1 Einleitung

Der EDT (EDITOR) dient zur Dateiaufbereitung. Er bearbeitet SAM- und ISAM-Dateien, Elemente aus Programm-Bibliotheken und POSIX-Dateien.

Mit dem EDT kann der Benutzer Dateien und Bibliothekselemente

- eröffnen, neu erstellen, schließen und speichern
- ändern (durch Löschen, Einfügen und Ändern von Daten),
- nach bestimmten Daten durchsuchen
- miteinander vergleichen
- auf dem Bildschirm oder dem Drucker ausgeben

Der EDT bietet für die Datenbearbeitung folgende Möglichkeiten:

1. Virtuelle Bearbeitung von Dateien und Bibliothekselementen im Dialog

   a) Erstellen und Bearbeiten im Benutzeradreßraum.

   b) Schreiben und Speichern einer Datei oder eines Bibliothekselementes vom Benutzeradreßraum auf Platte oder Band.
   Das Arbeiten im Benutzeradreßraum hat die Vorteile, daß
   - die Datei während der Bearbeitung geschlossen ist
   - die Zahl der Plattenzugriffe äußerst gering ist

2. Reale Bearbeitung von Dateien im Dialog
   Die Dateien können direkt auf Platte bearbeitet werden.

3. Bearbeitung von Dateien und Bibliothekselementen mit EDT-Prozeduren
   Dateibearbeitungen, die häufig in gleicher oder ähnlicher Form auszuführen sind, lassen sich mit EDT-Prozeduren programmieren.

4. Bearbeitung im Stapelbetrieb
   Obwohl der EDT als Dialogprogramm konzipiert ist, kann er Dateien und Bibliothekselemente im Stapelbetrieb virtuell oder real bearbeiten.

Der EDT kann

- ein anderes Programm als Unterprogramm aufrufen
- von einem Benutzerprogramm als Unterprogramm aufgerufen werden

1.1 Konzept der EDT-Dokumentation

Die vollständige Dokumentation des EDT besteht aus den Handbüchern:

- Anweisungen
- Unterprogrammschnittstellen
- Anweisungsformate (Tabellenheft)
- EDT-Operanden (Referenzkarte)
- Additional Information for Arabic (nur in englischer Sprache)
- Additional Information for Farsi (nur in englischer Sprache)

Das Handbuch zu den EDT-Anweisungen beschreibt alle Anweisungen des EDT und sollte für jeden EDT-Anwender zugänglich sein. Es dient, neben einem kleinen Einstieg in den EDT, vornehmlich als Nachschlagewerk für die zahlreichen Anweisungen des EDT.

Das Handbuch zu den EDT-Unterprogrammschnittstellen beschreibt die Unterprogamm-schnittstellen des EDT. Es kann nur in Verbindung mit dem Handbuch zu den EDT-Anwei-sungen sinnvoll genutzt werden.

Das Tabellenheft enthält eine Kurzbeschreibung aller EDT-Anweisungen.

1.2 Zielgruppen der EDT-Handbücher

Während sich das Handbuch zu den EDT-Anweisungen an den EDT-Einsteiger und den EDT-Anwender richtet, wendet sich dasHandbuch zu den EDT-Unterprogrammschnittstel-len auf den Kreis der erfahrenen EDT-Anwender und Programmierer, die den EDT in eige-ne Programme einbinden wollen.

Dieses Handbuch zu den EDT-Anweisungen richtet sich an den Benutzer, der den EDT noch nicht kennt, bis hin zum erfahrenen EDT-Anwender, für den vor allem das Kapitel „Anweisungen des EDT“ auf Seite 147ff. mit den Beschreibungen aller EDT-Anwei-sungen ein notwendiges Nachschlagewerk darstellt. Der EDT-Anwender, der selber EDT-Prozeduren schreiben oder bestehende EDT-Prozeduren warten will, findet im Kapitel „EDT-Prozeduren“ auf Seite 127ff. einen wertvollen Einstieg in das Prozeduren-schreiben mit dem EDT.

Zum Aufruf des EDT sollten Sie mit den wichtigsten BS2000-Kommandos vertraut sein.

1.3 Konzept des Handbuches EDT-Anweisungen

Dieses Handbuch beschreibt nach einer Einführung in den EDT die Bearbeitung von Dateien und Bibliothekselementen und die Anwendung und Erstellung von EDT-Prozeduren und gibt einen Überblick über alle EDT-Anweisungen mit einer detaillierten Beschreibung und einer Vielzahl von Beispielen.

Dieses Handbuch enthält folgende Kapitel:

– **Einführung in den EDT**

  Kleiner Einstieg für den Benutzer, der den EDT noch nicht kennt.

– **Anwendung des EDT**

  Aufruf und Beenden des EDT. Dateibearbeitung von Dateien und Bibliothekselementen.

– **Arbeitsmodi des EDT**

  Dateibearbeitung im F-Modus: Bildschirmorientiertes Arbeiten mit dem EDT, Beschreibung der Kurzanweisungen und der Anweisungen, die ausschließlich im F-Modus benutzt werden können.
  Dateibearbeitung im L-Modus.

– **EDT-Prozeduren**

  Anwendung von EDT-Prozeduren. Erstellen, Aufruf und Ablauf von EDT-Prozeduren.

– **Anweisungen des EDT**

  Anweisungen des EDT in alphabetischer Reihenfolge mit zahlreichen Beispielen. Übersicht der EDT-Operanden.

– **Meldungen des EDT**

  Auflistung aller Meldungen des EDT mit ihren Bedeutungen und Maßnahmen.

– **Installationshinweise**

  Installationshinweise für den Systemverwalter.

