihenfolge

dateiname, CHECKSUM ERROR
Prüfsummenfehler

dateiname, ILLEGAL RECORD FORMAT
Unzulässiges Satzformat

dateiname, PREMATURE EOF
Vorzeitiges Dateiende

dateiname, RECORD TOO LONG
Zu langer Satz

dateiname, RECORD TOO SHORT
Zu kurzer Satz

dateiname, RELO FILE SEQUENCE ERROR
Reihenfolgefehler in relativ-adressierter Datei

Diese Meldungen zeigen einen Fehler im internen Format eines zu bindenden relativ-adressierten Objektmoduls an. Wahrscheinlich ist dieser aufgetreten, als der Übersetzer den Modul erzeugt hat. Das Primärprogramm ist nochmals zu übersetzen und erneut zu binden.

INSUFFICIENT MEMORY
Diese Meldung bedeutet, daß LINK die Verarbeitung nicht vollenden konnte, weil der Arbeitsspeicher, der hauptsächlich für die Symboltabelle benötigt wird, nicht ausreichend war.

dateiname, PHASE ERROR
Diese Meldung sollte nur dann auftreten, wenn ein Hardwarefehler bewirkt hat, daß Daten beim zweiten Lesen einer Eingabedatei unvereinbar sind mit den Daten beim ersten Lesen. In allen anderen Fällen dürfte ein Problem in LINK vorhanden sein. Dieses sollte durch einen »Software-Problem-Report (SPR)« bekanntgegeben werden.

dateiname, SEGMENT TOO LARGE
Falls ein Ausgabesegment größer als 64 K ist, wird diese Meldung ausgegeben und ein schwerwiegender Fehler tritt auf. Die in der Meldung genannte Datei kann nicht zu dem Ausgabesegment gebunden werden, weil dieses dann größer als 64 K würde.

Nicht schwerwiegende Fehler, die von LINK angezeigt werden, werden in die Liste der Informationen zur Ausgabedatei geschrieben. LINK vollendet die Verarbeitung, bevor es die Steuerung an ISIS II zurückgibt.

MORE THAN 1 MAIN MODULE
Diese Meldung zeigt an, daß LINK mehr als einen Hauptmodul in der Eingabeliste gefunden hat. Alle Hauptmoduln werden in den Ausgabemodul gebunden, aber die Startadresse des Ausgabemoduls wird dem ersten Hauptmodul in der Eingabeliste entnommen.

modname, MODULE NOT FOUND IN LIBRARY
Diese Meldung bedeutet, daß ein Modul aus der Eingabeliste nicht in der festgelegten Bibliothek gefunden worden ist.

name, HAS DIFFERENT TYPES
Dieser Fehler tritt auf, wenn ein EXTERNAL-Name und der passende PUBLIC-Name nicht vom selben Typ sind.

name, MULTIPLE DEFINITION
Dieser Fehler tritt auf, wenn zwei oder mehr PUBLIC-Symbole identisch sind.

9. LIB-Operation

Das Programm LIB erlaubt es, Dateien mit Bibliotheken von Objektcodemoduln zu erzeugen, diese Bibliotheken durch Hinzufügen und Löschen von Moduln zu warten und eine Liste der Moduln in der Bibliotheksdatei zu erhalten. Bibliotheken können als Eingabe für den Binder dienen, der aus der Bibliothek automatisch die Moduln bindet, welche die Externaufrufe in den zu bindenden Moduln befriedigen.

9.1. LIB-Kommando

Das Bibliotheksverwaltungsprogramm wird zu seiner Ausführung durch das LIB-Kommando in der Form
LIB
aufgerufen.
Die Tätigkeit des Bibliotheksverwaltungsprogramms wird durch die Eingabe von Kommandos gesteuert, die anzeigen, welche Operation LIB ausführen soll. LIB fordert durch einen Stern (*) zur Eingabe von Kommandos auf. Die Kommandos lauten:

CREATE
ADD
DELETE
LIST
EXIT

9.1.1. Fortsetzungszeilen

Falls ein Kommando für LIB länger als eine Zeile auf dem Bediengerät ist (nicht länger als 122 Zeichen), kann man es fortsetzen, indem man ein Und-Zeichen (&) vor dem Wagenrücklauf eingibt. Das Und-Zeichen darf nicht innerhalb eines Dateinamens oder einer Steuerangabe erscheinen. LIB fordert zur Fortsetzung der Zeile durch Ausgabe von zwei Sternen (**) auf. Falls notwendig, können in derselben Art weitere Zeilen angeschlossen werden.

9.1.2. CREATE-Kommando (Erzeugen einer Bibliotheksdatei)

Das CREATE-Kommando erzeugt eine leere Bibliotheksdatei. Man muß das ADD-Kommando benutzen, um der Bibliotheksdatei Moduln hinzuzufügen. Das Create-Kommando hat folgende Syntax:

CREATE dateiname

dateiname, legt den Namen fest, welcher der neuen Bibliotheksdatei zugewiesen werden soll. Falls eine Datei mit diesem Namen bereits vorhanden ist, wird eine Meldung an das Bediengerät gegeben und durch LIB ein anderes Kommando angefordert.

9.1.3. ADD-Kommando (Hinzufügen von Moduln zu einer Bibliotheksdatei)

Das ADD-Kommando fügt Objektmoduln zu einer Bibliotheksdatei hinzu. Das ADD-Kommando hat folgende Syntax:

ADD ursprdatei (modname, ...), ... TO bibldatei

ursprdatei kann der Name einer Bibliotheksdatei sein oder der Name einer Datei, die einen Objektmodul enthält. Falls eine Bibliotheksdatei festgelegt ist, werden alle in ihr enthaltenen Moduln hinzugefügt, außer, wenn modnamen festgelegt sind.

modname kann nur festgelegt werden, wenn ursprdatei eine Bibliotheksdatei ist. Es werden nur die durch modnamen festgelegten Objektmoduln der bibldatei hinzugefügt.

bibldatei ist die Bibliotheksdatei, die durch das Hinzufügen von Moduln aus der ursprdatei geändert werden soll.

9.1.4. DELETE-Kommando (Löschen von Moduln in einer Bibliotheksdatei)

Das DELETE-Kommando löscht Moduln in einer Bibliotheksdatei. Das DELETE-Kommando hat folgende Syntax:

DELETE bibldatei (modname, ...)

modname legt die Objektmoduln fest, die in der Bibliotheksdatei bibldatei gelöscht werden sollen.

9.1.5. LIST-Kommando (Auflisten der Bibliotheksmoduln und ggf. ihrer PUBLIC-Symbole)

Das LIST-Kommando listet das Modulinhaltsverzeichnis von Bibliotheksdateien auf. Das LIST-Kommando hat folgende Syntax:

LIST bibldatei {(modname, ...)}, ... {TO listdatei} {PUBLICS}

bibldatei ist der Name der Bibliotheksdatei, deren Modulinhaltsverzeichnis aufgelistet werden soll, außer, wenn modnamen festgelegt sind. In solchem Falle werden nur die Informationen über die festgelegten Moduln aufgelistet.

listdatei ist der Name der Datei, die das Modulinhaltsverzeichnis enthalten soll. Fehlt diese Angabe, wird das Inhaltsverzeichnis auf dem Bediengerät aufgelistet.

PUBLICS legt fest, daß die PUBLIC-Namen jedes Moduls aufgelistet werden sollen. Fehlt diese Angabe, werden nur die Modulnamen aufgelistet.

Werden mehrere Modulinhaltsverzeichnisse mit ihren PUBLIC-Namen angefordert, erscheint folgendes Format:

Bibliotheksname
Modulname
PUBLIC-Symbol
.
.
.
.
Modulname
PUBLIC-Symbol
.
.
.
.

9.1.6. EXIT-Kommando (Rückgeben der Steuerung an ISIS II)

Das EXIT-Kommando gibt die Steuerung an ISIS II zurück. Wenn ein Arbeitsgang zur Bibliotheksverwaltung beendet ist, gibt man das EXIT-Kommando ein, gefolgt von einem Wagen-

rücklauf. Dies beendet das Bibliotheksverwaltungsprogramm und gibt die Steuerung an ISIS II zurück, das durch einen Bindestrich (–) zur Eingabe eines Kommandos auffordert.

Beispiel:
Im folgenden Beispiel werden die Objektcodemoduln, aus denen sich der Modul TELE.LNK u.a. zusammensetzt, in der Bibliothek TELE.LIB zusammengefaßt.

```
– LIB
  ISIS-II LIBRARY MANAGER,V1.Ø
* CREATE :F1:TELE.LIB
* ADD :F1:ORG.OBJ,:F1:TEIN.OBJ,:F1:ZAUSW.OBJ,&
**:F1:SCAN.OBJ,:F1:ZOUT.OBJ,:F1:MATRIX:OBJ. TO &
**:F1:TELE.LIB
* LIST:F1:TELE.LIB TO :LP: PUBLICS
      :F1:TELE.LIB
        ORG
         HILF1
         HILF2
        TEXTEINGABE
         STATUS
         TEPU1
         TEIN
         ACTUALCOUNT
        ZEICHENAUSWERTUNG
         ZAUSW
         MATADR
        UEBERWACHER
         SCAN
        ZEICHENAUSGABE
         ZOUT
         AUSPU
        MATRIX
         L5X7
      :F1:TELE.LIB
        ORG
        TEXTEINGABE
        ZEICHENAUSWERTUNG
        UEBERWACHER
        ZEICHENAUSGABE
        MATRIX
* EXIT
–
```

10. Relativieren und Binden für Fortgeschrittene

Die in diesem Kapitel gebrachten Informationen werden benötigt, falls ein Programm eine der folgenden Anforderungen stellt:

1. Das Programm soll so mit einem Modul gebunden werden, daß zwar alle Externaufrufe befriedigt, ohne daß aber die Segmente mit ihm zusammengefügt werden, so daß getrennt geladen werden kann.
2. Das 8080-Assembler-Programm oder -Programmsegment darf keine Seitengrenze überschreiten oder muß immer an einer Seitengrenze beginnen.

10.1. Binden ohne Zusammenfügen

Die Forderung, Moduln zu binden, ohne sie in einen Lademodul zusammenzufügen, entsteht oft, wenn ein Programm in »Overlaystruktur« geschrieben werden muß, weil es größer als der verfügbare Speicherplatz ist. Wenn ein Programmteil nicht mehr benötigt wird, kann in denselben Speicherbereich ein anderer Teil geladen werden, der dann den nicht mehr benötigten Teil überlagert. Bei ISIS II müssen getrennt zu ladende Programme oder Programmteile in getrennten Dateien stehen. Das erste Laden kann dadurch geschehen, daß man den Dateinamen als Kommando eingibt. Die anschließenden Ladevorgänge geschehen vom Programm her mit dem LOAD-Systemaufruf oder mit einer Ein-/Ausgaberoutine.

Eine andere Forderung zum Binden ohne Zusammenfügen entsteht aus der Notwendigkeit, ein Produkt an diverse Kundenwünsche anpassen zu können, so daß man verschiedenartige ROM's an derselben Stelle einsetzen kann, abhängig von der Produktausführung, die der Kunde auswählt. Der für jedes dieser ROMs geschriebene Code muß zum Hauptmodul gebunden werden, ohne mit ihm zusammengefügt zu werden.

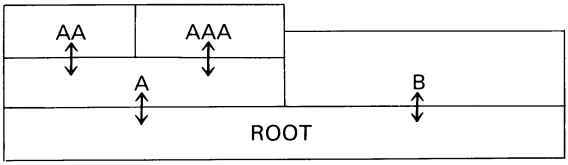
Bei der typischen Anwendung von LINK und LOCATE werden Moduln mit Externaufrufen mit Moduln zusammengefügt, die passende PUBLIC-Symbole haben, um einen Modul ohne unbefriedigte Externaufrufe zu erzeugen. Beim Binden ohne Zusammenfügen müssen die Externaufrufe ebenfalls befriedigt werden, d. h. sie müssen sich auf die entsprechenden PUBLIC-Symbole beziehen können. Wenn man das Schlüsselwort PUBLICS vor einer Modulliste verwendet, so sagt dies dem LINK-Programm, daß die Moduln nicht mit dem Objektmodul zusammengefügt werden sollen, sondern nur dazu verwendet werden sollen, die Adressen ihrer PUBLIC-Symbole zu liefern. Auf diese Art werden die Externaufrufe der vorher in der Eingabeliste aufgelisteten Moduln befriedigt. Zum Beispiel ergibt

LINK A, PUBLICS (B,C) TO A.SAT

einen Modul A SAT, dessen Externaufrufe für in B und C stehende Symbole befriedigt sind. Allerdings müssen B und C absolute Moduln sein, weil LINK die absoluten Adressen der PUBLIC-Symbole kennen muß. Deshalb wird die normale Reihenfolge der Anwendung von LINK vor LOCATE genau umgedreht. Man muß zuerst die Moduln B und C mit LOCATE absolutadressieren, also Moduln mit absoluten Adressen erzeugen, möglicherweise sogar mit unbefriedigten Externaufrufen. Der auf diese Art erzeugte Modul kann als temporärer Modul angesehen werden, der nur dazu verwendet wird, um die von anderen Moduln benötigten Adressen der PUBLIC-Symbole zu liefern.

Es soll folgendes Beispiel betrachtet werden. Ein Grundsegment ROOT ruft das Segment A und A später das Segment AA auf, dann überlagert AAA das Segment AA. Im folgenden Dia-

gramm dieser Überlagerungsstruktur trennen die senkrechten Linien Segmente, die nacheinander im gleichen Speicher (etwa A und B) stehen, die waagerechten Linien trennen Segmente, die gleichzeitig in verschiedenen Speicherbereichen stehen (Etwa A und AA mit AAA).



Alle Teile dieser Overlaystruktur müssen in getrennten Dateien stehen, weil sie getrennt geladen werden. Das ROOT-System ruft A und B auf; A ruft AA und AAA auf. Wenn man das ROOT-Segment durch LOCATE als erstes absolut-adressiert, kann man die von LOCATE erzeugte Speicherübersicht verwenden, um die Basisadresse von A und B zu ermitteln. Dann sind A und B durch LOCATE absolut zu adressieren. Die von LOCATE erzeugte Speicherübersicht ist zu verwenden, um die Basisadresse von AA und AAA zu ermitteln. Die von LOCATE erzeugten Moduln haben absolute Adressen zugewiesen bekommen, besitzen aber noch unbefriedigte Externaufrufe. Diese Moduln können nicht zur Ausführung in den Speicher geladen werden, können aber mit dem Schlüsselwort PUBLICS beim Bindevorgang benutzt werden.

Angenommen, den von LOCATE erzeugten Moduln sei der Zusatz TMP gegeben worden. Es wird ferner angenommen, daß ROOT Externaufrufe für PUBLIC-Symbole in A und B enthalte, A Externaufrufe für PUBLIC-Symbole in ROOT, AA und AAA, ferner AA und AAA Externaufrufe für PUBLIC-Symbole in A, ferner B Externaufrufe für PUBLIC-Symbole in ROOT. Dann würden die folgenden LINK-Kommandos die befriedigten Moduln fertig zur Ausführung liefern:

```
LINK ROOT.TMP,PUBLICS (A.TMP,B.TMP) TO ROOT
LINK A.TMP, PUBLICS (ROOT.TMP,AA.TMP,AAA.TMP) TO A
LINK AA.TMP, PUBLICS(A.TMP) TO AA
LINK AAA.TMP, PUBLICS(A.TMP) TO AAA
LINK B.TMP, PUBLICS (ROOT.TMP) TO B
```

Um nachzuweisen, daß alle Externaufrufe in den Moduln, die von LINK erzeugt wurden, befriedigt sind, kann man sie erneut mit LOCATE bearbeiten. Jetzt sollte LOCATE keine Meldung über unbefriedigte Externaufrufe mehr bringen.

Vorsicht!
Wenn man bindet, ohne die Moduln zusammenzufügen, um »Overlays« zu erzeugen, muß man die Overlay-Verwaltung bereits beim Entwurf des Systems berücksichtigen. Das heißt, bevor ein Programm sich auf ein Overlaysegment bezieht, muß gewährleistet sein, daß dieses sich im Speicher befindet. Wenn nicht, muß es in den Speicher gelesen werden. Wenn ein Segment im Speicher neue Daten enthält, die erhalten bleiben sollen, so muß es gerettet werden, bevor es von einem anderen Segment überschrieben wird. Die Möglichkeit zu binden, ohne die Moduln zusammenzufügen, bietet sozusagen das Gerüst für eine Overlaystruktur. Die Verwaltung während des Ablaufs eines in Overlaytechnik geschriebenen Programms muß selbstverständlich im Programm enthalten sein.

10.2. Spezielle Arten der Speicherzuordnung

Wenn man sich der Assemblersprache zum Schreiben von Programmen bedient, kann man die Speicherplatzzuordnung für Code und Daten besonders festlegen, d.h.

1. ohne Seitenstruktur,
2. an Seitengrenzen beginnend oder
3. als Ganzes innerhalb einer Speicherseite gespeichert.

10.2.1. Speicherzuordnung ohne Seitenstruktur

Das bedeutet, daß das Segment auf jede beliebige Adresse plaziert werden kann. Es gibt keine Einschränkung dafür, wo es im Speicher untergebracht wird. Der PL/M80-Übersetzer erzeugt ausschließlich Objektmoduln ohne Seitenstruktur.

10.2.2. Speicherzuordnung an einer Seitengrenze beginnend

Das bedeutet, daß das Segment bei einer Seitengrenze beginnen muß. Eine Speicherseite besteht aus 256 Bytes. Die erste Adresse einer Seite (die Seitengrenze) ist ein Mehrfaches von 256. Die sedezimalen Adressen der Seitengrenzen im SME sind ØØH, 1ØØH, 2ØØH, 3ØØH, 4ØØH usw. bis zum möglichen Höchstwert von FFØØH. Zu beachten ist, daß diese Adressen auf ØØH enden, weil sie alle ein Vielfaches von 256 sind (256 = 1ØØH).

10.2.3. Speicherzuordnung als Ganzes innerhalb einer Seite gespeichert

Das bedeutet, daß das Segment bei einer beliebigen Adresse beginnen kann, sofern das Segment keine Seitengrenze überschreitet. Diese Relativierungsart ist nur für Segmente mit weniger als 256 Bytes anwendbar. LINK und LOCATE geben Warnungsmeldungen aus und ändern die Speicheranordnungsart in die an einer Seitengrenze beginnende, wenn ein Segment größer als 256 Bytes ist.