Die genaue Beschreibung der Unterprogrammschnittstellen des EDT finden Sie im Handbuch:

EDT (BS2000) V16.5A
Unterprogrammschnittstellen
Benutzerhandbuch

Die Kurzbeschreibung aller EDT-Anweisungen finden Sie im Handbuch:

EDT (BS2000) V16.5
Anweisungsformate
Tabellenheft

Die Erweiterungen für die Sprache Arabisch finden Sie im Handbuch (nur in englischer Sprache erhältlich):

EDT (BS2000) V16.5A
Additional Information for Arabic
Supplement

Die Erweiterungen für die Sprache Farsi finden Sie im Handbuch (nur in englischer Sprache erhältlich):

EDT (BS2000) V16.5A
Additional Information for Farsi
Supplement

Readme-Datei

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei. Sie finden die Readme-Datei auf Ihrem BS2000-Rechner unter dem Dateinamen `SYSRME.produkt.version.sprache`. Die USERID, unter der sich die Readme-Datei befindet, erfragen Sie bitte bei Ihrem zuständigen Systemverwalter. Die Readme-Datei können Sie mit dem Kommando SHOW-FILE oder mit einem Editor ansehen oder auf einem Standarddrucker mit folgendem Kommando ausdrucken:

```
/PRINT-FILE FILE-NAME=dateiname,LAYOUT-CONTROL=
    PARAMETERS(CONTROL-CHARACTERS=EBCDIC)
```

1.4 Änderungen gegenüber EDT V16.4

Im folgenden wird ein Überblick über die neuen bzw. erweiterten Funktionen des EDT V16.5 gegeben.

Autosave-Funktion

Damit in Bearbeitung befindliche Arbeitsdateien bei einem Systemabsturz oder Leitungsverlust nicht verloren gehen, kann ein automatisches Sichern der Arbeitsdateien eingeschaltet werden. Die Zeitdifferenz zwischen den Sicherungen und die Namen der von EDT erzeugten Sicherungsdateien können vom Anwender vorgegeben werden.

@AUTOSAVE
@STATUS

Umschalten der Syntaxanalyse im L-Modus und Beenden des EDT-Laufs

Der EDT bietet die Möglichkeit, die Strenge der Syntaxprüfung im L-Modus zu bestimmen und einen Test-Modus einzuschalten, in dem nur die Syntax geprüft wird, ohne die Anweisungen auszuführen.

Anhand der Fehlerklasse im Rückkehrcode der überwachenden Monitor-Jobvariable kann festgestellt werden, ob im EDT-Lauf ein Syntaxfehler aufgetreten ist. Bei Beendigung des EDT wird der Kommando-Returncode gesetzt, der in S-Prozeduren (SDF-P) zur Steuerung verwendet werden kann.

@SYNTAX
@STATUS
Beenden des EDT-Laufs

Erweitertes Logging und Anweisungswiederholung

Die Protokollierung der Anweisungen ist auch im Dialog möglich. Es können protokolliert werden

– im L-Modus Anweisungen und Datenzeilen
– im F-Modus die Eingaben in der Anweisungszeile

Es kann angegeben werden, ob die Ausgabe auf SYSLST, eine der SYSLSTxx zugeordneten Datei oder in eine SDF-P-Listenvariable erfolgen soll.

EDT gibt nicht nur die letzte Anweisung im F-Modus-Dialog wieder aus, sondern auch vorhergehende. Die Tiefe, bis zu der Anweisungen wieder ausgegeben werden, hängt von der Länge der eingegebenen Anweisungen ab.

@LOG
Anweisungswiederholung

EDT und SDF

Wird mit dem EDT eine Systemprozedur geschrieben, so kann mittels SDF die Syntax der eingegebenen Kommandos geprüft werden, ohne den EDT zu verlassen. Ist ein Korrektur-Dialog möglich, werden die im geführten SDF-Dialog eingegebenen Operanden in den EDT-Datenbereich übernommen. Bei Angabe eines Programmnamens, dem eine SDF-Syntax-Datei zugeordnet ist, werden auch Anweisungen syntaktisch geprüft.

@SDFTEST
Kurzanweisung T
@PAR
@STATUS

Erweiterung der SDF-P-Unterstützung

Es können auch S-Variablen vom Typ INTEGER definiert, gelesen und beschrieben werden. Eine zusammengesetzte S-Variable vom Typ LIST kann beschrieben und erweitert werden. Ihre Elemente können in den EDT-Datenbereich übernommen werden.

@GETVAR
@SETVAR
@GETLIST
@SETLIST

Bei Beenden des EDT-Laufs wird ein Kommando-Returncode gesetzt, der in S-Prozeduren abgefragt werden kann.

SDF-P-Listenvariablen können zur Protokollierung verwendet werden.

@LOG
Beenden des EDT-Laufs

Sicheres Umnumerieren

Um zu verhindern, daß beim Umnumerieren ungewollt Zeilen verloren gehen, wenn die Anzahl der Zeilen größer wird als die größtmögliche Zeilennummer, wird im Dialog bei der Ausführung von @RENUMBER beim Erreichen der höchsten Zeilennummer eine Sicherheitsabfrage ausgegeben. @RENUMBER wird nur bei positiver Antwort ausgeführt. Information über die Anzahl der Datenzeilen kann mit @LIMITS abgefragt werden.

@RENUMBER
@LIMITS

Dateibearbeitung

Dateien, deren Dateiattribute von den Standardwerten abweichen, können ohne vorhergehendes SET-FILE-LINK-Kommando eröffnet werden.

@OPEN Format 2

EDT und POSIX (ab BS2000/OSD-BC V2.0)

Wenn das POSIX-Subsystem aktiviert ist, können UFS/POSIX-Dateien, die im POSIX-Dateisystem abgelegt sind, eröffnet, gelesen, geschrieben und geschlossen werden.

@XOPEN
@XCOPY
@XWRITE
@CLOSE
@PAR
@STATUS
@IEDTPARL (EDT-Unterprogrammschnittstelle)

Zur Unterstützung der ASCII-Codierung bei POSIX-Dateien bietet der EDT

– einen Eingabemodus für den L-Modus
– eine Darstellungsform im Hexadezimal-Modus, die die ASCII-Werte zeigt.

@INPUT Format 3
@PAR CODE
@BLOCK

Bibliotheksbearbeitung

Bibliothekselemente mit freien Typnamen (maximal 8-stellig) können eröffnet, gelesen und zurückgeschrieben werden, wenn sie editierbar sind. Sie werden mit @SHOW im Inhaltsverzeichnis angezeigt und können mit @DELETE Format 2 gelöscht werden.

Bibliothekselemente vom Typ U und F können im Inhaltsverzeichnis angezeigt werden und gelöscht werden.

@SHOW
@PAR
@STATUS
@OPEN Format 2
@COPY Format 2
@WRITE Format 2
@INPUT Format 2
IEDTPARL (EDT-Unterprogrammschnittstelle)

Starten des EDT-Laufs

Der EDT wird mit der Syntaxdefinition eines Kommandos START-EDT ausgeliefert (ab BS2000/OSD-BC V2.0).

Es kann die gewünschte EDT-Version angegeben werden. Beim Nachladen aus der „EDTLIB“ und an der Unterprogrammschnittstelle wird die Version geprüft.

IEDTINF (EDT-Unterprogrammschnittstelle)
IEDTGLE (EDT-Unterprogrammschnittstelle)
Starten des EDT

Beenden des EDT-Laufs

Im F-Modus-Dialog und im L-Modus-Dialog kann in der Arbeitsdatei 0 mit der Anweisung @END der EDT-Lauf beendet werden.

@END

Informationen über den EDT-Lauf werden in einer überwachenden Monitor-Jobvariablen abgelegt. In einer S-Prozedur kann der Kommando-Returncode des EDT zur weiteren Steuerung abgefragt werden.

Beenden des EDT-Laufs
@HALT

Weitere Anweisungen mit neuen Operanden

@PRINT
@LIST
@INPUT
@PAR
@IF
@PROC
@END

Eliminieren fester Meldungen

Wird der EDT an der alten Line-Modus-Unterprogrammschnittstelle aufgerufen, werden auch Abbruchmeldungen über MSG7 ausgegeben.

Eliminieren fester Dateinamen

Wenn IMON installiert ist, werden die Namen der Dateien, die zum Produkt EDT gehören, dynamisch versorgt.

1.5 Verwendete Metasprache

In diesem Handbuch werden folgende Darstellungsmittel verwendet:

„Anführungszeichen“	Kapitelnamen und Begriffe, die hervorgehoben werden sollen.
␣	Leerzeichen.
[Zahl]	Verweis auf ein Handbuch im Literaturverzeichnis.
Taste	Symbolisiert eine Taste auf der Tastatur.
	Hinweis für zusätzliche Informationen.
	Warnhinweis, z.B. Warnung vor Datenverlust.

Die Beschreibung der Syntax und der EDT-Operanden siehe Abschnitt „Beschreibung der Syntax“ auf Seite 147ff. und Abschnitt „Beschreibung der Operanden“ auf Seite 150ff.

2 Einführung in den EDT

Dieses Kapitel wendet sich an den Benutzer, der den EDT noch nicht kennt.

Es behandelt nur ausgewählte Funktionen, um den Einstieg in das Arbeitsprinzip und die Handhabung des EDT zu erleichtern.

Der EDT ist ein Hilfsmittel zur rationellen Texterstellung und Textaufbereitung.

Mit dem EDT lassen sich:

– Dateien und Bibliothekselemente erzeugen und aufbauen
– Daten eingeben, ändern, einfügen und löschen
– Dateien und Bibliothekselemente auf Platte schreiben sowie von Platte lesen
– Daten in einer Datei oder einem Bibliothekselement suchen
– Daten auf den Bildschirm oder auf den Drucker ausgeben

Der Benutzer kann mit dem EDT Dateien oder Bibliothekselemente für eine Vielzahl von Anwendungen bearbeiten, z.B.:

– Textdateien oder Tabellen (für Buchhaltung, Lösungsstudien, ...)
– Quellprogramme
– Daten für Testläufe von Programmen
– Daten für Produktivläufe von Programmen
– Arbeitsdateien

2.1 Arbeitsweise des EDT

Mit dem EDT lassen sich Dateien (SAM, ISAM, POSIX) und Bibliothekselemente erzeugen und verändern. Diese Dateibearbeitung erfolgt innerhalb von **3 Speicherbereichen**:

– dem auf dem Bildschirm dargestellten **Arbeitsfenster**,

– dem Arbeitsbereich des EDT im virtuellen Speicher, im folgenden als **Arbeitsdatei** bezeichnet,

– dem **gemeinschaftlichen Speicher** auf Platte.

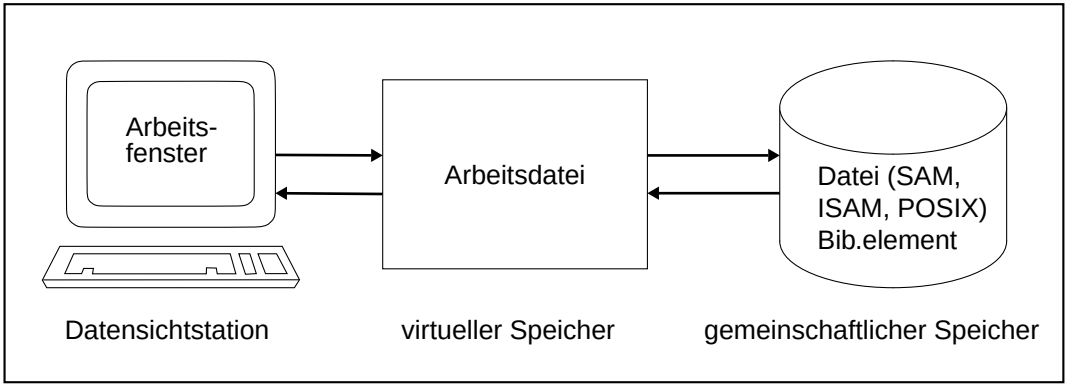


Bild 1: Speicherbereiche des EDT

Der EDT legt im virtuellen Speicher einen Arbeitsbereich an, die sogenannte Arbeitsdatei. Beim Aufruf des EDT ist diese Arbeitsdatei leer.