10.2.4. Speicherseiten und Register H und L

Die Speicherzuordnungsarten sind für Programme gedacht, die sich bei Speicheraufrufen der Register H und L unabhängig voneinander bedienen (Für die Operatoren HIGH und LOW wird auf die im Vorwort genannte Beschreibung der Assembler-Programmiersprache verwiesen). Man kann ein Feld an einer Seitengrenze speichern und die Feldelemente durch Änderung nur des L-Registers adressieren. Falls das Feld keine Seitengrenze überschreitet, braucht man das H-Register überhaupt nicht zu ändern. Durch diese Methode kann eine Einsparung an Speicherplatz und Ausführungszeit erreicht werden, aber auch der Zugriff auf gewisse Speicherbereiche verloren gehen auf Grund der Art, wie LINK und LOCATE die Speicherzuordnung realisieren.

Wenn LINK Segmente mit unterschiedlichen Speicherzuordnungsarten zusammenfügen soll, geht es nach folgenden Regeln vor:

1. Segmente ohne Seitenstruktur folgen auf das vorangegangene Segment beim nächsten verfügbaren Byte. Das Ausgabesegment ist nur dann ohne Seitenstruktur, wenn alle Eingabesegmente von dieser Art sind, andernfalls beginnt es an einer Seitengrenze.
2. An einer Seitengrenze beginnende Segmente folgen auf das vorangegangene Segment an der nächsten verfügbaren Seitengrenze. Die Bytes vom Ende des vorangegangenen Segments bis zur Seitengrenze bleiben, falls vorhanden, ungenutzt. Das Ausgabesegment beginnt wieder an einer Seitengrenze.

3. Segmente, die als Ganzes innerhalb einer Seite zu speichern sind, werden, falls sie noch in die Seite passen, an die erste Stelle hinter dem vorangegangenen Segment plaziert. Andernfalls werden sie an die erste verfügbare Seitengrenze plaziert. Die Bytes am Ende der vorangegangenen Seite bleiben dann ungenutzt. Das Ausgabesegment ist nur dann innerhalb einer Seite speicherbar, wenn es kleiner als 256 Bytes ist und alle Eingabesegmente innerhalb einer Seite speicherbar sind. Andernfalls ist das Ausgabesegment von der 2. Speicherzuordnungsart, also an einer Seitengrenze beginnend.

Die Bytes, die beim Einhalten der Speicherzuordnungsart der Segmente ungenutzt bleiben, werden in der Liste mit Informationen des Binders LINK durch das Wort »gap« gekennzeichnet. Diese Lücken stellen ungenutzte Abschnitte des Speichers dar und sind gewissermaßen verloren.

Wenn LOCATE die Basisadressen der zu adressierenden Segmente zuweist, führt es dies so durch, daß die Speicherzuordnungsart jedes Segments gewahrt bleibt. Es gibt auch eine Meldung aus, wenn ein Segment der Art 3 in eins der Art 2 geändert wird. Falls man eine Basisadresse festlegt (und dazu die Steuerangabe CODE oder DATA im LOCATE-Kommando verwendet), die der Speicherzuordnungsart des Segments widerspricht, plaziert LOCATE das Segment an die nächsthöhere Seitengrenze und gibt eine Meldung aus.

11. ISIS-II-Programmierung

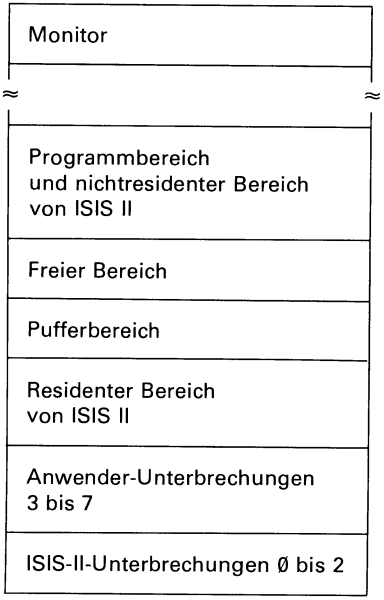
ISIS-II-Systemprogramme werden in der gleichen Weise geschrieben, wie 8080-Programme in Assemblersprache oder PL/M.

Zusätzlich stehen die ISIS-Systemaufrufe dem Anwender zur Verfügung. Sollen Programme unter der Steuerung von ISIS II ablaufen, ist die Lage des Anwenderprogramms im Speicher von Bedeutung.

Den ISIS-II-Systemprogrammen steht für den Zugriff auf ASCII-Dateien eine besondere Pufferdatei zur Verfügung. Diese Datei heißt Zeilenaufbereitungsdatei (Line-edit-file). Sie erlaubt dem Anwender, Fehler bei der Eingabe über Tastatur zu korrigieren. Erst, wenn eine Zeile fehlerfrei ist, wird sie an das anfordernde Programm übertragen.

11.1. Speicherorganisation

Die Organisation des SME-Speichers unter ISIS II wird im folgenden Diagramm gezeigt:



64K (FFFFH)
62K (F8ØØH)

Oberstes Ende des Speichers 32K, 48K, 64K

Programm-Basisadresse ≧3180H
Oberstes Ende des Pufferbereichs ≧3180H

Pufferbereich-Basisadresse = 3ØØØH

Speicherstellen 24 bis 63 (18H bis 3FH)

Speicherstellen Ø bis 23 (ØH bis 17H)

Die Unterbrechungen Ø bis 2 sind für ISIS II und den Monitor reserviert. Die Unterbrechungen 3 bis 7 sind frei für den Anwender. Diese Speicherstellen 24 bis 63 sind die einzigen Speicherstellen unterhalb von 12K (3ØØØH), die dem Anwender zur freien Verfügung stehen. Der residente ISIS-II-Bereich ist für den Teil von ISIS II reserviert, der ständig im RAM-Speicher steht. In diesen residenten Speicher kann kein Anwenderprogramm geladen werden (Ladeschutz).

Es ist aber möglich, daß ein ablaufendes Programm infolge fehlerhafter Adressierung auf diesen geschützten Bereich zugreift und Teile der Systemprogramme des Betriebssystems überschreibt. Werden nun über die Systemaufrufe diese Systemprogramme aufgerufen, entstehen unvorhergesehene Fehler.

Der Pufferbereich wird von ISIS II für Ein-/Ausgabepuffer mit je 128 Bytes verwendet. Ein ständiger Puffer wird von ISIS II für die Eingabe über das Bediengerät verwendet. Die anderen Puffer werden von ISIS II dynamisch, entsprechend den Ein-/Ausgabeaufrufen des Anwenderprogramms, zugewiesen oder freigegeben. Als Minimalgröße für den Pufferbereich sind drei Puffer zugelassen, einschließlich des ständigen ISIS-II-Puffers. Falls vom Programm mehr als zwei Puffer gefordert werden, vergrößert sich der Pufferbereich auf Kosten des freien Bereichs.

Der Programmbereich liegt oberhalb des Pufferbereichs. Die Programm-Basisadresse wird vom Programmierer bestimmt oder durch LOCATE zugewiesen oder mit der ORG-Anweisung absoluter 8080-Assemblerprogramme festgelegt.

Der Kommandointerpreter des ISIS II, andere nichtresidente ISIS-II-Routinen (alle Kommandos außer DEBUG), der Texteditor, der Assembler, der PL/M-Übersetzer sowie die Programme LINK, LOCATE und LIB laufen im Programmbereich ab. Wenn man mit Hilfe von Kommandos mit ISIS II verkehrt, geschieht dies immer mit dem nichtresidenten Kommandointerpreter, der im Programmbereich abläuft.

Nichtresidente ISIS-II-Routinen können den ganzen RAM-Speicher für Puffer verwenden. Deshalb sollten Anwenderprogramme, die ständig resident sein müssen, in einem ROM-Speicher oder in einem RAM-Speicher, der unabhängig vom ersten zusammenhängenden RAM-Speicherblock ist, untergebracht sein.

Die ersten 32 KB des Speichers müssen Arbeitsspeicher (RAM) sein. Oberhalb 32 KB des Speichers ist jede Kombination von ROM- und RAM-Speichern möglich. Die Monitorroutine MEMCHK wird aufgerufen, wenn ein Anwenderprogramm die höchste Stelle des zusammenhängenden RAM-Speichers wissen muß.

Der Monitor belegt die letzten 2 K des Speichers. Wenn die Speicherkonfiguration in den Adreßbereich F8ØØH bis FFFFH hineinreicht, können diese Adressen vom Anwender nicht benutzt werden, weil hier der ROM-Speicher des Monitors liegt.

11.2. Bestimmen des Programm-Anfangpunktes

Der unter ISIS II maximal für E/A-Puffer benötigte Bereich muß berücksichtigt werden, wenn die Anfangsadresse des Programms bestimmt wird (siehe Kapitel 7 über LOCATE-Operation). Die Pufferanzahl variiert dynamisch, doch muß der Pufferbereich so groß sein, wie es die maximale Anzahl des gleichzeitig belegten Puffers erfordert. Ist der Programmanfangspunkt zu niedrig, meldet ISIS II einen »nichtbehebbaren Fehler«, sobald das Programm einen weiteren Puffer benötigt.

Die Programmbasisadresse kann durch Verwendung folgender Formel berechnet werden,
12K + (128*N)

wobei N die maximale Anzahl der vom Programm zur gleichen Zeit benötigten Puffer ist. Um N zu bestimmen, sind folgende Regeln zu verwenden:

1. Jede eröffnete Diskettendatei benötigt solange zwei Puffer, bis die Datei geschlossen wird.

2. Eine eröffnete zeilenaufbereitete Datei benötigt solange einen Puffer, bis die Datei geschlossen ist.
3. Ein Systemaufruf, der auf das Diskettenin haltsverzeichnis zugreift (LOAD, DELETE, RENAME, ATTRIB und CONSOL), benötigt zwei Puffer während seiner Verarbeitung. Die Puffer werden bei der Rückkehr ins rufende Programm wieder freigegeben.
4. Wird dem CONSOL-Systemaufruf das Bedienungs-Eingabegerät oder -Ausgabegerät einer Diskettendatei zugewiesen, werden für jede Datei zwei Puffer benötigt. Diese Puffer werden bis zum Auftreten des Dateiendes benötigt. Ein Programm, das durch ein Systemkommando in einer SUBMIT-Datei aufgerufen wird, muß beim Bestimmen seines Anfangspunktes diese Puffer ebenfalls berücksichtigen.

Beispiel 1:
Ein Programm, das keine Systemaufrufe enthält, bei dem das Bediengerät keiner Diskettendatei zugewiesen ist und das nicht mit einem Kommando in einer Diskettendatei aufgerufen wird, benötigt drei Puffer; es hat die Programmanfangsadresse von 3180H. Wird das Programm dahingehend geändert, daß es eine Diskettendatei eröffnet, benötigt es fünf Puffer. Die Programmanfangsadresse muß dann 3280H sein: 3000H + (128 × 5) = 3280H. Will man die Möglichkeit absichern, das Programm durch eine SUBMIT-Datei aufzurufen, worin das Bedienungs-Ausgabegerät ebenfalls eine Diskettendatei ist, werden weitere vier Puffer benötigt. Dadurch wird eine Adresse von 3480H erforderlich: 3000H + (128 + 9) = 3480H.

Beispiel 2
Angenommen, ein Programm eröffnet eine zeilenaufbereitete Diskettendatei (3 Puffer) sowie eine Kontrollausgabedatei auf Diskette (2 Puffer) und ändert die Attribute einer Diskettendatei mit dem ATTRIB-Systemaufruf, ergibt sich folgender Programmanfangspunkt (Basisadresse): 3000H + (128 * 8) = 3400H.

Hinweis
Soll ein Programm unabhängig vom Gerätetyp, der für die Datenübertragung verwendet wird, sein und auch unabhängig sein, wie es aufgerufen wird (vom Bediengerät aus oder durch eine SUBMIT-Datei), dann sollte man die maximale Pufferanzahl, die es benötigen könnte, berücksichtigen. Dies bedeutet, daß man für jede offene Datei zwei Puffer reservieren würde, ob es sich um eine Diskettendatei handelt oder nicht. Man würde ferner fünf Puffer für die Bedienungs-Eingabe- und Ausgabedateien reservieren.

11.3. Zeilenaufbereitete Eingabedatei

ISIS II bietet eine besondere Art des Lesens von ASCII-Dateien. Dieses Verfahren der »Zeilenaufbereitung« ist als Hilfe für den Anwender gedacht, der Daten über die Tastatur eingibt und dabei Fehler machen kann. Die Löschtaste und die Steuerzeichen erlauben die Korrektur von Eingabefehlern, um dann eine fehlerfreie Zeile nach der Eingabe des Wagenrücklaufzeichens, wobei automatisch ein Zeilenvorschubzeichen angefügt wird, zu übertragen. ISIS II wird durch einen Parameter im OPEN-Systemaufruf davon unterrichtet, daß eine Datei zeilenaufbereitet sein soll. Zugeordnet zu jeder zeilenaufbereiteten Datei gehört eine Kontrolldatei, in die die eingegebenen Daten für Modifizierungszwecke übertragen werden. Diese Kontrolldatei muß vor der zeilenaufbereiteten Datei eröffnet werden. Wird keine Kontrolldatei gewünscht, kann die Leerdatendatei :BB: als Kontrolldatei festgelegt werden.

11.3.1. Beenden einer Zeile

Die vom Anwender über ein Eingabegerät eingegebene Zeile wird von ISIS in einem 122 Zeichen langen Puffer gespeichert. Der Pufferinhalt wird von ISIS II geändert, wenn ein Aufbereitungszeichen eingegeben wird. Dazu wird auf die Erklärung der Aufbereitungszeichen weiter unten verwiesen. Die Daten werden an das anfordernde Programm erst dann übertragen, wenn die Zeile in einer von drei Arten beendet worden ist:

1. Nach der Eingabe von ›Wagenrücklauf‹ wird ein ›Zeilenvorschub‹ automatisch angehängt.
2. Das Übertragungskriterium ist der Zeilenvorschub. Es ist ein ESC-Zeichen eingegeben worden.
3. Es ist ein nicht-zeilenaufbereitendes Zeichen als 122stes Zeichen eingegeben worden.

11.3.2. Lesen aus dem Zeilenpuffer

Wird das Endkriterium erkannt, überträgt ein READ-Systemaufruf die Daten vom Zeilenpuffer zum Puffer des anfordernden Programms. Ein ISIS-interner Zeiger sorgt dafür, daß die Zeichen stellengerecht übertragen werden. Enthält z. B. der Zeilenaufbereitungspuffer 1ØØ Zeichen, aber in dem READ-Systemaufruf ist der Parameter COUNT auf 5Ø gesetzt, werden nur die ersten 5Ø Zeichen vom Zeilenpuffer in den Programmpuffer übertragen und der Pufferzeiger auf das 51ste Zeichen im Zeilenpuffer gesetzt. Mit dem nächsten READ-Systemaufruf werden die restlichen 5Ø Zeichen übertragen.

Fordert der READ-Systemaufruf 1ØØ Bytes an und der Zeilenaufbereitungspuffer enthält nur 5Ø Bytes, werden nur diese 5Ø Bytes übertragen.

Wenn alle Zeichen im Zeilenaufbereitungspuffer gelesen sind, steht der Pufferzeiger hinter dem letzten Zeichen. Der Puffer selbst wird dabei nicht gelöscht. Mit dem RESCAN-Systemaufruf kann die Position des Zeigers im Zeilenpuffer wieder auf den Anfang des Puffers gesetzt werden, um mit einem erneuten READ-Systemaufruf den Pufferinhalt erneut zu lesen.

11.3.3. Aufbereitungszeichen

Die folgenden Zeichen werden verwendet, um die Eingabe eines Zeilenpuffers aufzubereiten. Steuerzeichen werden eingegeben, indem die Steuertaste (CTRL) gedrückt wird, während das Zeichen eingegeben wird.

Rubout Löschen des zuletzt eingegebenen Zeichens. Das gelöschte Zeichen wird zur Kontrolle nochmals ausgegeben. Sollen mehrere Zeichen gelöscht werden, ist dies durch wiederholtes Drücken dieser Taste möglich. Sind alle Zeichen der Zeile gelöscht, wird dies mit einem Klingelzeichen mitgeteilt.

CTRL-P Dieses Steuerzeichen bewirkt, daß das nächste eingegebene Zeichen mit seinem Zeichenwert in den Zeilenpuffer eingetragen wird. Das Steuerzeichen CTRL-P wird benutzt, um ein Aufbereitungs- oder ein Endezeichen im Puffer abzuspeichern, ohne daß es wirksam wird.

CTRL-R Mit der Eingabe dieses Steuerzeichens wird der augenblicklich gültige Text ausgegeben. Dies ist vorteilhaft, wenn durch mehrere Korrekturen die Übersichtlichkeit des eingegebenen Textes beeinträchtigt ist.

CTRL-X Dieses Steuerzeichen bewirkt, daß die zuletzt eingebene Zeile gelöscht wird. Als Quittung antwortet ISIS II mit dem Zeichen #, Wagenrücklauf und Zeilenvorschub.

CTRL-Z Dieses Steuerzeichen ist der einzige Weg, Dateiende über ein Tastatureingabegerät anzuzeigen. Es arbeitet ähnlich wie CTRL-X, aber es gibt hier keine Kontrollausgabe. Mit CTRL-Z wird bewirkt, daß über READ-Systemaufruf unverzüglich eine Dateiendekennung (EOF) simuliert wird. Falls nach dem CTRL-Z weitere Zeichen eingegeben werden, werden diese in den Zeilenpuffer eingetragen und können durch einen anschließenden READ-Systemaufruf gelesen werden.

11.3.4. Lesen einer Kommandozeile

Einen Spezialfall der Zeilenaufbereitung stellt das Lesen einer Kommandozeile vom Konsolgerät dar (siehe Abschnitt 4.3.7., Ausführen eines Programms in einer Diskettendatei).

Wenn am Bediengerät ein Kommando eingegeben wird, wird es von ISIS II im Aufbereitungspuffer von :CI: gesammelt. Es ist erst für den Kommandoübersetzer (eine nichtresidente ISIS-II-Routine) verfügbar, wenn es beendet ist. Der Kommandointerpreter liest den Kommandonamen und ruft dann das Programm mit diesem Namen auf. Dabei bleibt der Zeiger des Aufbereitungspuffers hinter dem Kommandonamen stehen. Das geladene Programm kann einen READ-Systemaufruf geben, der die Daten – beginnend beim ersten Parameter – überträgt oder es kann einen RESCAN-Systemaufruf geben, der den Pufferzeiger an den Pufferanfang setzt, so daß das Programm den Kommandonamen lesen kann.