Soll eine Datei oder ein Bibliothekselement erstellt werden, werden die Daten im Arbeitsfenster eingegeben. Mit Funktionstasten wird das Arbeitsfenster in die Arbeitsdatei übertragen. Bei Abschluß der Dateneingabe wird der Inhalt der Arbeitsdatei über Anweisungen als SAM- oder ISAM- oder POSIX-Datei oder als Bibliothekselement auf Platte geschrieben.

Ist eine existierende Datei oder ein existierendes Bibliothekselement zu ändern, muß die Datei bzw. das Bibliothekselement zuerst in die Arbeitsdatei eingelesen werden. Mit dem Einlesen werden die ersten 23 Zeilen dieser Arbeitsdatei am Arbeitsfenster dargestellt. Die Plattendatei bleibt unverändert erhalten.

In der Arbeitsdatei werden über Anweisungen die geforderten Funktionen, wie z.B. Einfügen, Ändern und Löschen ausgeführt. Die Änderungen werden im Arbeitsfenster gespeichert und angezeigt. Mit DUE wird der Inhalt des Arbeitsfensters in die Arbeitsdatei übertragen. Textneueingaben und Korrekturen werden an der richtigen Stelle eingeordnet. Die korrigierte Arbeitsdatei wird über eine Anweisung in die Plattendatei zurückgeschrieben. Der ursprüngliche Inhalt der Plattendatei wird dabei überschrieben.

Bei jeder Übertragung der Daten von einem Speicher in einen anderen bleibt jeweils die ursprüngliche Information im sendenden Speicher erhalten. Sie steht dann in beiden Speicherbereichen. Bei Bearbeitungsfehlern kann man deswegen immer auf die ursprünglichen Daten zurückgreifen und den Fehler korrigieren.

Ändert man z.B. eine Datei, die von Platte in die Arbeitsdatei gelesen wurde, kann man bei einem Bearbeitungsfehler jederzeit auf die Plattendatei zurückgreifen. Ebenso kann man vom Bildschirm aus jederzeit wieder auf die in der Arbeitsdatei vorhandenen Daten zurückgreifen und am Bildschirm gemachte Korrekturen verwerfen.

2.2 Arbeiten mit dem EDT

2.2.1 EDT-Bildschirm

Der Dateibearbeiter EDT wird mit dem Systemkommando **START-PROGRAM $EDT** oder **START-EDT** (ab BS2000/OSD-BC V2.0) aufgerufen.

Auf dem Bildschirm erscheint das leere Arbeitsfenster.

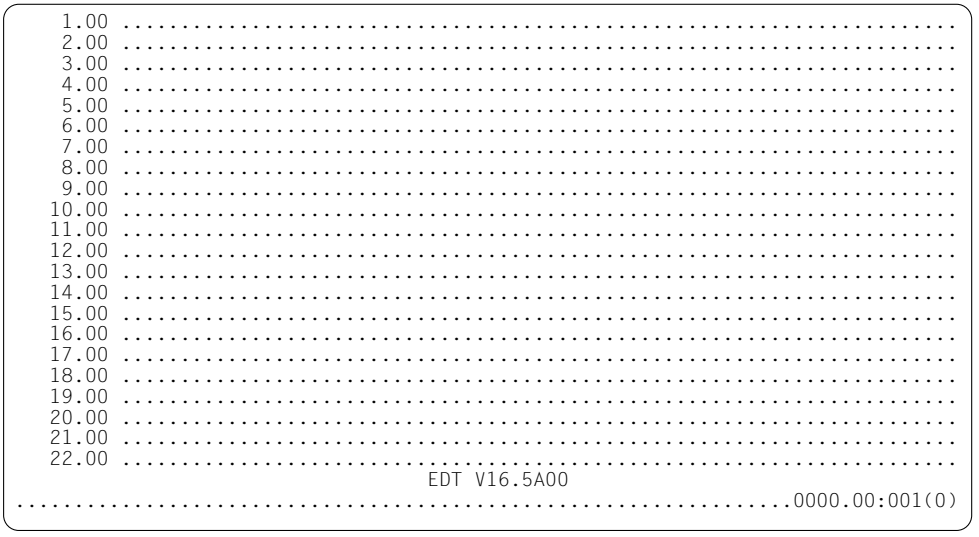


Bild 2: Arbeitsfenster des EDT nach dem Aufruf

Das Arbeitsfenster des EDT besteht aus 5 Teilbereichen:

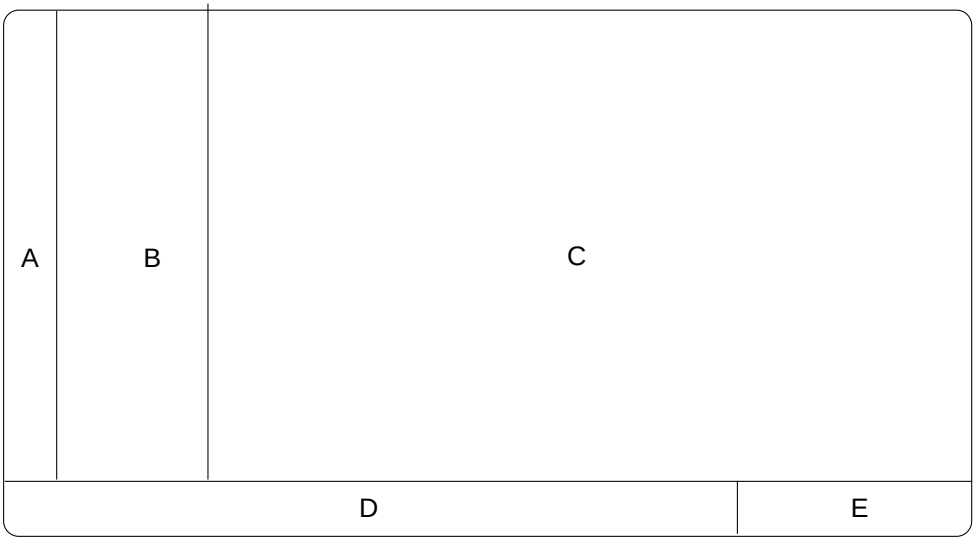


Bild 3: Teilbereiche des EDT-Arbeitsfensters

A **Markierungsspalte** (erste Bildschirmspalte)
Durch Eingeben bestimmter Zeichen in die Markierungsspalte können Zeilen des Arbeitsfensters für bestimmte Anwendungsfälle (s.u.) gekennzeichnet werden.

B **Zeilennummernanzeige**
Hier werden die Zeilennummern des Textes im Datenfenster angezeigt.

C **Datenfenster**
Hier werden die Datensätze eingegeben bzw. angezeigt.

D **Anweisungszeile** (letzte Bildschirmzeile)
Anweisungen an den EDT müssen in der Anweisungszeile eingegeben werden.

E **Zustandsanzeige**
Im ersten Teil der Zustandsanzeige wird die Zeilennummer der ersten Zeile des Datenfensters angezeigt. Von der Zeilennummer durch einen Doppelpunkt getrennt folgt die Spaltennummer. Im letzten Teil der Zustandsanzeige wird in runden Klammern die Nummer der Arbeitsdatei angezeigt.

Der EDT arbeitet **bildschirmorientiert**, d.h. der Benutzer kann über Anweisungen einen beliebigen Teil einer Datei in das Datenfenster holen und diesen Text im Datenfenster beliebig überschreiben oder Texte ein- und ausfügen. Mit DUE werden die Daten in die Arbeitsdatei übertragen.

2.2.2 Anweisungen im EDT

Funktionen können ausgelöst werden durch:

– Anweisungen, die in der Anweisungszeile des Arbeitsfensters einzugeben sind (siehe Kapitel „Anweisungen des EDT“ auf Seite 147ff.).

– Kurzanweisungen, die in der Markierungsspalte des Arbeitsfensters einzugeben sind (siehe Abschnitt „Kurzanweisungen im F-Modus“ auf Seite 75ff.).

– Anweisungen im Datenfenster (siehe Abschnitt „Anweisung im Datenfenster - Auftrennen eines Datensatzes“ auf Seite 107ff.).

Das **Übertragen der Anweisungen** an den EDT erfolgt wahlfrei durch:

– DUE oder DUE1

  Die Anweisungen bzw. Kurzanweisungen im Arbeitsfenster werden ausgeführt.

– F2

  Die Anweisungen bzw. Kurzanweisungen werden ausgeführt. Das Datenfenster wird außerdem auf überschreibbar (hell) gestellt. Die Daten können ohne irgendwelche Anweisungen beliebig verändert werden.

Anweisungen können in Groß- oder Kleinbuchstaben in der Anweisungszeile (siehe Kapitel „Anweisungen des EDT“ auf Seite 147ff.) bzw. in der Markierungsspalte (siehe Abschnitt „Das Arbeitsfenster“ auf Seite 65ff.) eingegeben werden.

2.3 Bearbeiten von Dateien

Erfassen der Daten

Über die Tastatur werden am Datenfenster Datensätze mit beliebigem Inhalt eingegeben. Dazu wird die Schreibmarke mit ⇥ an den Anfang der ersten Zeile des Datenfensters plaziert.

Danach wird der Text in die Zeile eingegeben. Mit ⇥ springt die Schreibmarke an den Anfang der nächsten Zeile. Die Texteingabe kann fortgesetzt werden.

Wenn die Dateneingabe beendet ist oder alle Zeilen des Datenfensters beschrieben sind, muß der Text mit DUE bzw. DUE1 abgeschickt werden.

Soll weiterer Text erfaßt werden, muß + in der Anweisungszeile eingegeben und mit DUE abgeschickt werden. Auf dem Bildschirm erscheint die letzte Zeile des alten Datenfensters als ersten Zeile des neuen Datenfensters.

⇥ muß zweimal gedrückt werden. Die Schreibmarke springt an den Anfang der 2. Zeile.

Danach kann der Text wie oben eingegeben werden.

Speichern (Schreiben) einer neuerstellten Arbeitsdatei

Nach dem Erfassen wird die Arbeitsdatei mit @WRITE oder @SAVE in eine Datei oder in ein Bibliothekselement geschrieben (siehe @WRITE, @SAVE).

Erstellen (Schreiben) einer SAM-Datei auf Platte oder Band:

```
@WRITE 'dateiname'
```

Erstellen (Schreiben) einer ISAM-Datei auf Platte oder Band:

```
@SAVE 'dateiname'
```

Erstellen (Schreiben) eines Bibliothekselements:

```
@WRITE LIBRARY = libname (ELEMENT = elemname)
```

Einlesen in die Arbeitsdatei

Mit @READ bzw. @GET wird eine bestehende Datei in die Arbeitsdatei eingelesen (siehe @READ, @GET).

Einlesen einer SAM-Datei:

```
@READ 'dateiname'
```

Einlesen einer ISAM-Datei

```
@GET 'dateiname'
```

Einlesen eines Bibliothekselementes:

```
@COPY LIBRARY = libname ( ELEMENT = elemname )
```

Im Datenfenster erscheint der Anfang der Datei oder des Bibliothekselementes.

Korrigieren von Zeichen

Mit +, +n, ++, –, –n, – – wird der gewünschte Datenbereich in das Datenfenster gebracht.

– Zum Überschreibbarstellen des Datenfensters kann F2 gedrückt werden oder die Zeile mit X in der Markierungsspalte markiert werden.

  Das Datenfenster erscheint hell, d.h. es ist überschreibbar.

  Mit den Positioniertasten ist die Schreibmarke auf die zu korrigierende Stelle im Datenfenster zu bringen.

  Textkorrekturen durch Überschreiben, Einfügen und Ausfügen sind jetzt möglich.

  Mit @PAR EDIT FULL = ON kann bei eingeschalteter Zeilennummernanzeige das Datenfenster immer überschreibbar gestellt werden. Gleichzeitig können auch Kurzanweisungen in der Markierungsspalte angegeben werden.
  Ein Positionieren mit ⇥ führt dann zur Markierungsspalte in der nächsten Zeile bzw. an den Anfang der Datenzeile.