Als Beispiel werde angenommen, daß folgendes Kommando eingegeben worden ist:

–TYPE :F1:PROG.SRC

Der Zeilenaufbereitungspuffer für :CI: enthält die folgenden 2Ø Zeichen (das CR = Wagenrücklauf, LF = Zeilenvorschub):

T	Y	P	E		:	F	1	:	P	R	O	G	A	.	S	R	C	CR	LF
1	2	3	4	5	6	7	8	9	1Ø	11	12	13	14	15	16	17	18	19	2Ø

Wenn das TYPE-Programm geladen ist, steht der Zeiger im Zeilenpuffer auf dem fünften Zeichen (dem Zwischenraum hinter TYPE). Ein READ-Systemaufruf beginnt mit der Zeichenübertragung beim fünften Zeichen. Eine neue Eingabe in den Zeilenaufbereitungspuffer kann erst durchgeführt werden, wenn durch einen Systemaufruf READ der Pufferzeiger im Puffer auf das Ende der Eingabe (hinter CR, LF) zeigt.

Es ist zu beachten, daß bei der Übergabe der Steuerung an das geladene Programm der Pufferzeiger hinter dem Kommandonamen steht, und nicht am Pufferende. Dies bedeutet, daß – falls keine Parameter übergeben werden – der erste READ-Aufruf über :CI: die Zeichen Wagenrücklauf und Zeilenvorschub zurückgibt, die von der Kommandozeile übriggeblieben sind.

Wird z.B. folgendes Kommando über :CI: eingegeben,
–PROGA.BIN
stehen im Zeilenpuffer folgende 11 Zeichen:

P	R	O	G	A	.	B	I	N	CR	LF
1	2	3	4	5	6	7	8	9	1Ø	11

Ist PROGA.BIN geladen, steht der Pufferzeiger beim Zeichen Wagerücklauf. Falls anschließend Eingabe vom Bedienungseingabegerät erwartet wird, muß zuerst mit dem READ-Aufruf der Puffer (CR, LF) gelöscht werden.

Werden die programmspezifischen Parameter nicht über :CI: eingelesen, wird vor dem Aufruf des nächsten Kommandos der Zeiger im Puffer wieder auf Anfang gesetzt.

12. ISIS-II-Systemaufrufe

Die ISIS-II-Systemaufrufe, die von einem Anwenderprogramm aufgerufen werden können, sind Bestandteil des ISIS-II-Betriebssystems und umfassen folgendes:

1. Ein-/Ausgabeoperationen für die Diskette und die standardmäßigen Peripheriegeräte, mit Ausnahme des Universellen PROM-Programmiergerätes (UPP).
2. Verwalten des Disketteninhaltsverzeichnisses (ATTRIB, DELETE, RENAME).
3. Zuweisen der Konsole und Ausgabe von Fehlermeldungen (CONSOL, WHOCON, ERROR).
4. Laden und Ausführen von Programmen sowie Rückgabe der Steuerung an das System (LOAD, EXIT).

Die Schnittstelle zu diesen ISIS-II-Funktionen ist ein Aufruf an das ISIS-II-Organisationsprogramm, der die gewünschte Funktion und die Adresse der Parameterliste festlegt, auf die das Organisationsprogramm zugreifen soll. Die speziellen Aufruffolgen sind in den Kapiteln 13 und 14 beschrieben. Zu beachten ist, daß die ISIS-II-Systemaufrufe einen eigenen Stack verwenden und die Tiefe des Programm-Stacks durch den Aufruf an das Organisationsprogramm nicht geändert wird. Ein Aufruf an ISIS II zerstört den Inhalt der Prozessor-Register.

Die verfügbaren Systemaufrufe werden in diesem Abschnitt beschrieben, und zwar hinsichtlich ihrer Arbeitsweise und der Parameter, die zu ihrer ordnungsgemäßen Abarbeitung vom Programm geliefert werden müssen. Die Parameter wurden wegen der besseren Lesbarkeit mit symbolischen Namen versehen.

12.1. Länge (LENGTH) und Zeiger (MARKER)

Um die Auswirkung bestimmter Systemaufrufe für die Dateien darzustellen, werden jeder Datei zwei ganzzahlige Größen, nämlich LENGTH und MARKER zugeordnet. LENGTH ist Anzahl der Bytes in einer Datei. MARKER gibt die Anzahl der Bytes an, die bereits in die Datei geschrieben oder aus ihr gelesen worden sind (MARKER arbeitet also wie ein Dateizeiger).

12.2. Datei-Ein-/Ausgabe

Einem Programm stehen sechs Systemaufrufe zur Steuerung der Datei-Ein-/Ausgabe zur Verfügung. Mit diesen Unterprogrammen können Dateien für Lese- und Schreiboperationen eröffnet, der Zeiger in einer eröffneten Datei verschoben und die Dateien, wenn die Verarbeitung beendet ist, wieder geschlossen werden.

Diese Datenübertragungsdienste des Organisationsprogramms übertragen Datenblöcke variabler Länge zwischen den Standard-Peripheriegeräten und einem Pufferspeicherbereich im Programm. Zusätzlich zu dem Datenübertragungspuffer im Programmbereich erfordert das ISIS-II-Betriebssystem zwei 128-Byte-Puffer für jede eröffnete Diskettendatei. Dieser Puffer ist im Pufferbereich des Betriebssystems angeordnet, wie in Kapitel 11 beschrieben.

12.2.1. OPEN-Aufruf (Initiieren einer Datei für Ein-/Ausgabeoperationen)

Allgemeines
Der OPEN-Systemaufruf initiiert die Systemtabellen und weist für die Ein-/Ausgabeverarbeitung einer bestimmten Datei die erforderlichen Puffer zu. Falls die zu eröffnende Datei ein

Lochstreifenstanzer ist (:HP: oder :TP:), werden etwa 30 Zentimeter Vorlauf gestanzt (ASCII-Nilzeichen).

Systemeinsprung
CALL ISIS

Code für OPEN
= Ø

Registerversorgung
Reg C = Code für OPEN, also Ø
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTNPTR	Adresse für die Rückmeldung der Tabellennummer (AFTN) der angesprochenen Datei.
FILEPTR	Adresse der ASCII-Zeichenfolge, die den Namen der zu eröffnenden Datei festlegt.
ACCESS	Wert 1, 2 oder 3 für Eingabe-, Ausgabe- oder Änderungsmodus.
ECHOAFTN	Tabellennummer der Kontrolldatei, falls Zeilenaufbereitung gewünscht wird, andernfalls Null.
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird.

Beschreibung der Parameter

AFTNPTR

Der Parameter AFNPTR enthält die Adresse, in der die Tabellennummer der eröffneten Datei gespeichert wird. Das System führt eine Tabelle, in der die eröffneten Dateien vermerkt sind. Jeder eröffneten Datei wird vom Betriebssystem eine Nummer, die sogenannte AFTN-Tabellennummer (AFTN = aktiv file table number) zugewiesen, die dem Anwenderprogramm übergeben wird. Diese Nummer dient zur Identifizierung der eröffneten Datei bei späteren Systemaufrufen.

Die Tabellennummern Ø und 1 sind den Dateien :CO: und :CI: fest zugeordnet und sind immer eröffnet.

Es dürfen nie mehr als sechs Dateien zusätzlich zu :CO: und :CI: gleichzeitig eröffnet sein.

FILEPTR

Der Parameter FILEPTR enthält die Adresse einer ASCII-Zeichenfolge, die den Namen der zu eröffnenden Datei angibt. Die Zeichenfolge darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf der Name der zu eröffnenden Datei nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden. Falls die Zeichenfolge :CI: oder :CO: festlegt – diese Dateien sind immer eröffnet –, geschieht nichts, außer daß die zugehörige Tabellennummer Ø bzw. 1 an die Speicherstelle zurückgegeben wird, die durch den AFTNPTR-Parameter festgelegt ist.

ACCESS

Der Wert des Parameters legt die Zugriffsart auf die eröffnete Datei fest.
Folgende Werte sind für ACCESS möglich:

ACCESS = 1 die Datei wird für Eingabe eröffnet (READ), d. h. es können Daten aus einer Datei gelesen werden. Der Zeiger (MARKER) wird auf Ø gesetzt und die Länge (LENGTH) bleibt unverändert. Ist die angegebene Datei nicht vorhanden, wird ein behebbarer Fehler gemeldet.

ACCESS = 2 die Datei wird für Ausgabe eröffnet (WRITE), d. h. es können Daten in eine Datei geschrieben werden. Der Zeiger (MARKER) und die Länge (LENGTH) werden auf Ø gesetzt. Ist die angegebene Datei nicht vorhanden, wird eine Datei mit dem Namen wie unter FILEPTR angegeben, eingerichtet. Alle Attribute werden rückgesetzt. Wird eine Diskettendatei angegeben, deren Attribute ›Schreibschutz‹ und/oder ›Format‹ gesetzt sind, wird ein behebbarer Fehler gemeldet.

ACCESS = 3 die Datei wird für Aktualisierung (UPDATE) eröffnet (READ und WRITE). Der Zeiger (MARKER) wird auf Ø gesetzt. Ist die Datei schon vorhanden, wird der Wert von Länge (LENGTH) nicht verändert. Wenn die Datei noch nicht vorhanden ist, wird eine Datei mit dem angegebenen Dateinamen eingerichtet. Die Attribute ›Schreibschutz‹ und ›Format‹ werden in diesem Fall zurückgesetzt und die Länge (LENGTH) auf Ø gesetzt. Die Eröffnung einer Datei für einen physisch nicht möglichen Zugriffsmodus bewirkt einen behebbaren Fehler. Z. B. bewirkt das Öffnen von :HP: (schneller Lochstreifenstanzer) einen Fehler.

ECHOAFTN

Auf eine Eingabedatei kann mit dem Zeilenaufbereiter zugegriffen werden, indem die Tabellennummer der Kontrolldatei im niederwertigen Byte des Parameters ECHOAFTN abgelegt wird. Die Kontrolldatei muß bereits zur Ausgabe eröffnet sein. Wenn der Parameter ECHOAFTN eine Ø enthält, unterbleibt die Zeilenaufbereitung. Ist die Tabellennummer der Datei = Ø (für :CO:), muß das höherwertige Byte von ECHOAFTN ungleich Ø und das niederwertige Byte gleich Ø sein. Beispielsweise kann 256 (1ØØH) für den Parameter ECHOAFTN eingegeben werden, wenn :CO: die Kontrolldatei ist.

STATUS

Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf OPEN erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
1	Pufferbereich für den benötigten Puffer zu klein (nicht behebbarer Fehler)
3	Versuch, mehr als 6 Dateien gleichzeitig zu eröffnen
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät im Dateinamen ist unzulässig oder dem System nicht bekannt
7	Operation abgebrochen, weil zuwenig Platz auf der Diskette ist (nicht behebbarer Fehler)

9	Disketteninhaltsverzeichnisbereich zu klein (nicht behebbarer Fehler)
12	Versuch, eine bereits eröffnete Datei zu eröffnen
13	Die angegebene Datei ist im Disketteninhaltsverzeichnis nicht eingetragen
14	Versuch, eine Datei mit dem gesetzten Attribut »Schreibschutz« zum Schreiben zu eröffnen
22	Der Wert im ACCESS-Parameter ist unzulässig (formal oder für die Datei nicht möglich, z.B. Eingabedatei mit :LP:)
23	Für eine Diskettendatei ist kein Dateiname angegeben
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
25	Falsche Angaben für die Kontrolldatei
28	Keine Erweiterung des Dateinamens vorhanden
3Ø	Floppy-Disk unklar (nicht behebbarer Fehler)

Beispiel

```
;
;  ****************************************
;
;  SYSTEMKONSTANTEN
        EXTRN    ISIS
OPEN    EQU      Ø

;
; OPEN ROUTINE
;
       MVI   C,OPEN         ; CODE FUER OPEN LADEN
       LXI   D,OBLK         ; ADR. DER PARAMETERLISTE LADEN
       CALL  ISIS           ; SYSTEMAUFRUF ISIS
       LDA   OSTAT          ; STATUS-BYTE AUF FEHLER PRUEFEN
       ORA   A
       JNZ   EXCEPT         ; FEHLERBEHANDLUNG
         .
         .
         .
         .
;
;  PARAMETERLISTE FUER DEN SYSTEMAUFRUF OPEN
```

```
;
OBLK:
        DW      OAFT            ; ADR.,UNTER DER DIE TABELLEN-
                                ; NUMMER ABGESPEICHERT WIRD
        DW      OFILE           ; ADR.,UNTER DER DER DATEINAME
                                ; STEHT
ACCESS:DW       1               ; ZUGRIFFSART
ECHO:   DW      Ø               ; TABELLENNUMMER DER KONTROLL-
                                ; DATEI
        DW      OSTAT           ; ADR. FEHLERNUMMER
;
OAFT:   DS      2               ; TABELLENNUMMER (RUECKGEMELDET)
OSTAT:  DS      2               ; FEHLERNUMMER (RUECKGEMELDET)
OFILE:  DB      ':F2:FILE.EXT'; NAME DER DATEI
```

12.2.2. READ-Aufruf (Datenübertragung von einer Datei zum Speicher)

Allgemeines
Der Systemaufruf READ überträgt Daten von einer eröffneten Datei in einen vom aufrufenden Programm festgelegten Speicherbereich. Dieser Bereich wird Puffer genannt. Für das Lesen von zeilenaufbereiteten Dateien gelten die Regeln wie im Kapitel 11 ›zeilenaufbereitete Eingabedateien‹ beschrieben.

Systemeinsprung
CALL ISIS

Code für READ
= 3

Registerversorgung
Reg C = Code für READ, also 3
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTN	Tabellennummer der eröffneten Datei, aus der gelesen oder in der geändert werden soll
BUFFER	Adresse der Speicherstelle, in die die gelesenen Daten übertragen werden sollen
COUNT	Anzahl der zu übertragenden Bytes
ACTUAL	Adresse der Speicherstelle, in die die Anzahl der tatsächlich übertragenen Bytes zurückgemeldet wird
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers zurückgemeldet wird.

Beschreibung der Parameter

AFTN
Der Parameter AFTN enthält die Tabellennummer einer für Eingabe eröffneten Datei. Die Tabellennummer wird durch einen vorhergehenden Systemaufruf OPEN bestimmt oder ist 1 für :CI:.

BUFFER

Der Parameter BUFFER enthält die Adresse einer Speicherstelle, in die die gelesenen Daten übertragen werden. Der Puffer sollte mindestens so groß sein wie in COUNT angegeben. Anderenfalls werden Speicherstellen, die auf den Puffer folgen, überschrieben.

COUNT

Die Anzahl der von der Datei in den Puffer zu übertragenden Bytes wird im Parameter COUNT festgelegt.

ACTUAL

Der Parameter ACTUAL enthält die Adresse einer Speicherstelle, in die die Anzahl der tatsächlich gelesenen Bytes als Rückmeldung eingetragen wird. Die Anzahl wird zum Stand des Zeigers (MARKER) addiert.
Die tatsächliche Anzahl übertragener Bytes ist niemals größer als der Wert des COUNT-Parameters. Bei zeilenaufbereiteten Dateien ist die Anzahl der tatsächlich übertragenen Bytes niemals größer als die Byteanzahl im Zeilenaufbereitungspuffer (siehe Kapitel 11). Wenn eine Datei nicht zeilenaufbereitet ist, ist die Anzahl der tatsächlich übertragenen Bytes entweder gleich dem Wert von COUNT oder gleich Länge (LENGTH) minus Zeiger (MARKER), je nachdem, was kleiner ist.
Dateiende wird erkannt, wenn die Anzahl der gelesenen Bytes kleiner als COUNT ist (außer bei zeilenaufbereiteten Dateien) oder wenn die Anzahl der gelesenen Bytes = Ø ist.

STATUS

Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf READ erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
2	Die Tabellennummer im AFTN-Parameter bezeichnet eine Datei, die nicht eröffnet ist.
8	Versuch, aus einer für Ausgabe eröffneten Datei zu lesen
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
3Ø	Floppy-Disk unklar (nicht behebbarer Fehler)

Beispiel

```
;
; ****************************************
;
; SYSTEMKONSTANTEN
          EXTRN    ISIS
READ      EQU      3
;
; READ ROUTINE
;
       MVI    C,READ      ; CODE FUER READ LADEN
       LXI    D,RBLK      ; ADR. DER PARAMETERLISTE LADEN
       CALL   ISIS        ; SYSTEMAUFRUF ISIS
       LDA    RSTAT       ; STATUS-BYTE AUF FEHLER PRUEFEN
       ORA    A
       JNZ    EXCEPT      ; FEHLERBEHANDLUNG

         .
         .
         .
         .

;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF READ
;
RBLK:
RAFT:  DS     2           ; TABELLENNUMMER DER DATEI, DIE GE-
                           ; LESEN WERDEN SOLL
       DW     RBUF        ; ADR. DES EINGABEPUFFERS
RCNT:  DW     128         ; ANZAHL DER ZU LESENDEN BYTES
       DW     RACT        ; ADR.DER ANZAHL DER GELESENEN BYTES
       DW     RSTAT       ; ADR. DER FEHLERNUMMER
;
RACT:  DS     2           ; ANZAHL DER GELESENEN BYTES
                           ; (RUECKGEMELDET)
RSTAT: DS     2           ; FEHLERNUMMER (RUECKGEMELDET)
RBUF:  DS     128         ; EINGABEPUFFER
```

12.2.3. WRITE-Aufruf (Datenübertragung vom Speicher zu einer Datei)

Allgemeines
Der Systemaufruf WRITE überträgt Daten von einem angegebenen Anwenderspeicher, Puffer genannt, in eine eröffnete Datei.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für WRITE
= 4

Registerversorgung
Reg C = Code für WRITE, also 4
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTN	Tabellennummer der eröffneten Datei, in die geschrieben oder in der geändert werden soll.
BUFFER	Adresse des Speicherfeldes, aus dem die Daten zu übertragen sind
COUNT	Anzahl der zu übertragenden Bytes
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers zurückgemeldet wird.