– Zum Ausfügen von Zeichen muß AFG gedrückt werden.

– Zum Einfügen von Zeichen muß EFG gedrückt werden.

  Danach können Zeichen an der gewünschten Stelle eingefügt werden. Es ist zu beachten, daß Zeichen, die dabei über den rechten Datenfensterrand geschoben werden, verloren gehen.

– Zum Ausschalten des Einfügemodus muß [RS] gedrückt werden.

  Das korrigierte Datenfenster wird mit [DUE] in die Arbeitsdatei übertragen.

  Sollen die Korrekturen nicht in die Arbeitsdatei übernommen werden, z.B. wegen Fehleingaben, kann mit [K3] der ursprüngliche Bildschirminhalt wiederhergestellt werden. Das fehlerhaft bearbeitete Datenfenster darf jedoch noch nicht mit [DUE] in die Arbeitsdatei übertragen worden sein.

Einfügen von Zeilen

Mit +, +n, ++,– , –n oder – – wird der Datenbereich, in dem Zeilen eingefügt werden sollen, in das Datenfenster gebracht (siehe oben).

Die Zeile, vor der die neuen Zeilen eingefügt werden sollen, ist in der Markierungsspalte durch eine Zahl zwischen 1 und 9 (Anzahl der einzufügenden Zeilen) zu kennzeichnen. Danach muß [DUE] gedrückt werden. Im Datenfenster erscheint die entsprechende Anzahl von Leerzeilen. Text kann wie beim Neuerstellen einer Datei in die Leerzeilen eingegeben werden. Anschließend muß [DUE] gedrückt werden.

Ist weiterer Text einzufügen, ist der Vorgang zu wiederholen (siehe auch Kurzanweisung I).

Löschen von Zeilen

Mit +, +n, ++,–, –n oder – – wird der Datenbereich, in dem Zeilen gelöscht werden sollen, in das Datenfenster gebracht.

Die zu löschenden Zeilen sind in der Markierungsspalte mit dem Buchstaben D zu kennzeichnen. Danach muß [DUE] gedrückt werden.

Die gekennzeichneten Zeilen werden in der Arbeitsdatei gelöscht. Das soeben bearbeitete Datenfenster wird wieder ausgegeben, evtl. nachfolgende Sätze werden wegen des freigewordenen Platzes im Datenfenster nachgezogen.

Speichern von Dateien und Bibliothekselementen

Geänderte Dateien bzw. Bibliothekselemente werden mit @WRITE oder @SAVE zurückgeschrieben.

2.4 Beispiel für das Bearbeiten einer Datei

In diesem Beispiel wird die SAM-Datei DROGERIE, die im gemeinschaftlichen Speicher auf Platte steht, in den virtuellen Speicher als Arbeitsdatei eingelesen. Im Datenfenster wird die Arbeitsdatei DROGERIE bearbeitet:

– In einer Zeile wird die Rubrik BESTELLT korrigiert.

– Vier Zeilen werden aus der Arbeitsdatei gelöscht, da die Artikel aus dem Angebot genommen werden sollen.

– Vier Zeilen werden in die Arbeitsdatei eingefügt und eine Zeile an die Arbeitsdatei angefügt, da fünf neue Artikel in das Angebot aufgenommen werden sollen.

Die im Datenfenster veränderte Arbeitsdatei DROGERIE wird anschließend auf Platte zurückgeschrieben. Dabei wird der ursprüngliche Zustand der Datei DROGERIE durch die veränderte Arbeitsdatei überschrieben.

```
/start-program $edt
```

```
   1.00 ..........................................................................
   2.00 ..........................................................................
   3.00 ..........................................................................
   4.00 ..........................................................................
   5.00 ..........................................................................
   6.00 ..........................................................................
   7.00 ..........................................................................
   8.00 ..........................................................................
   9.00 ..........................................................................
  10.00 ..........................................................................
  11.00 ..........................................................................
  12.00 ..........................................................................
  13.00 ..........................................................................
  14.00 ..........................................................................
  15.00 ..........................................................................
  16.00 ..........................................................................
  17.00 ..........................................................................
  18.00 ..........................................................................
  19.00 ..........................................................................
  20.00 ..........................................................................
  21.00 ..........................................................................
  22.00 ..........................................................................
                                EDT V16.5A00
@read 'drogerie' ..............................................0000.00:001(0)
```

Die SAM-Datei DROGERIE wird in die Arbeitsdatei 0 eingelesen. Alle Änderungen erfolgen im virtuellen Speicher des EDT. Sie sind anschließend mit einem @WRITE oder @SAVE in eine Datei zu sichern.

```
   1.00 LFD.NR   ART.NR. ART.NAME       BESTAND   BESTELLT.....................
   2.00 1        0024    SEIFE          3000      150.........................
   3.00 2        0015    DEODORANT      2500      600.........................
   4.00 3        0048    PARFUEM        400       60..........................
   5.00 4        0003    CREME          987       555.........................
   6.00 5        0091    RASIERSCHAUM   350       30..........................
   7.00 6        0090    RASIERWASSER   340       30..........................
   8.00 7        0092    RASIERPINSEL   200       30..........................
   9.00 8        0054    ZAHNCREME      400       50..........................
  10.00 9        0055    ZAHNBUERSTE    200       30..........................
  11.00 10       0061    DUSCHGEL       250       40..........................
  12.00 11       0062    BADEZUSATZ     150       55..........................
  13.00 12       0071    KOERPERLOTION  100       80..........................
  14.00 13       0073    HANDCREME      350       30..........................
  15.00 14       0075    NACHTCREME     240       20..........................
  16.00 15       0076    TAGESCREME     300       -...........................
  17.00 16       0105    SONNENMILCH    160       200.........................
  18.00 17       0107    SONNENOEL      220       200.........................
  19.00 18       0121    SONNENBRILLEN  50        50..........................
  20.00 19       0144    BADETASCHEN    30        35..........................
  21.00 20       0056    KAEMME         40        200.........................
  22.00 21       0057    HAARBUERSTEN   70        150.........................
  23.00 22       0058    MASSAGEBUERSTE 35        40..........................
+ ..............................................................0001.00:001(0)
```

Das Datenfenster soll in Richtung Dateiende um ein Datenfenster geblättert und auf überschreibbar gestellt werden.