Beschreibung der Parameter

AFTN
Der Parameter AFTN enthält die Tabellennummer einer für Ausgabe eröffneten Datei. Die Tabellennummer wird durch einen vorhergehenden Systemaufruf OPEN bestimmt oder ist Ø für :CO:

BUFFER
Der Parameter BUFFER enthält die Adresse eines Speicherfeldes, Puffer genannt, aus dem die Daten ausgegeben werden sollen.

COUNT
Der Parameter COUNT gibt die Anzahl der Bytes an, die vom Puffer in die Ausgabedatei übertragen werden sollen. Der Wert von Count wird zum Stand des Zeigers (MARKER) addiert. Wird der Stand des Zeigers (MARKER) durch die Ausgabeoperation größer als die Länge (LENGTH) der Datei, wird die Länge (LENGTH) dem Zeigerstand gleichgesetzt. Die Anzahl der tatsächlich übertragenen Bytes ist gleich dem Wert von COUNT. Ist der Puffer kleiner als der Wert von COUNT, werden die Bytes hinter dem Puffer mit in die Ausgabedatei übertragen.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf WRITE erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
2	Die Tabellennummer im AFTN-Parameter bezeichnet eine Datei, die nicht eröffnet ist
6	Versuch, in eine für Eingabe eröffnete Datei zu schreiben.
7	Operation abgebrochen, da zu wenig Platz auf der Diskette (nicht behebbarer Fehler)
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
3Ø	Floppy-Disk unklar (nicht behebbarer Fehler)

Beispiel

```
;
; ******************************************
;
; SYSTEMKONSTANTEN
        EXTRN   ISIS
WRITE   EQU     4
;
; WRITE ROUTINE
;
        MVI     C,WRITE     ; CODE FUER WRITE LADEN
        LXI     D,WBLK      ; ADR. DER PARAMETERLISTE LADEN
        CALL    ISIS        ; SYSTEMAUFRUF ISIS
        LDA     WSTAT       ; STATUSBYTE AUF FEHLER PRUEFEN
        ORA     A
        JNZ     EXCEPT      ; FEHLERBEHANDLUNG
          .
          .
          .
          .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF WRITE
;
WBLK:
```

```
WAFT:   DS      2           ; TABELLENNUMMER DER DATEI, IN
                            ; DIE GESCHRIEBEN WERDEN SOLL
        DW      WBUF        ; ADR. DES AUSGABEPUFFERS
WCNT:   DW      128         ; ANZAHL DER ZU SCHREIBENDEN
                            ; BYTES
        DW      WSTAT       ; ADR. DER FEHLERNUMMER
;
WSTAT:  DS      2           ; FEHLERNUMMER (RUECKGEMELDET)
WBUF:   DS      128         ; AUSGABEPUFFER
```

12.2.4. SEEK-Aufruf (Positionieren des Zeigers – MARKER – innerhalb der Datei)

Allgemeines
Mit dem Systemaufruf SEEK kann im Anwenderprogramm der Zeiger (MARKER) in einer für Eingabe oder Ändern eröffneten Diskettendatei manipuliert oder der aktuelle Stand des Zeigers (MARKER) festgestellt werden. Vier Modifikationsmöglichkeiten des Zeigers stehen zur Verfügung, nämlich

Schieben vorwärts
Schieben rückwärts
Schieben an eine bestimmte Stelle
Schieben zum Dateiende

Systemeinsprungstelle in ISIS
CALL ISIS

Code für SEEK
= 5

Registerversorgung
Reg C = Code für SEEK, also 5
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTN	Tabellennummer der eröffneten Datei, deren Zeiger (MARKER) manipuliert werden soll
MODE	Wert von Ø bis 4, der die auszuführende Operation angibt
BLOCKNO	Adresse einer Speicherstelle, unter der ein Wert steht, der zusammen mit BYTENO für die Berechnung von n benutzt wird
BYTENO	Adresse einer Speicherstelle, unter der ein Wert steht, der für die Berechnung von n benutzt wird.
STATUS	enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers zurückgemeldet wird.

Beschreibung der Parameter

AFTN
Der Parameter AFTN enthält die Tabellennummer einer für Eingabe oder Ändern eröffneten Datei. Die Tabellennummer wird durch einen vorhergehenden Systemaufruf OPEN bestimmt.

MODE

Der Parameter MODE muß einen der Werte von Ø bis 4 haben, andernfalls wird ein Fehler gemeldet.

MODE = Ø
Wird MODE auf Ø gesetzt, übergibt der Systemaufruf SEEK die Werte von BLOCKNO und BYTENO dem Programm, so daß dieses mit der aktuellen Stellung des Zeigers (MARKER) arbeiten kann.
Die Stellung des Zeigers (MARKER) kann mit folgender Formel berechnet werden:
Zeiger (MARKER) = 128 * (BLOCKNO) + BYTENO

MODE = 1
Wird MODE auf 1 gesetzt, wird der Zeiger (MARKER) um n Stellen rückwärts verschoben. Der Systemaufruf SEEK berechnet den Wert von n mit den Werten von BLOCKNO und BYTENO nach der Formel:
n = 128 * (BLOCKNO modulo 2**15) + BYTENO
Wenn sich dabei für die Stellung des Zeigers (MARKER) ein negativer Wert ergibt, wird der Zeiger (MARKER) auf Ø gesetzt und ein behebbarer Fehler gemeldet.

MODE = 2
Wird MODE auf 2 gesetzt, wird der Zeiger (MARKER) dem Wert n gleichgesetzt, wobei n nach folgender Formel berechnet wird:
n = 128 * (BLOCKNO modulo 2**15) + BYTENO
Ist n = Ø, wird der Zeiger (MARKER) auf Dateianfang gesetzt. Ist der neue Wert des Zeigers (MARKER) größer als die Länge (LENGTH), werden so viele ASCII-Nilzeichen (binäre Nullen) in der Datei angefügt, bis Länge (LENGTH) gleich dem Zeiger (MARKER) ist, vorausgesetzt, daß die Datei für Eingabe eröffnet ist. In diesem Falle tritt ein behebbarer Fehler auf.

MODE = 3
Wird MODE auf 3 gesetzt, wird der Zeiger (MARKER) um n Stellen vorwärts verschoben. Der Systemaufruf SEEK berechnet den Wert von n mit den Werten von BLOCKNO und BYTENO nach der folgenden Formel:
n = 128 * (BLOCKNO modulo 2**15) + BYTENO
Wenn der Wert des Zeigers (MARKER) größer als die Länge (LENGTH) ist, werden so viele ASCII-Nilzeichen (binäre Nullen) in der Datei angefügt, bis die Länge (LENGTH) gleich dem Wert des Zeigers (MARKER) ist, vorausgesetzt, daß die Datei für Eingabe eröffnet ist. In diesem Falle tritt ein behebbarer Fehler auf.

MODE = 4
Wird MODE auf 4 gesetzt, wird der Zeiger (MARKER) auf den Wert von Länge (LENGTH) geändert. Dies bewirkt, daß der Zeiger (MARKER) zum Dateiende verschoben wird. Die Werte von BLOCKNO und BYTENO haben dabei keinen Einfluß.

BLOCKNO

Der Parameter BLOCKNO enthält die Adresse einer Speicherstelle, unter der der aktuelle Wert des Blockes in der Datei steht. Ein Block entspricht, ebenso wie ein Sektor der Diskette, 128 Bytes. BLOCKNO wird zusammen mit BYTENO für die Berechnung von n zur Positionierung des Zeigers (MARKER) benötigt.

BYTENO

Der Paramter BYTENO enthält die Adresse einer Speicherstelle, unter der der aktuelle Wert des Bytes im Block steht. BYTENO wird zusammen mit BLOCKNO für die Berechnung von n zur Positionierung des Zeigers (MARKER) benötigt.

STATUS

Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf SEEK erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
2	Die Tabellennummer im AFTN-Parameter bezeichnet eine Datei, die nicht eröffnet ist
7	Operation abgebrochen, da zuwenig Platz auf der Diskette (nicht behebbarer Fehler)
19	Die angegebene Datei ist auf der Diskette nicht vorhanden
20	Versuch, den Zeiger über den Dateianfang hinaus zu positionieren
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
27	Falscher Wert im Parameter MODE
30	Floppy-Disk unklar (nicht behebbarer Fehler)
31	Die angegebene Datei ist für Ausgabe eröffnet
35	Versuch, eine für Eingabe eröffnete Datei über das Dateiende hinaus zu erweitern

Beispiel

```
;
; ***********************************************
;
; SYSTEMKONSTANTEN
        EXTRN   ISIS
SEEK    EQU     5
;
; SEEK ROUTINE
;
        MVI     C,SEEK  ; CODE FUER SEEK LADEN
        LXI     D,SBLK  ; ADR.DER PARAMETERLISTE LADEN
        CALL    ISIS    ; SYSTEMAUFRUF ISIS
        LDA     SSTAT   ; STATUSBYTE AUF FEHLER PRUEFEN
        ORA     A
        JNZ     EXCEPT  ; FEHLERBEHANDLUNG
          .
          .
          .
          .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF SEEK
;
SBLK:
SAFT:   DS      2       ; TABELLENNUMMER DER DATEI, DIE
                        ; FUER DEN AUFRUF SEEK EROEFFNET
                        ; WORDEN IST
MODE:   DS      2       ; AUSZUFUEHRENDE OPERATION
        DW      BLKS    ; ADR., IN DER DIE BLOCKANZAHL
                        ; STEHT
        DW      NBYTE   ; ADR., IN DER DIE BYTEANZAHL
                        ; STEHT
        DW      STAT    ; ADR. DER FEHLERNUMMER
;
BLKS:   DW      2       ; ANZAHL DER ZU UEBERSPRINGENDEN BLOECKE
NBYTE:  DW      2       ; ANZAHL DER ZU UEBERSPRINGENDEN BYTES
STAT:   DS      2       ; FEHLERNUMMER (RUECKGEMELDET)
```

12.2.5. RESCAN-Aufruf (Positionieren des Zeigers – MARKER – auf den Anfang des Zeilenaufbereitungspuffers)

Allgemeines
Der Systemaufruf RESCAN wird nur für zeilenaufbereitete Dateien verwendet. Mit dem Aufruf kann das Programm den Zeiger (MARKER) auf den Anfang einer schon gelesenen logischen Zeile setzen, so daß der nächste Systemaufruf READ am Anfang der zuletzt gelesenen logischen Zeile im Zeilenaufbereitungspuffer beginnt. Diese Zeile wird nicht in die Kontrolldatei übertragen, da sie über die Tastatur eingegeben wurde. Der nächste Systemaufruf READ liest nur Daten aus dem Zeilenaufbereitungspuffer des Zeilenaufbereiters.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für RESCAN
= 11

Registerversorgung
Reg C = Code für RESCAN, also 11
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTN	Tabellennummer einer für Zeilenaufbereitung eröffneten Datei
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers zurückgemeldet wird.

Beschreibung der Parameter

AFTN
Der Parameter AFTN enthält die Tabellennummer einer für Eingabe eröffneten Datei. Die Tabellennummer wird durch einen vorhergehenden Systemaufruf OPEN bestimmt oder = 1 (:CI:). Wenn die Datei keine zeilenaufbereitete Datei ist (ECHOAFTN = Ø), wird ein behebbarer Fehler gemeldet.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf RESCAN erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
2	Die Tabellennummer im AFTN-Parameter bezeichnet eine Datei, die nicht eröffnet ist
21	Versuch, eine nicht zeilenaufbereitete Datei zu bearbeiten

Beispiel

```
;
; ***************************************
;
; SYSTEMKONSTANTEN
        EXTRN    ISIS
RESCAN  EQU      11
;
; RESCAN ROUTINE
;
        MVI   C,RESCAN   ; CODE FUER RESCAN LADEN
        LXI   D,IBLK     ; ADR. DER PARAMETERLISTE LADEN
        CALL  ISIS       ; SYSTEMAUFRUF ISIS
        LDA   ISTAT      ; STATUSBYTE AUF FEHLER PRUEFEN
        ORA   A
        JNZ   EXCEPT     ; FEHLERBEHANDLUNG
          .
          .
          .
          .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF RESCAN
;
IBLK:
IAFT:   DS    2          ; TABELLENNUMMER DER DATEI, DIE
                         ; FUER DEN SYSTEMAUFRUF RESCAN
                         ; EROEFFNET WORDEN IST
        DW    ISTAT      ; ADR. DER FEHLERNUMMER
;
ISTAT:  DS    2          ; FEHLERNUMMER (RUECKGEMELDET)
```

12.2.6. CLOSE-Aufruf (Beenden einer Ein-/Ausgabeoperation)

Allgemeines
Der Systemaufruf CLOSE löscht die zu schließende Datei aus den Systemtabellen und gibt die durch den Systemaufruf OPEN zugeordneten Puffer frei. Sind alle Ein-/Ausgabeoperationen durchgeführt, sollten alle eröffneten Dateien geschlossen werden, um sie in einen definierten Ausgangszustand zu versetzen.

Wenn die zu schließende Datei ein Lochstreifenstanzer ist (:HP: oder :TP:), wird ein Nachspann von ≈ 30 cm Länge (ASCII-Nilzeichen) gestanzt.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für CLOSE
= 1

Registerversorgung
Reg C = Code für CLOSE, also 1
REG D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTN	Tabellennummer der eröffneten Datei
STATUS	enthält die Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers zurückgemeldet wird

Beschreibung der Parameter

AFTN
Der Parameter AFTN enthält die Tabellennummer der eröffneten Datei. Die Tabellennummer wurde durch einen vorhergehenden Systemaufruf OPEN bestimmt. Wenn die Tabellennummer eine der Dateien :CO: oder :CI: angibt, die immer eröffnet sind, wird keine Aktion durchgeführt und der STATUS = Ø an das Programm zurückgemeldet.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf CLOSE erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
2	Die Tabellennummer im AFTN-Parameter bezeichnet eine Datei, die nicht eröffnet ist.

Beispiel

```
;
; ****************************************
;
; SYSTEMKONSTANTEN
        EXTRN ISIS
CLOSE   EQU   1
;
; CLOSE ROUTINE
;
        MVI   C,CLOSE    ; CODE FUER CLOSE LADEN
        LXI   D,CBLK     ; ADRESSE DER PARAMETERLISTE
                         ; LADEN
        CALL  ISIS       ; SYSTEMAUFRUF ISIS
        LDA   CSTAT      ; STATUSBYTE AUF FEHLER PRUEFEN
        ORA   A
        JNZ   EXCEPT     ; FEHLERBEHANDLUNG
          .
          .
          .
          .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF CLOSE
;
CBLK:
CAFT:   DS    2          ; TABELLENNUMMER DER DATEI, DIE
                         ; GESCHLOSSEN WERDEN SOLL
        DW    CSTAT      ; ADR. DER FEHLERNUMMER
;
CSTAT:  DS    2          ; FEHLERNUMMER (RUECKGEMELDET)
```

12.3. Verwaltung des Disketten-Inhaltsverzeichnisses

Dem Anwender stehen drei Systemaufrufe zur Verfügung, um Informationen in einem Diskett-eninhaltsverzeichnis zu ändern. Diese Systemaufrufe erlauben es, eine Diskettendatei zu löschen, sie umzubenennen oder ihre Attribute zu ändern.

12.3.1. DELETE-Aufruf (Löschen einer Datei im Disketteninhaltsverzeichnis)

Allgemeines
Der Systemaufruf DELETE löscht eine Datei, d.h. den Namen der Datei, auf der Diskette und stellt den von ihr belegten Speicherplatz für Neuzuweisungen zur Verfügung.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für DELETE
= 2

Registerbelegung
Reg C = Code für DELETE, also 2
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

FILEPTR	Adresse einer ASCII-Zeichenfolge, die den Namen der zu löschenden Datei angibt
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird.

Beschreibung der Parameter

FILEPTR
Der Parameter FILEPTR enthält die Adresse einer ASCII-Zeichenfolge, die den Namen der zu löschenden Datei angibt. Die Datei darf nicht eröffnet sein. Die Zeichenfolge darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf der Name der zu löschenden Datei nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf DELETE erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
1	Pufferbereich für den benötigten Puffer zu klein (nicht behebbarer Fehler)
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät im Dateinamen ist unzulässig oder dem System nicht bekannt
13	Die angegebene Datei ist im Diskettenetikettbereich nicht eingetragen
14	Versuch, eine Datei mit dem gesetzten Attribut »Schreibschutz« zu löschen
17	Versuch, eine Datei zu löschen, die auf der Diskette nicht vorhanden ist
23	Für eine Diskettendatei ist kein Dateiname angegeben
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
28	Keine Erweiterung des Dateinamens vorhanden
3Ø	Floppy-Disk unklar (nicht behebbarer Fehler)
32	Versuch, eine eröffnete Datei zu löschen

Beispiel

```
;
; ************************************
;
; SYSTEMKONSTANTEN
        EXTRN ISIS
DELETE  EQU   2

;
; DELETE ROUTINE
;
        MVI   C,DELETE    ; CODE FUER DELETE LADEN
        LXI   D,DBLK      ; ADR. DER PARAMETERLISTE LADEN
        CALL  ISIS        ; SYSTEMAUFRUF ISIS
        LDA   DSTAT       ; STATUSBYTE AUF FEHLER PRUEFEN
        ORA   A
        JNZ   EXCEPT      ; FEHLERBEHANDLUNG
```

```
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF DELETE
;
DBLK:    DW    DFILE       ; ADR., UNTER DER DER NAME DER
                           ; ZU LOESCHENDEN DATEI STEHT
         DW    DSTAT       ; ADR. DER FEHLERNUMMER
;
DSTAT:   DS    2           ; FEHLERNUMMER (RUECKGEMELDET)
DFILE:   DB    'FILE.EXT'  ; NAME DER DATEI
```

12.3.2. RENAME-Aufruf (Ändern eines Diskettendateinamens)

Allgemeines
Mit dem Systemaufruf RENAME kann der Name einer Diskettendatei geändert werden. Für NEWFILEPTR und OLDFILEPTR muß dasselbe Gerät angegeben werden.

Systemeinsprungstelle in ISIS
CALL ISIS

CODE für RENAME
= 7

Registerversorgung
Reg C = Code für RENAME, also 7
REG D/E = Adresse der Parameterliste

Aufbau der Parameterliste

OLDFILEPTR	Adresse einer ASCII-Zeichenfolge, die den alten Dateinamen angibt
NEWFILEPTR	Adresse einer ASCII-Zeichenfolge, die den neuen Dateinamen angibt
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers zurückgemeldet wird.

Beschreibung der Parameter

OLDFILEPTR
Der Parameter OLDFILEPTR enthält die Adresse einer ASCII-Zeichenfolge, die den alten Dateinamen angibt. Die Zeichenfolge darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf der Name der Datei nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden.

NEWFILEPTR
Der Parameter NEWFILEPTR enthält die Adresse einer ASCII-Zeichenfolge, die den neuen Dateinamen angibt. Die Zeichenfolge ist in derselben Weise anzugeben, wie in OLDFILEPTR.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf RENAME erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
1	Pufferbereich für den benötigten Puffer zu klein (nicht behebbarer Fehler)
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät im Dateinamen ist unzulässig oder dem System nicht bekannt.
10	Die Geräteteile in den Dateinamen sind unterschiedlich
11	Eine Umbenennung des Dateinamens ist nicht möglich, weil der neue Name bereits vorhanden ist.
23	Für eine Diskettendatei ist kein Dateiname angegeben
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
28	Keine Erweiterung des Dateinamens vorhanden
30	Floppy-Disk unklar (nicht behebbarer Fehler)

Beispiel

```
;
; ****************************************
;
; SYSTEMKONSTANTEN
          EXTRN   ISIS
RENAME    EQU     7
;
; RENAME ROUTINE
;
          MVI     C,RENAME    ; CODE FUER RENAME LADEN
          LXI     D,NBLK      ; ADR. DER PARAMETERLISTE
                              ; LADEN
          CALL    ISIS        ; SYSTEMAUFRUF ISIS
```

```
          LDA     NSTAT       ; STATUSBYTE AUF FEHLER
                              ; PRUEFEN
          ORA     A
          JNZ     EXCEPT      ; FEHLERBEHANDLUNG
            .
            .
            .
            .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF RENAME
;
NBLK:
          DW      OFILE       ; ADR., UNTER DER DER ALTE
                              ; DATEINAME STEHT
          DW      NFILE       ; ADR., UNTER DER DER NEUE
                              ; DATEINAME STEHT
          DW      NSTAT       ; ADR. DER FEHLERNUMMER
;
NSTAT:    DS      2           ; FEHLERNUMMER (RUECKGEMELDET)
NFILE:    DB      'FILE.NEW'  ; NEUER NAME DER DATEI
                              ;
OFILE:    DB      'FILE.OLD'  ; ALTER NAME DER DATEI
```

12.3.3. ATTRIB-Aufruf (Ändern der Attribute einer Diskettendatei)

Allgemeines
Der Systemaufruf ATTRIB erlaubt einem Anwenderprogramm, ein Attribut einer Diskettendatei zu ändern. Für eine Erörterung der Dateiattribute wird auf Kapitel 3 »Disketteninhaltsverzeichnis« verwiesen.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für ATTRIB
= 1Ø

Registerversorgung
Reg C = Code für ATTRIB, also 1Ø
REG D/E = Adresse der Parameterliste

Aufbau der Parameterliste

FILEPTR	Adresse einer ASCII-Zeichenfolge, die den Namen der Datei angibt, deren Attribut geändert werden soll.
SWID	Kennung, welches Attribut geändert werden soll.
VALUE	Wert des Attributs
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers zurückgemeldet wird.

Beschreibung der Parameter

FILEPTR
Der Parameter FILEPTR enthält die Adresse einer ASCII-Zeichenfolge, die den Namen der Datei angibt, deren Attribut geändert werden soll. Die Zeichenfolge darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf der Dateiname nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden.

SWID
Der Wert des Parameters SWID bezeichnet das zu ändernde Attribut. Die Werte und deren Zuordnung zu den Attributen sind:

Ø = »Unsichtbar«
1 = »System«
2 = »Schreibschutz«
3 = »Format«

Ist das Attribut »Unsichtbar« gesetzt (Wert = 1), wird die Datei beim Aufruf des Kommandos DIR nicht aufgelistet, es sei denn, der Parameter I ist im DIR-Kommando angegeben.
Ist das Attribut »System« gesetzt (Wert = 1), wird die Datei auf der Diskette im Laufwerk Ø mit dem Kommando FORMAT, wenn der Parameter S angegeben ist, auf die Diskette im Laufwerk 1 kopiert.

Dateien, bei denen das Attribut »Schreibschutz« gesetzt ist, sind für Ausgabeoperationen oder Änderungen gesperrt. Sie können weder gelöscht noch umbenannt werden.

Für das Attribut »FORMAT« gelten dieselben Regeln wie für schreibgeschützte Dateien. Zusätzlich gilt, daß Dateien, bei denen das Attribut »FORMAT« gesetzt ist, automatisch beim Aufruf des Kommandos FORMAT auf eine neue Diskette übertragen werden. Dieses Attribut sollte für die 4 ISIS-Dateien nie rückgesetzt werden (siehe Kapitel 3).

VALUE
Das niedrigstwertige Bit ($2^{Ø}$) des Parameters VALUE bestimmt den Zustand des ausgewählten Attributs. Wenn der Wert = Ø ist, wird das Attribut rückgesetzt, ist der Wert = 1, so wird das Attribut gesetzt.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf ATTRIB erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
1	Pufferbereich für den benötigten Puffer zu klein (nicht behebbarer Fehler)
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät im Dateinamen ist unzulässig oder dem System nicht bekannt
13	Die angegebene Datei ist im Diskettenetikettbereich nicht eingetragen
23	Für eine Diskettendatei ist kein Dateiname angegeben
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt (nicht behebbarer Fehler)
26	Unzulässiger Wert im Parameter SWID
28	Keine Erweiterung des Dateinamens vorhanden
3Ø	Floppy-Disk unklar (nicht behebbarer Fehler)

Beispiel

```
;
; ****************************************
;
; SYSTEMKONSTANTEN
         EXTRN   ISIS
ATTRIB   EQU     1Ø
;
; ATTRIB ROUTINE
;
         MVI     C,ATTRIB    ; CODE FUER ATTRIB LADEN
         LXI     D,ABLK      ; ADR. DER PARAMETERLISTE LADEN
         CALL    ISIS        ; SYSTEMAUFRUF ISIS
         LDA     ASTAT       ; STATUSBYTE AUF FEHLER PRUEFEN
         ORA     A
         JNZ     EXCEPT      ; FEHLERBEHANDLUNG
           .
           .
           .
           .
```

```
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF ATTRIB
;
ABLK:
        DW      AFILE       ; ADR., UNTER DER DER DATEINAME
                            ; STEHT
SWID:   DW      Ø           ; KENNUNG WELCHES ATTRIBUT
                            ; GEAENDERT WIRD
VALUE:  DW      1           ; WERT DES ATTRIBUTS
        DW      ASTAT       ; ADR. DER FEHLERNUMMER
;
ASTAT:  DS      2           ; FEHLERNUMMER (RUECKGEMELDET)
AFILE:  DB      'ISIS.CLI'  ; NAME DER DATEI
```

12.4. Zuweisen der Konsole und Ausgabe von Fehlermeldungen

Dem Anwender stehen Systemaufrufe zur Verfügung, um das System-Konsolgerät zu steuern. Diese Systemaufrufe erlauben es, das als Konsolgerät benutzte Gerät zu ändern, das augenblicklich als Konsolgerät benutzte Gerät zu ermitteln und eine Fehlermeldung auf dem Konsolgerät auszugeben.

12.4.1. CONSOL-Aufruf (Ändern des Konsolgerätes)

Allgemeines
Mit dem Systemaufruf CONSOL kann das Anwenderprogramm für die Ein- bzw. Ausgabe eine andere Konsole (:CI: und :CO:) zuweisen.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für CONSOL
= 8

Registerversorgung
Reg C = Code für CONSOL, also 8
Reg D/E = Adresse für Parameterliste

Aufbau der Parameterliste

CIPTR	Adresse einer ASCII-Zeichenfolge, die den Namen des Konsoleingabegerätes angibt
COPTR	Adresse einer ASCII-Zeichenfolge, die den Namen des Konsolausgabegerätes angibt
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird.

Beschreibung der Parameter

CIPTR
Der Parameter CIPTR enthält die Adresse einer ASCII-Zeichenfolge, die den Namen für das neue Konsoleingabegerät angibt. Die Zeichenfolge darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf das Konsoleingabegerät nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden.
Bevor die neue Datei für die Konsoleingabe eröffnet wird, wird die alte Datei geschlossen. Ist als neue Datei :CI: angegeben, hat der Systemaufruf CONSOL keinen Einfluß auf das bestehende Konsoleingabegerät. Kann die angegebene Datei für Eingabe nicht eröffnet werden, wird ein nicht behebbarer Fehler gemeldet.

COPTR
Der Parameter COPTR enthält die Adresse einer ASCII-Zeichenfolge, die den Namen für das neue Konsolausgabegerät angibt. Die Zeichenfolge ist in derselben Weise anzugeben, wie in CIPTR beschrieben.

Bevor die neue Datei für die Konsolausgabe eröffnet wird, wird die alte Datei geschlossen. Ist als neue Datei :CO: angegeben, hat der Systemaufruf CONSOL keinen Einfluß auf das bestehende Konsolausgabegerät. Kann die angegebene Datei für Ausgabe nicht eröffnet werden, wird ein nicht behebbarer Fehler gemeldet.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Der Systemaufruf CONSOL übergibt immer eine Ø, weil bei diesem Systemaufruf keine behebbaren Fehler auftreten können.

Fehlermöglichkeiten
Alle hier aufgeführten Fehler sind nicht behebbar.

Fehlernummer	Bedeutung
1	Pufferbereich für den benötigten Puffer zu klein
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät ist unzulässig oder dem System nicht bekannt
12	Versuch, eine schon eröffnete Datei zu eröffnen
13	Die angegebene Datei ist im Dateietikettbereich nicht eingetragen
14	Versuch, eine Datei mit dem gesetzten Attribut »Schreibschutz« umzuweisen
22	Zugriffsart für die angegebene Datei unzulässig
23	Für eine Diskettendatei ist kein Dateiname angegeben
24	Disketten-E/A-Fehler. Eine genaue Erläuterung ist in Kapitel 15 aufgeführt.
28	Keine Erweiterung des Dateinamens vorhanden
3Ø	Floppy-Disk unklar

Beispiel

```
;
; ***************************************
;
; SYSTEMKONSTANTEN
        EXTRN   ISIS
CONSOL  EQU     8
;
; CONSOL ROUTINE
;
        MVI     C,CONSOL    ; CODE FUER CONSOL LADEN
        LXI     D,KBLK      ; ADR. DER PARAMETERLISTE LADEN
        CALL    ISIS        ; SYSTEMAUFRUF ISIS
          .
          .
          .
          .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF CONSOL
;
KBLK:
        DW      INPUT       ; ADR., UNTER DER DAS KONSOLEIN-
                            ; GABEGERAET STEHT
        DW      OUTPUT      ; ADR., UNTER DER DAS KONSOLAUS-
                            ; GABEGERAET STEHT
        DW      KSTAT       ; ADR. DER FEHLERNUMMER
                            ;
KSTAT:  DS      2           ; FEHLERNUMMER
INPUT:  DB      ':TI:'      ; KONSOLEINGABEGERAET/-DATEI
                            ;
OUTPUT: DB      ':TO:'      ; KONSOLAUSGABEGERAET/-DATEI
```

12.4.2. WHOCON-Aufruf (Abfrage nach dem aktuellen Konsolgerät)

Allgemeines
Mit dem Systemaufruf WHOCON kann das Anwenderprogramm feststellen, welche Datei für das Konsoleingabe- bzw. Konsolausgabegerät zugewiesen ist.

Systemeinsprung in ISIS
CALL ISIS

Code für WHOCON
= 13

Registerversorgung
Reg C = Code für WHOCON, also 13
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

AFTN	Wert Ø für :CO: bzw. Wert 1 für :CI:
FILEPTR	Adresse eines Puffers für die Übergabe des Dateinamens
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird

Beschreibung der Parameter

AFTN
Der Parameter AFTN enthält die Tabellennummer Ø für :CO: bzw. 1 für :CI:. Es wird nur das niedrigstwertige Bit ($2^{Ø}$) ausgewertet.

FILEPTR
Der Parameter FILEPTR enthält die Adresse eines 15 Bytes langen Puffers im Anwenderprogramm, in dem der Name der zugewiesenen Datei für :CI: oder :CO: abgelegt wird. Der übergebene Name ist eine ASCII-Zeichenfolge, die mit einem Leerzeichen abgeschlossen ist.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Der Systemaufruf WHOCON erkennt keine Fehler, deshalb ist hier STATUS ein rein formaler Eintrag.

Beispiel

```
;
; ****************************************
;
; SYSTEMKONSTANTEN
          EXTRN   ISIS
WHOCON    EQU     13
;
; WHOCON ROUTINE
;
          MVI     C,WHOCON ; CODE FUER WHOCON LADEN
          LXI     D,HBLK   ; ADR. DER PARAMETERLISTE LADEN
          CALL    ISIS     ; SYSTEMAUFRUF ISIS
             .
             .
             .
             .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF WHOCON
;
HBLK:
IOD:      DS      2        ; TABELLENNUMMER Ø BZW 1
FILEPTR:  DW      CBUF     ; ADR. DES PUFFERS
          DW      CSTAT    ; ADR. DER FEHLERNUMMER
;
CBUF:     DS      15       ; PUFFER FUER KONSOLGERAET
                           ; (RUECKGELIEFERT)
CSTAT:    DS       2       ; FEHLERNUMMER
```

12.4.3. ERROR-Aufruf (Ausgabe einer Fehlermeldung über die Konsole)

Allgemeines
Mit dem Systemaufruf ERROR kann das Anwenderprogramm eine Fehlermeldung über die ursprünglich zugewiesene Konsole ausgeben.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für ERROR
= 12

Registerversorgung
Reg C = Code für ERROR, also 12
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

ERRNUM	Fehlernummer in den niederwertigen 8 Bits
STATUS	Adresse einer Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird.

Beschreibung der Parameter
ERRNUM
Der Parameter ERRNUM enthält die auszugebende Fehlernummer. Es dürfen nur die Nummern 1Ø1 bis einschließlich 199 verwendet werden, da die anderen Nummern für Systemprogramme reserviert sind.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Der Systemaufruf ERROR erkennt keine Fehler, deshalb ist hier STATUS ein rein formaler Eintrag.

Format der Fehlermeldung

ERROR nnn, USER PC mmmm

wobei nnn die im Systemaufruf ERROR angegebene Fehlernummer und mmmm die Rücksprungadresse in das rufende Programm ist.

Beispiel

```
;
; ************************************
;
; SYSTEMKONSTANTEN
        ISIS    EXTRN
ERROR   EQU     12
;
; ERROR ROUTINE
;
        MVI     C,ERROR     ; CODE FUER ERROR LADEN
        LXI     D,XBLK      ; ADR. DER PARAMETERLISTE LADEN
        CALL    ISIS        ; SYSTEMAUFRUF ISIS
            .
            .
            .
            .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF ERROR
;
XBLK:
ERRNUM    DS    2           ; AUSZUGEBENDE FEHLERNUMMER
          DW    XSTAT       ; ADR. DER FEHLERNUMMER
;
XSTAT:    DS    2           ; FEHLERNUMMER (RUECKGEMELDET)
```

12.5. Programmausführung

Dem Anwender stehen zwei Systemaufrufe zur Verfügung, die Ablaufsteuerung aus seinem Programm einem anderen Programm (LOAD) oder an ISIS II (EXIT) zu übergeben. Der Systemaufruf LOAD kann verwendet werden, ein anderes Programm zu laden und die Steuerung diesem Programm oder dem Monitor zu übergeben. Es besteht auch die Möglichkeit, die Steuerung dem aufrufenden Programm zurückzugeben. Der Systemaufruf EXIT wird verwendet, um die Verarbeitung zu beenden und zu ISIS II zurückzukehren.

12.5.1. LOAD-Aufruf (Laden einer Datei mit ablauffähigem Code)

Allgemeines
Mit dem Systemaufruf LOAD kann ein Anwenderprogramm eine Datei mit ablauffähigem Code laden. Nachdem die Datei geladen ist, kann die Steuerung dem geladenen Programm,

dem rufenden Programm oder dem SME-Monitor übergeben werden, abhängig vom Wert des Parameters RETSW.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für LOAD
= 6

Registerversorgung
Reg C = Code für LOAD, also 6
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

FILEPTR	Adresse einer ASCII-Zeichenfolge, die den Namen der zu ladenden Datei angibt.
BIAS	Basisadresse (modulo 64 K), die zu der Ladeadresse des Anwenderprogramms addiert wird.
RETSW	Wert, der anzeigt, wohin die Steuerung nach dem Laden des Programms übergeben werden soll
ENTRY	Adresse einer Speicherstelle für die Rückmeldung der Einsprungadresse des geladenen Programms
STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird.

Beschreibung der Parameter

FILEPTR
Der Parameter FILEPTR enthält die Adresse einer ASCII-Zeichenfolge, die den Namen der Datei angibt, die geladen werden soll. Der Inhalt der Datei muß ein ablauffähiges Programm im absoluten ISIS-II-Format sein. Die Zeichenfolge darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf der Dateiname nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden.

BIAS
Der Parameter BIAS enthält eine Basisadresse, die zu der Ladeadresse des zu ladenden Programms addiert wird. Das Programm wird dann an die Speicheradresse geladen, die sich durch diese Addition ergibt. Das heißt nicht, daß das Programm im Speicher verschiebbar ist; es ist nicht unter der errechneten Adresse ablauffähig, sondern nur unter der Adresse, wohin es mit dem LOCATOR gelegt wurde. Deshalb ist bei den meisten Anwendungen der Parameter BIAS = Ø.

RETSW
Der Wert des Parameters RETSW bestimmt, wem nach dem Laden des Programms die Steuerung übertragen werden soll.

RETSW = Ø, nach dem Laden wird die Steuerung dem rufenden Programm zurückgegeben. Der Testschalter wird nicht verändert.

RETSW = 1, nach dem Laden wird die Steuerung dem geladenen Programm übertragen und dieses an seinem Einsprungpunkt gestartet. Der Testschalter wird auf Ø gesetzt. Falls das Programm kein Hauptprogramm ist, ist der Einsprung = Ø, was die Übergabe der Steuerung an den Monitor bewirkt.

RETSW = 2, nach dem Laden wird die Steuerung dem SME-Monitor übertragen. Der Testschalter wird auf 1 gesetzt. Mit dem Monitorkommando G kann dann das geladene Programm gestartet werden.