In der Anweisungszeile wird + eingegeben und mit der Funktionstaste F2 abgeschickt.

```
  24.00 23       0039    HAARSHAMPOO    600       300.........................
  25.00 24       0010    TASCHENTUECHER 1500      500.........................
  26.00 25       0053    MANIKUERE-SET  80        50..........................
  27.00 26       0201    WINDELN        2000      500.........................
  28.00 27       0210    BABY-CREME     1300      100.........................
  29.00 28       0211    BABY-OEL       700       400.........................
  30.00 29       0220    BABY-NAHRUNG   4000      200.........................
  31.00 ..........................................................................




#21 ............................................................0021.00:001(0)
```

Mit F2 wurde das Datenfenster auf überschreibbar gestellt. Dadurch kann in der letzten Spalte BESTELLT der Zeile 26.00 direkt die Zahl 50 eingetragen werden.

Das Datenfenster soll auf Zeile 21 positioniert werden. Dazu wird in der Anweisungszeile #21 eingegeben und mit DUE abgeschickt.

```
   21.00 20       0056    KAEMME          40        200.........................
d  22.00 21       0057    HAARBUERSTEN    70        150.........................
d  23.00 22       0058    MASSAGEBUERSTE  35        40..........................
d  24.00 23       0039    HAARSHAMPOO     600       300.........................
d  25.00 24       0010    TASCHENTUECHER  1500      500.........................
   26.00 25       0053    MANIKUERE-SET   80        50..........................
   27.00 26       0201    WINDELN         2000      500.........................
   28.00 27       0210    BABY-CREME      1300      100.........................
   29.00 28       0211    BABY-OEL        700       400.........................
   30.00 29       0220    BABY-NAHRUNG    4000      200.........................
   31.00 ..........................................................................
```

Die Zeilen 22.00 bis 25.00 sollen gelöscht werden. Dazu werden sie in der Markierungspalte mit D markiert und das Datenfenster mit DUE abgeschickt.

```
   21.00 20       0056    KAEMME          40        200.........................
4  26.00 25       0053    MANIKUERE-SET   80        50..........................
   27.00 26       0201    WINDELN         2000      500.........................
   28.00 27       0210    BABY-CREME      1300      100.........................
   29.00 28       0211    BABY-OEL        700       400.........................
   30.00 29       0220    BABY-NAHRUNG    4000      200.........................
   31.00 ..........................................................................
```

Die Zeilen 22.00 bis 25.00 wurden gelöscht.

Vor Zeile 26 sollen 4 neue Zeilen eingefügt werden. Dazu ist in der Markierungsspalte die Zeile 26.00 mit 4 zu markieren und mit DUE abzuschicken.

```
   21.00 20       0056    KAEMME          40        200.........................
   22.00 21       0133    schnebrille     100       20..........................
   23.00 22       0134    skiwachs        500       - ..........................
   24.00 23       0138    handschuhe      150       - ..........................
   25.00 24       0139    schal           15        30..........................
   26.00 25       0053    MANIKUERE-SET   80        50..........................
   27.00 26       0201    WINDELN         2000      500.........................
   28.00 27       0210    BABY-CREME      1300      100.........................
   29.00 28       0211    BABY-OEL        700       400.........................
   30.00 29       0220    BABY-NAHRUNG    4000      200.........................
   31.00 ..........................................................................
   32.00 ..........................................................................
   33.00 ..........................................................................
   34.00 ..........................................................................
   35.00 ..........................................................................
   36.00 ..........................................................................
   37.00 ..........................................................................
   38.00 ..........................................................................
   39.00 ..........................................................................
   40.00 ..........................................................................
   41.00 ..........................................................................
   42.00 ..........................................................................
   43.00 ..........................................................................
+  ..............................................................0021.00:001(0)
```

In die vier bereitgestellten Zeilen 22.00 bis 25.00 werden neue Artikel eingegeben.

Anschließend soll um ein Datenfenster Richtung Dateiende geblättert werden. Dazu ist in der Anweisungszeile + einzugeben und mit DÜE abzuschicken.

```
   30.00 29       0220    BABY-NAHRUNG   4000      200.........................
   31.00 30       0130    nasenspray     250       40..........................
   32.00 ..........................................................................
@write 'drogerie' ..............................................0030.00:001(0)
```

Da mit Zeile 30.00 bereits das Ende der Arbeitsdatei erreicht wurde, positioniert der EDT auf diese Zeile. Am Dateiende können nun ohne weitere Anweisung neue Artikel eingegeben werden: Zeile 31 wird neu angelegt.

Anschließend sollen die Änderungen in die Datei DROGERIE übernommen werden. Dazu ist in der Anweisungszeile @WRITE 'DROGERIE' einzugeben und mit DÜE abzuschikken.

```
   30.00 29       0220    BABY-NAHRUNG   4000      200.........................
   31.00 30       0130    NASENSPRAY     250       40..........................
   32.00 ..........................................................................
%  EDT0903 FILE 'DROGERIE' IS IN THE CATALOG, FCBTYP = SAM
y  EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO).................0030.00:001(0)
```

Da die Datei DROGERIE bereits existiert, fragt der EDT nach, ob die Datei überschrieben werden soll.

Die Datei soll überschrieben werden. Dazu ist in der Anweisungszeile Y einzugeben und mit DÜE abzuschicken.

```
   30.00 29       0220    BABY-NAHRUNG   4000      200.........................
   31.00 30       0130    NASENSPRAY     250       40..........................
   32.00 ..........................................................................
```

```
%  EDT0171 FILE ':3:$USID.DROGERIE' REPLACED AND WRITTEN
@halt ..................................................0030.00:001(0)
```

Der EDT bestätigt, daß die Arbeitsdatei in die Datei DROGERIE geschrieben wurde.