Hinweis!
Wenn die Steuerung an den Monitor übergeben wird (RETSW = 2), stellt dieser die CPU-Register und den Systemzustand sicher, indem er sie in den Stack bringt, der zu dem Programm gehört, das den LOAD-Systemaufruf gegeben hat. Falls das neu geladene Programm den vorher durch den Stack belegten Bereich überlagert, wird das Programm überschrieben, wenn der Monitor Daten in den Stack bringt.

Deshalb sollte der Stack in einem Speicherbereich untergebracht werden, der es unmöglich macht, daß das neue Programm, wenn der Monitor den Stack benutzt, überschrieben wird.

ENTRY
Der Parameter ENTRY enthält die Adresse einer Speicherstelle für die Rückmeldung der Einsprungadresse des geladenen Programms, wenn RETSW = Ø ist. Die Einsprungadresse wird vom geladenen Programm übernommen. Falls das Programm kein Hauptprogramm ist, wird die Adresse Ø zurückgemeldet.

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Die Fehler, die vom Systemaufruf LOAD erkannt werden, sind anschließend aufgeführt.

Fehlermöglichkeiten

Fehlernummer	Bedeutung
1	Pufferbereich für den benötigten Puffer zu klein (nicht behebbarer Fehler)
3	Versuch, mehr als 6 Dateien gleichzeitig zu bearbeiten
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät im Dateinamen ist unzulässig oder dem System nicht bekannt
12	Versuch, eine schon eröffnete Datei zu laden
13	Die angegebene Datei ist im Diskettenetikettbereich nicht vorhanden
15	Versuch, eine Datei in den ISIS-residenten Bereich zu laden (nicht behebbarer Fehler)
16	Summenfehler beim Laden des ablauffähigen Programms (nicht behebbarer Fehler)
23	Für eine Diskettendatei ist kein Dateiname angegeben

24	Disketten-E/A-Fehler. Eine genaue Erläuterung wird in Kapitel 15 gegeben (nicht behebbarer Fehler)
28	Keine Erweiterung des Dateinamens vorhanden
30	Floppy-Disk unklar (nicht behebbarer Fehler)
34	Der Wert des Parameters RETSW ist unzulässig

Beispiel

```
;
; ****************************************
;
; SYSTEMKONSTANTEN
        EXTRN   ISIS
LOAD    EQU     6
;
; LOAD ROUTINE
;
        MVI     C,LOAD  ; CODE FUER LOAD LADEN
        LXI     D,LBLK  ; ADR. DER PARAMETERLISTE LADEN
        CALL    ISIS    ; SYSTEMAUFRUF ISIS
        LDA     LSTAT   ; STATUSBYTE AUF FEHLER PRUEFEN
        ORA     A
        JNZ     EXCEPT  ; FEHLERBEHANDLUNG
          .
          .
          .
          .
;
; PARAMETERLISTE FUER DEN SYSTEMAUFRUF LOAD
;
LBLK:
        DW      LFILE       ; ADR., UNTER DER DER DATEINAME
                            ; STEHT
BIAS:   DW      Ø           ; BIASWERT
RETSW:  DW      Ø           ; SCHALTER FUER DIE UEBERGABE
                            ; DER STEUERUNG
```

```
        DW      ENTRY       ; ADR., UNTER DER DIE EINSPRUNG-
                            ; ADRESSE DES GERUFENEN PROGRAMMS
                            ; STEHT
        DW      LSTAT       ; ADR. DER FEHLERNUMMER
;
ENTRY:  DS      2           ; EINSPRUNGADRESSE (RUECKGEMELDET)
LSTAT:  DS      2           ; FEHLERNUMMER (RUECKGEMELDET)
LFILE:  DB      'FILE.EXT'; DATEINAME
```

12.5.2. EXIT-Aufruf (Beenden des Programms und Rückgabe der Steuerung an ISIS II)

Allgemeines
Mit dem Systemaufruf EXIT wird ein Anwenderprogramm beendet und die Steuerung ISIS übergeben. Dabei werden, bis auf die Dateien :CO: und :CI: alle Dateien geschlossen. Die augenblickliche Gerätezuordnung für die Konsole bleibt bestehen.

Systemeinsprungstelle in ISIS
CALL ISIS

Code für EXIT
= 9

Registerversorgung
Reg C = Code für EXIT, also 9
Reg D/E = Adresse der Parameterliste

Aufbau der Parameterliste

STATUS	Adresse der Speicherstelle, wohin die Nummer eines behebbaren Fehlers rückgemeldet wird.

Beschreibung des Parameters

STATUS
Der Parameter STATUS enthält die Adresse einer Speicherstelle, unter der die Nummer eines behebbaren Fehlers rückgemeldet wird. Der Parameter STATUS ist hier ein rein formaler Eintrag.

Beispiel

```
;
; ****************************************
; SYSTEMKONSTANTEN
        EXTRN   ISIS
EXIT    EQU     9
;
; EXIT ROUTINE
;
        MVI     C,EXIT          ; CODE FUER EXIT LADEN
        LXI     D,EBLK          ; ADR.DER PARAMETERLISTE LADEN
        CALL    ISIS            ; SYSTEMAUFRUF ISIS
            .
            .
            .
            .
            ;
            ; PARAMETERLISTE FUER DEN SYSTEMAUFRUF EXIT
                                ;
            EBLK:  DW ESTAT     ; ADR. DER FEHLERNUMMER
            ;
            ESTAT: DS 2         ; FEHLERNUMMER (RUECKGEMELDET)
```

13. PL/M-Schnittstelle

13.1. PL/M-Schnittstelle

Ein PL/M-Programm kann mit ISIS II verknüpft werden, indem es Prozeduren aus der Systembibliotheksdatei SYSTEM.LIB aufruft. Diese schaffen die Verbindung zu den Systemroutinen von ISIS II. Die Systemaufrufe sind in Kapitel 12 beschrieben. Das Programm muß die Deklarationen für die externen ISIS-II-Prozeduren enthalten. Die folgenden Vereinbarungen gelten für die Variablen, die bei den ISIS-II-Systemaufrufen zu übergeben sind:

1. Jede Variable eines Systemaufrufs ist vom Typ ADDRESS.
2. Die meisten Systemprogramme liefern ein Kennzeichen, die Fehlernummer, an das aufrufende Programm zurück, wenn während der Abarbeitung des Systemprogrammes ein Fehler auftritt. Im folgenden wird diese Variable mit STATUS bezeichnet. Ist der Inhalt von STATUS nach der Übergabe der Steuerung an das Anwenderprogramm = Ø, ist bei der Abarbeitung des Systemaufrufs kein Fehler erkannt worden. Die Behandlung behebbarer und nichtbehebbarer Fehler ist in Kapitel 2 beschrieben.
3. Die Variable FILEPTR bezeichnet die Anfangsadresse eines Speicherbereiches, in dem der Dateiname steht. Dieser Name muß dem Aufbau von ISIS-Dateinamen entsprechen. Der Dateiname darf führende Leerzeichen, aber keine eingeschlossenen enthalten. Außerdem darf der Dateiname nicht mit einem Buchstaben, einer Ziffer, einem Doppelpunkt (:) oder einem Punkt (.) abgeschlossen werden.

13.2. Externe PL/M-Prozedurdeklarationen für ISIS-II-Systemaufrufe

Im folgenden Abschnitt werden die externen Prozedurdeklarationen für alle Systemaufrufe aufgeführt. Es sind nur die Deklarationen für solche Systemaufrufe anzugeben, die vom Programm tatsächlich verwendet werden. Dazu wird auf das Programmbeispiel am Schluß dieses Kapitels verwiesen.

```
OPEN:
  PROCEDURE (AFTPTR,FILE,ACCESS,MODE,STATUS) EXTERNAL;
    DECLARE (AFTPTR,FILE,ACCESS,MODE,STATUS) ADDRESS;
  END OPEN;

CLOSE:
  PROCEDURE (AFT,STATUS) EXTERNAL;
    DECLARE (AFT,STATUS) ADDRESS;
  END CLOSE;

DELETE:
  PROCEDURE (FILE,STATUS) EXTERNAL;
    DECLARE (FILE,STATUS) ADDRESS;
  END DELETE;
```

```
READ:
  PROCEDURE (AFT,BUFFER,COUNT,ACTUAL,STATUS) EXTERNAL;
    DECLARE (AFT,BUFFER,COUNT,ACTUAL,STATUS) ADDRESS;
  END READ;

WRITE:
  PROCEDURE (AFT,BUFFER,COUNT,STATUS) EXTERNAL;
    DECLARE (AFT,BUFFER,COUNT,STATUS) ADDRESS;
  END WRITE;

SEEK:
  PROCEDURE (AFT,BASE,BLOCKNUM,BYTENUM,STATUS) EXTERNAL;
    DECLARE (AFT,BASE,BLOCKNUM,BYTENUM,STATUS) ADDRESS;
  END SEEK;

LOAD:
  PROCEDURE (FILE,BIAS,RETSW,ENTRY,STATUS) EXTERNAL;
    DECLARE (FILE,BIAS,RETSW,ENTRY,STATUS) ADDRESS:
  END LOAD;

RENAME:
  PROCEDURE (OLDFILE,NEWFILE,STATUS) EXTERNAL;
    DECLARE (OLDFILE,NEWFILE,STATUS) ADDRESS;
  END RENAME;

CONSOL:
  PROCEDURE (INFILE,OUTFILE,STATUS) EXTERNAL;
    DECLARE (INFILE,OUTFILE,STATUS) ADDRESS;
  END CONSOL;

EXIT:
  PROCEDURE EXTERNAL;
  END EXIT;

ATTRIB:
  PROCEDURE (FILE,SWID,VALUE,STATUS) EXTERNAL;
    DECLARE (FILE,SWID,VALUE,STATUS) ADDRESS;
  END ATTRIB;
```

```
RESCAN:
  PROCEDURE (AFT,STATUS) EXTERNAL;
    DECLARE (AFT,STATUS) ADDRESS;
  END RESCAN;

ERROR:
  PROCEDURE (ERRNUM) EXTERNAL;
    DECLARE (ERRNUM) ADDRESS;
  END ERROR;

WHOCON:
  PROCEDURE (AFT,BUFFER) EXTERNAL;
    DECLARE (AFT,BUFFER) ADDRESS;
  END WHOCON;
```

13.3. PL/M-Programmbeispiel mit ISIS-II-Systemaufrufen

In dem Beispiel wird eine Datei über die Konsole ausgegeben. Das Programm wird aufgerufen mit:

TYPE dateiname

Zu beachten ist, daß nur die benötigten Systemaufrufe deklariert zu werden brauchen. Nach dem Übersetzen des Programms muß es mit dem Systemprogramm LINK und der SYSTEM.LIB gebunden und mit dem Systemprogramm LOCATE entrelativiert werden, bevor es ausgeführt werden kann.

```
TYPE:
  DO;
    DECLARE BUFFER (128) BYTE;
    DECLARE ACTUAL$COUNT ADDRESS;
    DECLARE STATUS ADDRESS;
    DECLARE AFT$IN ADDRESS;
    DECLARE READ$ACCESS LITERALLY '1';

OPEN:
  PROCEDURE (AFT,FILE,ACCESS,MODE,STATUS) EXTERNAL;
    DECLARE (AFT,FILE,ACCESS,MODE,STATUS) ADDRESS;
  END OPEN;
```

```
CLOSE:
  PROCEDURE (AFT,STATUS) EXTERNAL;
    DECLARE (AFT,STATUS) ADDRESS;
  END CLOSE;

READ:
  PROCEDURE (AFT,BUFFER,COUNT,ACTUAL,STATUS) EXTERNAL;
    DECLARE (AFT,BUFFER,COUNT,ACTUAL,STATUS) ADDRESS;
  END READ;

WRITE:
  PROCEDURE (AFT,BUFFER,COUNT,STATUS) EXTERNAL;
    DECLARE (AFT,BUFFER,COUNT,STATUS) ADDRESS;
  END WRITE;

EXIT:
  PROCEDURE EXTERNAL;
    DECLARE STATUS ADDRESS;
  END EXIT;

ERROR:
  PROCEDURE (ERRNUM) EXTERNAL;
    DECLARE (ERRNUM,STATUS) ADDRESS;
  END ERROR;



/* DIE ZU EROEFFNENDE DATEI VON :CI: LESEN */

   CALL READ (1,.BUFFER,128,.ACTUAL$COUNT,.STATUS);

/* DIE AKTUELLE DATEI EROEFFNEN */

   CALL OPEN (.AFT$IN,.BUFFER,READ$ACCESS,Ø,.STATUS);

/* SOLANGE BYTES VON DER DISKETTE LESEN UND UEBER DIE
   KONSOLE AUSGEBEN, BIS EOF ERREICHT IST. */
```

```
   ACTUAL$COUNT = 1;
   DO WHILE ACTUAL$COUNT  < >Ø
      CALL READ (AFT$IN,.BUFFER,128,.ACTUAL$COUNT,.STATUS);
      CALL WRITE (Ø,.BUFFER,ACTUAL$COUNT,.STATUS);
   END;

/* DATEI SCHLIESSEN UND UEBERGABE DER STEUERUNG AN ISIS */

   CALL CLOSE (AFT$IN,.STATUS);
   CALL EXIT;

END;
```

14. Assembler-Schnittstelle

14.1. Assembler-Schnittstelle

Die Verknüpfung von einem in der 8080-Assemblersprache geschriebenen Programm zu ISIS II wird dadurch hergestellt, daß ein einzelner Einsprungpunkt in ISIS II aufgerufen wird (genannt ISIS) und zwei Parameter übergeben werden. Der erste Parameter ist eine Zahl, die den Systemaufruf (den Code) kennzeichnet; der zweite ist die Adresse einer Parameterliste, welche die zusätzlich vom Systemaufruf benötigten Parameter enthält (wie in Kapitel 12 beschrieben). Der erste Parameter wird im Register C übergeben, die Adresse der Parameterliste im Registerpaar D und E. Das allgemeine Format der Schnittstelle lautet:

```
MVI  C, Kennziffer des Systemaufrufs
LXI  D, Adresse der Parameterliste
CALL ISIS
```

Der ISIS-Einsprungpunkt ist in einer Routine in der SYSTEM.LIB definiert. Diese Routine muß mit dem LINK-Kommando eingebunden werden. Dazu muß im LINK-Kommando die SYSTEM.LIB angegeben werden. Für weitere Informationen zum LINK-Kommando wird auf die Kapitel über das Relativieren und Binden verwiesen.

Die Kennziffern der Systemaufrufe können durch EQU-Anweisungen definiert werden. Es sind nur die Systemaufrufe zu definieren, die speziell vom vorliegenden Programm benötigt werden.

Die Einzelheiten jedes Systemaufrufs und seiner zugehörigen Parameter wurden bereits bei der Beschreibung der Systemaufrufe dargestellt.

14.2. Assembler-Programmbeispiel mit ISIS-II-Systemaufrufen

In dem Beispiel wird eine Datei über die Konsole ausgegeben. Das Programm wird aufgerufen mit:

```
TYPE dateiname
```

Nach dem Übersetzen des Programms muß es mit dem Systemprogramm LINK und der SYSTEM.LIB gebunden und mit dem Systemprogramm LOCATE entrelativiert werden, bevor es ausgeführt werden kann.

```
      NAME  TYPE
;     PROGRAMMBEISPIEL
;
;
OPEN  EQU   Ø          ;KENNZIFFERN DER BENOETIGTEN
CLOSE EQU   1          ;SYSTEMAUFRUFE
READ  EQU   3
WRITE EQU   4
EXIT  EQU   9
ERROR EQU   12
;
      EXTRN ISIS       ;EXTERNAUFRUF DES ISIS EINSPRUNGPUNKTES
;
      CSEG             ;ANFANG DES CODESEGMENTS
BEGIN:
      LXI   SP,STACK+4 ;STACKPOINTER LADEN
      MVI   C,READ
      LXI   D,RBLK     ;DATEINAMEN VON DER KONSOLE
      CALL  ISIS       ;LESEN
      LDA   STATUS
      ORA   A
      JNZ   ERR
;
      MVI   C,OPEN     ;DIEJENIGE DATEI EROEFFNEN; DEREN NAME
      LXI   D,OBLK     ;DURCH DIE EINGABE VON :CI: IM PUFFER
      CALL  ISIS       ;STEHT
      LDA   STATUS
      ORA   A
      JNZ   ERR
      LHLD  AFT        ;AFTNR, DIE BEIM OPEN ZUGEWIESEN WURDE,
      SHLD  CAFT       ;IN PARAMETERLISTE FUER CLOSE
;
;
```

```
LOOP: MVI   C,READ      ;EINGABEDATEI LESEN (128 BYTES)
      LXI   D,RBLK
      CALL  ISIS
      LDA   STATUS
      ORA   A
      JNZ   ERR
;
      LHLD  ACTUAL      ;ABFRAGE OB LETZTER BLOCK SCHON
      MOV   A,H         ;VERARBEITET, D.H. EOF ERREICHT
      ORA   L
      JZ    DONE
;
      MVI   C,WRITE     ;BLOCK AUF DIE KONSOLE AUSGEBEN
      LXI   D,WBLK      ;(< = 128 BYTES)
      CALL  ISIS
      LDA   STATUS
      ORA   A
      JNZ   ERR
      JMP   LOOP        ;SPRUNG ZUM ERNEUTEN LESEN
;
DONE: MVI   C,CLOSE     ;IST DIE GESAMTE DATEI AUSGEGEBEN
      LXI   D,CBLK      ;MUSS DIE AKT. DATEI GESCHLOSSEN
      CALL  ISIS        ;WERDEN
;
      MVI   C,EXIT      ;PROGRAMM BEENDEN UND STEUERUNG
      LXI   D,XBLK      ;AN ISIS ZURUECKGEBEN
      CALL  ISIS
;
ERR:    MVI  C,ERROR    ;WIRD WAEHREND EINER E/A-OPERATION
        LXI  D,EBLK     ;EIN FEHLER ERKANNT, DIESEN AN-
        CALL ISIS       ;ZEIGEN UND PROGRAMM BEENDEN
        MVI  C,EXIT
        LXI  D,XBLK
        CALL ISIS
;
        DSEG            ;ANFANG DES DATENSEGMENTS
```

```
;
;PARAMETERLISTE ZUM LESEN VON :CI: UND DER AKT. DATEI
;
RBLK:   DW   1        ;1 = > :CI:
        DW   PUFF     ;PUFFER FUER GELESENE DATEN
        DW   128      ;ANZ DER ZU LES. BYTES
        DW   ACTUAL   ;ANZ DER GELES. BYTES
        DW   STATUS   ;STATUS NACH DEM LESEN
;
;PARAMETERLISTE FUER DAS SCHREIBEN DER AKT. DATEI
;
WBLK:   DW   Ø        ;Ø =>:CO:
        DW   PUFF     ;PUFFER FUER ZU SCHREIBENDE DATEN
ACTUAL: DS   2        ;ANZ. DER ZU SCHREIBENDEN BYTES
        DW   STATUS   ;STATUS NACH DEM SCHREIBEN
;
;PARAMETERLISTE FUER DAS EROEFFNEN DER AKT. DATEI
;
OBLK:   DW   AFT      ;NR. DER ZU LESENDEN DATEI
        DW   PUFF     ;NAME DER AKT. DATEI
        DW   1        ;ZUGRIFFSART: LESEN
        DW   Ø        ;KEINE KONTROLLDATEI
        DW   STATUS   ;STATUS NACH DEM OPEN
;PARAMETERLISTE FUER DAS SCHLIESSEN DER AKT. DATEI
;
CBLK:
CAFT:   DS   2        ;NR. DER ZU SCHLIESSENDEN DATEI
        DW   STATUS   ;STATUS NACH DEM CLOSE
;
;PARAMETERLISTE FUER DIE NORMALE BEENDIGUNG DES PROGRAMMS
;BZW. FUER DIE FEHLERBEHANDLUNG UND RUECKSPRUNG ZU ISIS
;
XBLK:
EBLK:     DW   STATUS
;
;
```

```
;SPEICHERDEFINITIONEN
;
PUFF:     DS   128
STATUS:   DS   2
          END  BEGIN
```

14.3. Verwendung von Makros für die Schnittstelle zu ISIS II

Um das Arbeiten mit den ISIS-II-Systemaufrufen zu erleichtern, können Makros verwendet werden.

Der folgende Makro lädt das Register C mit dem Code des Systemaufrufs, das Registerpaar D/E mit der Adresse der Parameterliste ruft ISIS auf und springt zu einer Fehlerroutine, wenn ein Fehler bei der Bearbeitung des Systemaufrufes erkannt worden ist.

```
DOISI  MACRO  TYPE,BLOCK,ERR
       MVI    C,TYPE
       LXI    D,BLOCK
       CALL   ISIS
       LDA    STATUS
       ORA    A
       JNZ    ERR
       ENDM
```

Mit diesem Makro kann das Schreiben der Assemblerbefehle für jeden Systemaufruf auf ein Minimum beschränkt werden. So kann z. B. der Makroaufruf für READ folgendermaßen aussehen:

```
DOISI  READ,RBLK,ERR
```

14.4. Makrodefinitionen im Assemblerprogramm

Der nachfolgend aufgeführte Satz von Makros erlaubt ISIS-Systemaufrufe in Assemblerprogrammen, wobei der Aufbau der Makros wie bei PL/M-Aufrufen strukturiert ist, mit dem Unterschied, daß die Parameter den Makros als Adressen übergeben werden müssen und nicht als Literale. Der vom ISIS-Systemaufruf benötigte Parameterblock wird dynamisch im Stack verwaltet. Somit ist es möglich, die Makros in einen Festwertspeicher (ROM) zu legen. Es werden maximal 12 Bytes Stack benötigt.

14.5. TYPE-Programm unter Verwendung von Makros für die ISIS-II-Aufrufe

```
;
; ********************************************************
;
; OPEN
;
OPEN  MACRO  AFT,FILE,ACESS,ECHO,STATUS
      LXI    D,STATUS      ;ADRESSE VON STATUS
      PUSH   D             ;IN DEN STACK
      LHLD   ECHO          ;WERT DES PARAMETERS ECHO
      PUSH   H             ;IN DEN STACK
      LHLD   ACCESS        ;WERT DES PARAMETERS ACCESS
      PUSH   H             ;IN DEN STACK
      LXI    D,FILE        ;ADRESSE DES DATEINAMENS
      PUSH   D             ;IN DEN STACK
      LXI    D,AFT         ;ADRESSE UNTER DER DIE TAB.NUMMER
      PUSH   D             ;GESPEICHERT WIRD IN DEN STACK
      LXI    H,Ø
      DAD    SP            ;STACKPOINTER NACH
      XCHG                 ;REG D/E
      MVI    C,Ø           ;CODE FUER OPEN LADEN
      CALL   ISIS          ;SYSTEMAUFRUF ISIS
      POP    D
      POP    D             ;STACK WIEDER FREIGEBEN
      POP    D
      POP    D
      POP    D
      ENDM
```

```
;
; ******************************************************
;
; CLOSE
;
CLOSE    MACRO  AFT,STATUS
         LXI    D,STATUS     ;ADRESSE VON STATUS
         PUSH   D            ;IN DEN STACK
         LHLD   AFT          ;WERT DER TABELLENNUMMER
         PUSH   H            ;IN DEN STACK
         LXI    H,Ø
         DAD    SP           ;STACKPOINTER NACH REG D/E
         XCHG
         MVI    C,1          ;CODE FUER CLOSE LADEN
         CALL   ISIS         ;SYSTEMAUFRUF ISIS
         POP    D
         POP    D            ;STACK WIEDER FREIGEBEN
         ENDM
;
; ******************************************************
;
; DELETE
;
DELETE   MACRO  FILE,STATUS
         LXI    D,STATUS     ;ADRESSE VON STATUS
         PUSH   D            ;IN DEN STACK
         LXI    D,FILE       ;ADRESSE DES DATEINAMENS
         PUSH   D            ;IN DEN STACK
         LXI    H,Ø
         DAD    SP           ;STACKPOINTER NACH REG D/E
         XCHG
         MVI    C,2          ;CODE FUER DELETE LADEN
         CALL   ISIS         ;SYSTEMAUFRUF ISIS
         POP    D
         POP    D            ;STACK WIEDER FREIGEBEN
         ENDM
```

```
;
; ********************************************************
;
; READ
;
READ    MACRO  AFT,BUFFER,COUNT,ACTUAL,STATUS
        LXI    D,STATUS     ;ADRESSE VON STATUS
        PUSH   D            ;IN DEN STACK
        LXI    D,ACTUAL     ;ADRESSE UNTER DER DIE ANZAHL
                            ;DER GELESENEN BYTES STEHT
        PUSH   D            ;IN DEN STACK
        LHLD   COUNT        ;ANZAHL DER ZU LESENDEN BYTES
        PUSH   H            ;IN DEN STACK
        LXI    D,BUFFER     ;ADRESSE DES EINGABEPUFFERS
        PUSH   D            ;IN DEN STACK
        LHLD   AFT          ;WERT DER TABELLENNUMMER
        PUSH   H            ;IN DEN STACK
        LXI    H,Ø
        DAD    SP           ;STACKPOINTER NACH REG D/E
        XCHG
        MVI    C,3          ;CODE FUER READ LADEN
        CALL   ISIS         ;SYSTEMAUFRUF ISIS
        POP    D
        POP    D
        POP    D            ;STACK WIEDER FREIGEBEN
        POP    D
        POP    D
        ENDM
```

```
;
; ***********************************************
;
; WRITE
;
WRITE  MACRO  AFT,BUFFER,COUNT,STATUS
       LXI    D,STATUS     ; ADRESSE VON STATUS
       PUSH   D            ; IN DEN STACK
       LHLD   COUNT        ; ADRESSE IN DER DIE ANZAHL
       PUSH   H            ; DER ZU SCHREIBENDEN BYTES
                           ; STEHT IN DEN STACK
       LXI    D,BUFFER     ; ADRESSE DES AUSGABEPUFFERS
       PUSH   D            ; IN DEN STACK
       LHLD   AFT          ; WERT DER TABELLENNUMMER
       PUSH   H            ; IN DEN STACK
       LXI    H,Ø
       DAD    SP           ; STACKPOINTER NACH REG D/E
       XCHG
       MVI    C,4          ; CODE FUER WRITE LADEN
       CALL   ISIS         ; SYSTEMAUFRUF ISIS
       POP    D
       POP    D
       POP    D            ; STACK WIEDER FREIGEBEN
       POP    D
       ENDM
```

```
;
; ***********************************************
;
; SEEK
;
SEEK    MACRO  AFT,BASE,BLKNO,BYTENO,STATUS
        LXI    D,STATUS     ; ADRESSE VON STATUS
        PUSH   D            ; IN DEN STACK
        LXI    D,BYTENO     ; ADRESSE, IN DER DIE ANZAHL
        PUSH   D            ; DER ZU UEBERSPRINGENDEN
                            ; BYTES STEHT, IN DEN STACK
        LXI  D,BLKNO        ; ADRESSE IN DER DIE ANZAHL DER ZU
        PUSH D              ; UEBERSPRINGENDEN BLOECKE STEHT
                            ; IN DEN STACK
        LHLD BASE           ; WERT DER MODIFIKATIONSART
        PUSH H              ; IN DEN STACK
        LHLD AFT            ; WERT DER TABELLENNUMMER
        PUSH H              ; IN DEN STACK
        LXI  H,Ø
        DAD  SP             ; STACKPOINTER NACH REG D/E
        XCHG
        MVI  C,5            ; CODE FUER SEEK LADEN
        CALL ISIS           ; SYSTEMAUFRUF ISIS
        POP  D
        POP  D
        POP  D              ; STACK WIEDER FREIGEBEN
        POP  D
        POP  D
        ENDM
```

```
;
; ***************************************************
;
; LOAD
;
LOAD  MACRO  FILE,BIAS,RETSW,ENTRY,STATUS
      LXI    D,STATUS  ; ADRESSE VON STATUS
      PUSH   D         ; IN DEN STACK
      LXI    D,ENTRY   ; ADRESSE IN DIE DIE EINSPRUNG-
      PUSH   D         ; ADRESSE EINGETRAGEN WIRD, IN DEN STACK
      LHLD   RETSW     ; WERT, WEM DIE STEUERUNG AN-
      PUSH   H         ; SCHLIESSEND UEBERGEBEN WIRD,
                       ; IN DEN STACK
      LHLD   BIAS      ; WERT DER BASISADRESSE BEIM LADEN
      PUSH   H         ; IN DEN STACK
      LXI    D,FILE    ; ADRESSE DES DATEINAMENS
      PUSH   D         ; IN DEN STACK
      LXI    H,Ø
      DAD    SP        ; STACKPOINTER NACH REG D/E
      XCHG
      MVI    C,6       ; CODE FUER LOAD LADEN
      CALL   ISIS      ; SYSTEMAUFRUF ISIS
      POP    D
      POP    D
      POP    D         ; STACK WIEDER FREIGEBEN
      POP    D
      POP    D
      ENDM
```

```
;
; ******************************************************
;
; RENAME
;
RENAME  MACRO   OLD,NEW,STATUS
        LXI     D,STATUS        ; ADRESSE VON STATUS
        PUSH    D               ; IN DEN STACK
        LXI     D,NEW           ; ADRESSE DES NEUEN DATEINAMENS
        PUSH    D               ; IN DEN STACK
        LXI     D,OLD           ; ADRESSE DES ALTEN DATEINAMENS
        PUSH    D               ; IN DEN STACK
        LXI     H,Ø
        DAD     SP              ; STACKPOINTER NACH REG D/E
        XCHG
        MVI     C,7             ; CODE FUER RENAME LADEN
        CALL    ISIS            ; SYSTEMAUFRUF ISIS
        POP     D
        POP     D               ; STACK WIEDER FREIGEBEN
        POP     D
        ENDM
```

```
;
; ************************************************
;
; CONSOL
;
CONSOL  MACRO   INF,OUTF,STATUS
        LXI     D,STATUS        ; ADRESSE VON STATUS
        PUSH    D               ; IN DEN STACK
        LXI     D,OUTF          ; ADRESSE DER AUSGABEKONSOLE
        PUSH    D               ; IN DEN STACK
        LXI     D,INF           ; ADRESSE DER EINGABEKONSOLE
        PUSH    D               ; IN DEN STACK
        LXI     H,Ø
        DAD     SP              ; STACKPOINTER NACH REG D/E
        XCHG
        MVI     C,8             ; CODE FUER CONSOLE LADEN
        CALL    ISIS            ; SYSTEMAUFRUF ISIS
        POP     D
        POP     D               ; STACK WIEDER FREIGEBEN
        POP     D
        ENDM
;
; ********************************************************
;
; EXIT
;
EXIT    MACRO
        LXI     H,-2            ; STACKPOINTER -2 SETZEN
        DAD     SP
        PUSH    H               ; STACKPOINTER IN DEN STACK
        XCHG                    ; STACKPOINTER NACH REG D/E
        MVI     C,9             ; CODE FUER EXIT LADEN
        CALL    ISIS            ; SYSTEMAUFRUF ISIS
        EI                      ; RUECKSPRUNG NUR,WENN
        HLT                     ; ISIS II CLI ZERSTOERT IST
        ENDM
```

```
;
; ******************************************************
;
; ATTRIB
;
ATTRIB  MACRO   FILE,SWID,VALUE,STATUS
        LXI     D,STATUS    ; ADRESSE VON STATUS
        PUSH    D           ; IN DEN STACK
        LHLD    VALUE       ; WERT DES ATTRIBUTES
        PUSH    H           ; IN DEN STACK
        LHLD    SWID        ; WERT WELCHES ATTRIBUT ANGE-
                            ; SPROCHEN WIRD
        PUSH    H           ; IN DEN STACK
        LXI     D,FILE      ; ADRESSE DES DATEINAMENS
        PUSH    D           ; IN DEN STACK
        LXI     H,Ø
        DAD     SP          ; STACKPOINTER NACH REG D/E
        XCHG
        MVI     C,1Ø        ; CODE FUER ATTRIB LADEN
        CALL    ISIS        ; SYSTEMAUFRUF ISIS
        POP     D
        POP     D
        POP     D           ; STACK WIEDER FREIGEBEN
        POP     D
        ENDM
```

```
;
; ***************************************************
;
; RESCAN
;
RESCAN  MACRO   AFT,STATUS
        LXI     D,STATUS    ; ADRESSE VON STATUS
        PUSH    D           ; IN DEN STACK
        LHLD    AFT         ; WERT DER TABELLENNUMMER
        PUSH    H           ; IN DEN STACK
        LXI     H,Ø
        DAD     SP          ; STACKPOINTER NACH REG D/E
        XCHG
        MVI     C,11        ; CODE FUER RESCAN LADEN
        CALL    ISIS        ; SYSTEMAUFRUF ISIS
        POP     D
        POP     D           ; STACK WIEDER FREIGEBEN
        ENDM
```

```
;
; ************************************************
;
; ERROR
;
ERROR  MACRO   ERRNUM
       LXI     H,-2      ; STACKPOINTER -2 SETZEN
       DAD     SP        ; STACKPOINTER
       PUSH    H         ; IN DEN STACK
       XCHG              ; STACKPOINTER NACH REG D/E
       LHLD    ERRNUM    ; FEHLERNUMMER LADEN UND
       PUSH    H         ; IN DEN STACK
       DCX     D         ; REG D/E -2 = ADR.DER
       DCX     D         ; FEHLERNUMMER
       MVI     C,12      ; CODE FUER ERROR LADEN
       CALL    ISIS      ; SYSTEMAUFRUF ISIS
       POP     D
       POP     D         ; STACK WIEDER FREIGEBEN
       ENDM
```

```
;
; ************************************************
;
; WHOCON
;
WHOCON MACRO   AFT,BUFFER
       LXI     H,-2      ; STACKPOINTER -2 SETZEN
       DAD     SP
       PUSH    H         ; STACKPOINTER IN DEN STACK
       LXI     D,BUFFER  ; ADRESSE EINES 15 BYTES LANGEN
                         ; PUFFERS
       PUSH    D         ; IN DEN STACK
       LHLD    AFT       ; WERT VON :CO: ODER :CI:
       PUSH    H         ; IN DEN STACK
       LXI     H,Ø
       DAD   SP          ; STACKPOINTER NACH REG D/E
       XCHG
       MVI   C,13        ; CODE FUER WHOCON LADEN
       CALL  ISIS        ; SYSTEMAUFRUF ISIS
       POP   D
       POP   D           ; STACK WIEDER FREIGEBEN
       POP   D
       ENDM
```

Im folgenden Beispiel wird dasselbe Programm TYPE unter Verwendung der Makroaufrufe nochmals dargestellt.

```
        EXTRN   ISIS
        CSEG
          .
          .
          .
          .
EINFUEGEN DER OBIGEN MAKRODEFINITIONEN
          .
          .
          .
          .
BEGIN:
        LXI     SP,STACK
        READ    ONE,BUFFER,COUNT,ACTUAL,STATUS
        CALL    ERRCHK
        OPEN    IAFT,BUFFER,ONE,ZERO,STATUS
        CALL    ERRCHK
LOOP:
        READ    IAFT,BUFFER,COUNT,ACTUAL,STATUS
        CALL    ERRCHK
        LHLD    ACTUAL
        MOV     A,H
        ORA     L
        JZ      DONE
        WRITE   ZERO,BUFFER,ACTUAL,STATUS
        CALL    ERRCHK
        JMP     LOOP
DONE:
        CLOSE   IAFT,STATUS
        EXIT
```

```
;
ERRCHK:
        LDA         STATUS
        ORA         A
        RZ
        ERROR       STATUS
        EXIT
;
        DSEG
;
COUNT:  DW          128
ONE:    DW          1
ZERO:   DW          Ø

BUFFER: DS          128
ACTUAL: DS          2
STATUS: DS          2
IAFT:   DS          2
;
        END         BEGIN
```

15. ISIS-II-Fehlermeldungen

Durch Systemvereinbarungen ist festgelegt, daß die Fehlernummern 1 bis einschließlich 99 für Fehler reserviert sind, die in den residenten ISIS-II-Routinen auftreten oder von diesen erkannt werden. Die Fehlernummern 101 bis 199 sind für Anwenderprogramme reserviert und die Fehlernummern 200 bis einschließlich 255 werden für Fehler benutzt, die von nicht residenten ISIS-Routinen erkannt werden. Die in der folgenden Liste mit einem Stern (*) versehenen Fehlernummern bezeichnen die nicht behebbaren Fehler. Alle anderen Fehler sind behebbar, ausgenommen die, die vom Systemaufruf CONSOLE erkannt werden.

Fehlernummer	Bedeutung
0	Kein Fehler
*1	Pufferbereich für den benötigten Puffer zu klein
2	Die Tabellennummer im AFTN-Parameter bezeichnet eine Datei, die nicht eröffnet ist
3	Versuch, mehr als 6 Dateien gleichzeitig zu eröffnen
4	Der angegebene Dateiname hat ein falsches Format
5	Das angegebene Gerät im Dateinamen ist unzulässig oder dem System nicht bekannt
6	Versuch, in eine für Eingabe eröffnete Datei zu schreiben
*7	Operation abgebrochen, weil zuwenig Platz auf der Diskette
8	Versuch, aus einer für Ausgabe eröffneten Datei zu lesen
*9	Disketteninhaltsverzeichnis zu klein
10	Die Geräteteile in den Dateinamen sind unterschiedlich
11	Eine Umbenennung des Dateinamens ist nicht möglich, da der neue Name bereits vorhanden ist
12	Versuch, eine schon eröffnete Datei zu eröffnen
13	Die angegebene Datei ist im Disketteninhaltsverzeichnis nicht eingetragen
14	Versuch, eine Datei mit dem gesetzten Attribut »Schreibschutz« zum Schreiben zu eröffnen oder umzubenennen oder zu löschen
*15	Versuch, eine Datei in den ISIS-residenten Bereich zu laden
*16	Summenfehler beim Laden
17	Versuch, eine Datei umzubenennen oder zu löschen, die auf der Diskette nicht vorhanden ist
18	Unbekannter Systemaufruf
19	Die in dem Systemaufruf SEEK angegebene Datei ist auf der Diskette nicht vorhanden
20	Mit dem Systemaufruf SEEK wurde versucht, den Zeiger über den Dateianfang hinaus zu positionieren

Fehlernummer	Bedeutung
21	Mit dem Systemaufruf RESCAN wurde versucht, eine nicht zeilenaufbereitete Datei zu lesen
22	Der Wert im ACCESS-Parameter des Systemaufrufs OPEN ist unzulässig
23	Für eine Diskettendatei ist kein Dateiname angegeben
*24	Disketten-E/A-Fehler. Eine nähere Erläuterung wird anschließend gegeben
25	Falsche Angaben für die Kontrolldatei im Systemaufruf OPEN
26	Der Wert im SWID-Parameter des Systemaufrufs ATTRIB ist unzulässig
27	Der Wert im MODE-Parameter des Systemaufrufs SEEK ist unzulässig
28	Keine Erweiterung des Dateinamens vorhanden
*29	Vom Konsoleingabegerät :CI: ist Dateiende erkannt worden
*3Ø	Laufwerk im Floppy-Disk-Speicher unklar
31	Die im Systemaufruf SEEK angegebene Datei ist für Ausgabe eröffnet
32	Versuch, eine eröffnete Datei zu löschen
33	Ein Parameter in einem Systemaufruf ist formal falsch
34	Der Wert im RETSW-Parameter des Systemaufrufs LOAD ist unzulässig
35	Versuch, mit dem Systemaufruf SEEK eine für Eingabe eröffnete Datei über das Dateiende hinaus zu erweitern

Hat das System einen Disketten-E/A-Fehler erkannt (Fehlernummer 24), wird eine zusätzliche Meldung über die Konsole ausgegeben.

FDCC = ØØ nn

nn kann folgende Werte annehmen:
Ø1 gelöschter Satz
Ø2 CRC-Fehler (Datenfeld)
Ø3 Falscher Adreßzeiger
Ø4 SEEK-Fehler
Ø8 falsche Adresse
ØA CRC-Fehler (Identifikationsfeld)
ØE kein Adreßzeiger
ØF falscher Wert im Adreßzeiger
1Ø Datenüber- oder Unterlauf
2Ø Schreibschutz
4Ø Schreibfehler
8Ø unklar

Unsere Geschäftsstellen

Bundesrepublik Deutschland und Berlin (West)

Siemens AG
Salzufer 6–8
Postfach 11 05 60
1000 Berlin 11
☎ (030) 39 39-1, Tx 1 83 766

Siemens AG
Contrescarpe 72
Postfach 10 78 27
2800 Bremen 1
☎ (0421) 3 64-1, Tx 2 45 451

Siemens AG
Märkische Straße 8–14
Postfach 6 58
4600 Dortmund 1
☎ (0231) 54 90-1, Tx 8 22 312

Siemens AG
Lahnweg 10
Postfach 11 15
4000 Düsseldorf 1
☎ (0211) 30 30-1, Tx 8 581 301

Siemens AG
Kruppstraße 16
Postfach 22
4300 Essen 1
☎ (0201) 20 13-1, Tx 8 57 437

Siemens AG
Gutleutstraße 31
Postfach 41 83
6000 Frankfurt 1
☎ (0611) 2 62-1, Tx 4 14 131

Siemens AG
Lindenplatz 2
Postfach 10 56 09
2000 Hamburg 1
☎ (040) 2 82-1, Tx 2 162 721

Siemens AG
Am Maschpark 1
Postfach 53 29
3000 Hannover 1
☎ (0511) 1 99-1, Tx 9 22 333

Siemens AG
Franz-Geuer-Straße 10
Postfach 30 11 66
5000 Köln 30
☎ (0221) 5 76-1, Tx 8 881 005

Siemens AG
N 7, 18 (Siemenshaus)
Postfach 20 24
6800 Mannheim 1
☎ (0621) 2 96-1, Tx 4 62 261

Siemens AG
Richard-Strauss-Straße 76
Postfach 20 21 09
8000 München 2
☎ (089) 92 21-1, Tx 5 29 421

Siemens AG
Von-der-Tann-Straße 30
Postfach 24 29
8500 Nürnberg 1
☎ (0911) 6 54-1, Tx 6 22 251

Siemens AG
Martin-Luther-Straße 25
Postfach 3 59
6600 Saarbrücken 3
☎ (0681) 30 08-1, Tx 4 421 431

Siemens AG
Geschwister-Scholl-Straße 24
Postfach 1 20
7000 Stuttgart 1
☎ (0711) 20 76-1, Tx 7 23 941

Europa

Belgien
Siemens S.A.
chaussée de Charleroi 116
B-1060 Bruxelles
☎ (02) 5 37 31 00, Tx 21 347

Bulgarien
RUEN,
Technisches Beratungsbüro
der Siemens Aktiengesellschaft
uliza Nikolai Gogol 5,
agal Boulevard Lenin
BG-1504 Sofia
☎ 45 70 82, Tx 22 763

Dänemark
Siemens A/S
Borupvang 3
DK-2750 Ballerup
☎ (02) 65 65 65, Tx 35 313

Finnland
Siemens Osakeyhtiö
Mikonkatu 8
Fach 8
SF-00101 Helsinki 10
☎ (90) 16 26-1, Tx 12 465

Frankreich
Siemens Société Anonyme
39–47, boulevard Ornano
B.P. 109
F-93203 Saint-Denis CEDEX 1
☎ (16-1) 8 20 61 20, Tx 620 853

Griechenland
Siemens Hellas E.A.E.
Voulis 7
P.O.B. 601
Athen 125
☎ (021) 32 93-1, Tx 2 16 291

Großbritannien
Siemens Limited
Siemens House
Windmill Road
Sunbury-on-Thames
Middlesex TW 16 7HS
☎ (09327) 85 691, Tx 89 51 091

Irland
Siemens Limited
8, Raglan Road
Dublin 4
☎ (01) 68 47 27, Tx 5341

Island
Smith & Norland H/F
Nóatún 4
P.O.B. 519
Reykjavik
☎ 2 83 22, Tx 2055

Italien
Siemens Elettra S.p.A.
Via Fabio Filzi, 25/A
Casella Postale 41 83
I-20124 Milano
☎ (02) 62 48, Tx 36 261

Jugoslawien
Generalexport
Masarikova 5/XV
Poštanski fah 223
YU-11001 Beograd
☎ (011) 68 48 66, Tx 11287

Luxemburg
Siemens Société Anonyme
17, rue Glesener
B.P. 1701
Luxembourg
☎ 4 97 11-1, Tx 3430

Niederlande
Siemens Nederland N.V.
Wilhelmina van Pruisenweg 26
Postbus 16068
Den Haag 2500
☎ (070) 78 27 82, Tx 31 373

Norwegen
Siemens A/S
Østre Aker vei 90
Postboks 10, Veitvet
N-Oslo 5
☎ (02) 15 30 90, Tx 18 477

Österreich
Siemens Aktiengesellschaft
Österreich
Apostelgasse 12
Postfach 326
A-1031 Wien
☎ (0222) 72 93-0, Tx 11 866

Polen
PHZ Transactor S.A.
ul. Stawki 2
P.O.B. 276
PL-00-950 Warszawa
☎ 39 89 10, Tx 81 32 288

Portugal
Siemens S.A.R.L.
Avenida Almirante Reis, 65
Apartado 1380
Lisboa 1
☎ (019) 53 88 05, Tx 12 563

Rumänien
Siemens birou
de consultaţii tehnice
Strada Edgar-Quinet 1
R-7 Bucureşti 1
☎ 15 18 25, Tx 11 473

Schweden
Siemens Aktiebolag
Avd. elektronikkomponenter
Norra Stationsgatan 69
Stockholm
(Fack, S-104 35 Stockholm)
☎ (08) 24 17 00, Tx 116 72

Schweiz
Siemens-Albis AG
Freilagerstraße 28
Postfach
CH-8047 Zürich
☎ (01) 2 47 31 11, Tx 52 131

Spanien
Siemens S.A.
Sede Central
Orense, 2
Apartado 155
Madrid 20
☎ (91) 4 55 25 00, Tx 27 769

Tschechoslowakei
EFEKTIM,
Technisches Büro Siemens AG
Anglická ulice 22
P.O.B. 1087
CS-120000 Praha 2
☎ 25 84 17, Tx 122 389

Türkei
Elektrik Tesiat ve Mühendislik A.Ş.
Meclisi Mebusan Caddesi,
55/35, Findikli
P.K. 64, Tophane
Istanbul
☎ 45 20 90, Tx 22 290

Ungarn
Intercooperation AG,
Siemens Kooperationsbüro
Böszörményi út 9–11
P.O.B. 1525
H-1126 Budapest
☎ (01) 15 49 70, Tx 224 133

Union der Sozialistischen Sowjetrepubliken
Vertretung der Siemens AG
Kurssowoj Pereulok, Dom 1/1,
Kwartira 4,
Wchod Sojmonowskij Projezd
Postf. 77, Internationales Postamt
SU-Moskau G 34
☎ 2 02 77 11, Tx 7413

Afrika

Ägypten
Siemens Resident Engineers
6, Salah El Din Street, Zamalek
P.O.B. 775
Cairo
☎ 81 72 28, Tx 321

Algerien
Siemens Algérie S.A.R.L.
3, Viaduc du Duc des Cars
B.P. 224, Alger-Gare
Alger
☎ 63 95 47/51, Tx 52 817

Äthiopien
Siemens Ethiopia Ltd.
Ras Bitwoded Makonen Building
P.O.B. 5505
Addis Ababa
☎ 15 15 99, Tx 21 052

Libyen
Assem Azzabi
17, 1st September Street,
Tariq Building
P.O.B. 2583
Tripoli
☎ 4 15 34, Tx 20 029

Marokko
SETEL S.A.
km 1, Route de Rabat
Casablanca-Ain Sebâa
☎ 35 10 25, Tx 21 914

Nigeria
Siemens Nigeria Limited
Industrial Estate 3 f,
Block A
P.O.B. 304
Lagos (Oshodi)
☎ 4 19 20, Tx 21 357

Südafrika
Siemens Limited
Siemens House,
Corner Wolmarans and
Biccard Streets, Braamfontein
P.O.B. 45 83
Johannesburg 2000
☎ (011) 7 15 91 11, Tx 58-7721

Sudan
National Electrical
& Commercial Company
Murad Sons Building,
Barlaman Street
P.O.B. 12 02
Khartoum
☎ 8 08 18, Tx 642

Tunesien
Sitelec S.A.,
Société d'Importation
et de Travaux d'Electricité
26, Avenue Farhat Hached
Tunis
☎ 24 28 60, Tx 12 326

Zaire
Siemens Zaire S.P.R.L.
1222, Avenue Tombalbaye
B.P. 98 97
Kinshasa 1
☎ 2 26 08, Tx 21 377

Amerika

Argentinien
Siemens Sociedad Anónima
Avenida Pte. Julio A. Roca 516
Casilla Correo Central 12 32
RA-1067 Buenos Aires
☎ 30 04 11, Tx 121 812

Bolivien
Sociedad Comercial é Industrial
Hansa Limitada
Calle Mercado esquina Yanacocha
Cajón Postal 14 02
La Paz
☎ 5 44 25, Tx 5261

Brasilien
ICOTRON S.A., Indústria de
Componentes Electrônicos
Avenida Mutinga, 3716
Caixa Postal 1375
BR-05110 São Paulo 1
☎ (011) 2 61 02 11
Tx 11-23 633, 11-23 641

Chile
Gildemeister S.A.C.,
Area Siemens
Amunátegui 178
Casilla 99-D
Santiago de Chile
☎ 8 25 23
Tx TRA SGO 392, TDE 40 588

Ecuador
Siemens S.A.
Avenida América y
Hernández Girón s/n.,
Sector 28
Casilla 35 80
Quito
☎ 24 53 63, Tx 22 190

Kanada
Siemens Canada Limited
Montreal Office
7300 Trans-Canada Highway
P.O.B. 7300
Pointe Claire, Québec H9R 4R6
☎ (514) 6 95 73 00, Tx 5 267 300

Kolumbien
Siemens S.A.
Carrera 65, No. 11–83
Apartado Aéreo 8 01 50
Bogotá 6
☎ 61 04 77, Tx 44 750

Mexico
Siemens S.A.
Poniente 116, No. 590
Apartado Postal 1 50 64
México 15, D.F.
☎ 5 67 07 22, Tx 1 772 700

Uruguay
Conatel S.A.
Ejido 1690
Casilla de Correo 13 71
Montevideo
☎ 91 73 31, Tx 934

Venezuela
Siemens S.A.
Avenida Principal,
Urbanización Los Ruices
Apartado 36 16
Caracas 101
☎ (02) 34 85 31, Tx 25 131

Vereinigte Staaten von Amerika
Siemens Corporation
186 Wood Avenue South
Iselin, New Jersey 08 830
☎ (201) 4 94-1000
Tx WU 844 491
TWX WU 710 998 0588

Asien

Afghanistan
Afghan Electrical Engineering
and Equipment Limited
Alaudin, Karte 3
P.O.B. 7
Kabul 1
☎ 4 04 46, Tx 35

Bangladesch
Siemens Bangladesh Ltd.
74, Dilkusha Commercial Area
P.O.B. 33
Dacca 2
☎ 24 43 81, Tx 824

Hongkong
Jebsen & Co., Ltd.
Prince's Building, 23rd floor
P.O.B. 97
Hong Kong
☎ 5 22 51 11, Tx 73 221

Indien
Siemens India Ltd.
134A, Dr. Annie Besant Road, Worli
P.O.B. 65 97
Bombay 400018
☎ 37 99 06, Tx 112 373

Indonesien
P.T. Siemens Indonesia
Kebon Sirih 4
P.O.B. 24 69
Jakarta
☎ 5 10 51, Tx 46 222

Irak
Samhiry Bros. Co. (W.L.L.)
Abu Nawas Street
P.O.B. 300
Baghdad
☎ 9 00 21, Tx 2255

Iran
Siemens Sherkate S. (K.)
Khiabane Takhte Djamshid 32,
Siemenshaus
Teheran 15
☎ (021) 6 14-1, Tx 212 351

Japan
Nippon Siemens K.K.
Furukawa Sogo Building,
6–1, Marunouchi 2-chome,
Chiyoda-ku
Central P.O.B. 11 44
Tokyo 100-91
☎ (03) 2 14 02 11, Tx 22 808

Jemen (Arab. Republik)
Tihama Tractors
& Engineering Co. Ltd.
P.O.B. 49
Sanaa
☎ 24 62, Tx 217

Korea (Republik)
Siemens Electrical
Engineering Co., Ltd.
Daehan Building, 8th floor,
75, Susomun-dong, Chung-ku
C.P.O.B. 30 01
Seoul
☎ 7 77 75 58, Tx 23 229

Kuwait
Abdul Aziz M. T. Alghanim Co.
& Partners
Abdulla Fahad Al-Mishan Building
Al-Sour Street
P.O.B. 32 04
Kuwait, Arabia
☎ 42 33 36, Tx 21 31

Libanon
Ets. F. A. Kettaneh S.A.
(Kettaneh Frères)
Rue du Port, Immeuble Fattal
P.B. 11 02 42
Beyrouth
☎ 22 11 80, Tx 20 614

Malaysia
Guthrie Engineering (Malaysia)
Sdn. Bhd.,
Electrical &
Communications Division
17, Jalan Semangat
P.O.B. 30
Petaling Jaya
☎ 77 33 44, Tx 37 573

Pakistan
Siemens Pakistan Engineering
Co. Ltd.
38, Davis Road
P.O.B. 7158
Lahore
☎ 51 60 61, Tx 820

Philippinen
Engineering Equipment, Inc.,
Machinery Division,
Siemens Department
2280 Pasong Tamo Extension
P.O.B. 71 60,
Airmail Exchange Office,
Manila International Airport,
Philippines 31 20
Makati, Rizal
☎ 85 40 11/19,
Tx RCA 7222 382, EEC 3695

Saudi-Arabien
E. A. Juffali & Bros.
Head Office
King Abdul-Aziz-Street
P.O.B. 10 49
Jeddah
☎ 2 22 22, Tx 40 130

Singapur
Siemens Components PTe. Ltd.
Promotion Office
19B – 45B, Jalan Tenteram
Singapore 12
☎ 55 08 11, Tx 21 000

Syrien
Syrian Import Export & Distribution
Co., S.A.S. SIEDCO
Port Saïd Street
P.O.B. 363
Damas
☎ 1 34 31, Tx 11 267

Taiwan
Delta Engineering Ltd.
42, Hsu Chang Street, 8th floor
P.O.B. 5 84 97
Taipei
☎ 3 11 47 31, Tx 21 826

Thailand
B. Grimm & Co., R.O.P.
1643/4, Petchburi Road (Extension)
P.O.B. 66
Bangkok 10
☎ 2 52 40 81, Tx 26 14

Australien und Ozeanien

Australien
Siemens Industries Limited
Melbourne Office
544 Church Street
Richmond, Vic. 3121
☎ (03) 4 29 71 11, Tx 30 425

SIEMENS

Bestell-Nr. B 1848
Printed in Germany
KG 47810.