Der EDT soll jetzt beendet werden. Dazu ist in der Anweisungszeile @HALT einzugeben und mit DUE abzuschicken.

Existieren beim Beenden noch ungesicherte Arbeitsdateien, wird der EDT nicht beendet. Nach der Meldung: % EDT0900 EDITED FILE(S) NOT SAVED! werden die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben. Danach folgt die Anfrage an den Benutzer: % EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)

Bei Antwort N:	Das EDT-Arbeitsfenster erscheint wieder. Der Benutzer kann ungesicherte Dateien schließen und zurückschreiben.
Bei Antwort Y:	Die ungesicherten, virtuellen Dateien gehen verloren. Der EDT wird beendet.

3 Anwendung des EDT

3.1 Starten des EDT

Aufruf des EDT als Hauptprogramm bis BS2000/OSD-BC V1.0

Der Aufruf des EDT als Hauptprogramm erfolgt mit dem Kommando:

START-PROGRAM $EDT	AMODE 31 (wenn von der Hardware unterstützt)

oder

START-PROGRAM *MODULE($EDTLIB,EDTC)	AMODE 24

Starten des EDT ab BS2000/OSD-BC V2.0

Ab BS2000/OSD-BC V2.0 kann der EDT mit dem Kommando START-EDT geladen und gestartet werden. Das START-EDT-Kommando ermöglicht die Auswahl einer bestimmten EDT-Version bei Koexistenz mehrerer Versionen. Der Alias-Name für das START-EDT-Kommando ist EDT

START-EDT	Kurzname: **EDT**
VERSION = *STD / <product-version 6..10> /<product-version 4..8 without-correction-state> / <product-version 3..7 without-manual-release> **,MONJV = *NONE** / <full-filename 1..54 without-gen-vers> **,CPU-LIMIT = *JOB-REST** / <integer 1..32767> **,PROGRAM-MODE = *ANY** / 24	

VERSION =
Produktversion des EDT, die gestartet werden soll.

VERSION = *STD
Die durch das Kommando SET-PRODUCT-VERSION definierte Version wird ausgewählt. Wenn es keine definierte Version gibt, wird vom System die höchstmögliche Version ausgewählt.

VERSION = <product-version 6..10> /
<product-version 4..8 without-correction-state> /
<product-version 3..7 without-manual-release>
Explizite Angabe der Produktversion.

MONJV = *NONE / <full-filename 1..54 without-gen-vers>
Name der Jobvariablen, die den EDT-Lauf überwachen soll. Die Jobvariable muß bereits katalogisiert sein (nur für Benutzer mit dem Software-Produkt Jobvariablen).

Während des EDT-Laufs setzt das System die Jobvariablen auf folgende Werte:

Wert	Bedeutung der Wertzuweisung
$R	EDT läuft
$T	EDT wurde normal beendet
$A	EDT wurde nicht normal beendet

MONJV = *NONE
Es wird keine Jobvariable zur Überwachung verwendet.

CPU-LIMIT = *JOB-REST / <integer 1..32767>
CPU-Zeit, die der EDT beim Ablauf verbrauchen darf. Wird diese Zeit überschritten, wird im Dialogbetrieb der Benutzer vom System benachrichtigt; im Stapelbetrieb wird der Lauf beendet.

CPU-LIMIT = *JOB-REST
Ist im SET-LOGON-PARAMETERS-Kommando der Operand CPU-LIMIT=STD angegeben worden, gibt es keine Zeitbeschränkung für das Programm.
Ist im SET-LOGON-PARAMETERS-Kommando der Operand CPU-LIMIT=t angegeben worden, wird als Zeitbeschränkung für den EDT-Lauf der bei der Systemgenerierung festgelegte Wert verwendet.

PROGRAM-MODE =
Bestimmt in welchem Adressierungsmodus der EDT ablaufen soll.

PROGRAM-MODE = *ANY
Der EDT wird im oberen Adreßraum geladen und läuft im 31-Bit-Modus ab.

PROGRAM-MODE = 24
Der EDT wird im unteren Adreßraum geladen und läuft im 24-Bit-Modus ab. Ist der EDT als Subsytem im oberen Adreßraum geladen, so wird eine private Kopie im unteren Adreßraum nachgeladen.

Der EDT wird im F-Modus gestartet.

Bei gesetztem Auftragsschalter 5 (siehe Abschnitt „Auftragsschalter“ auf Seite 60ff.) wird der L-Modus eingestellt. Der EDT liest die Eingaben mit RDATA von SYSDTA.

Bei Aufruf des EDT können Voreinstellungen auf folgende Weise gesetzt werden:

– Zeichenfolgevariable durch S-Variable initialisieren
– Ausführen einer EDT-Startprozedur.

Die Abarbeitung erfolgt in der angegebenen Reihenfolge.

Zeichenfolgevariable durch S-Variable initialisieren

Beim Aufruf des EDT sind die Zeichenfolgevariablen mit je einem Leerzeichen initialisiert. Ist das Subsystem SDF-P im System verfügbar, so können S-Variable zur Initialisierung der Zeichenfolgevariablen und bei Beenden des EDT zur Weitergabe von Werten verwendet werden. Dabei gilt folgendes:

– Wenn S-Variable SYSEDT-S00 bis SYSEDT-S20 mit TYPE=STRING existieren und einen Wert enthalten, werden deren Inhalte in die entsprechende Zeichenfolgevariable #S00 bis #S20 übernommen, #S00 bis #S20 sind damit initialisiert.
– Ist der Inhalt der S-Variablen länger als 256 Zeichen, so werden keine Zeichen übernommen, die entsprechende Zeichenfolgevariable wird nicht initialisiert. Es erfolgt keine Fehlermeldung.
– Bei Beenden des EDT durch @HALT werden die Werte der Zeichenfolgevariablen #S00 bis #S20 in existierende S-Variable SYSEDT-S00 bis SYSEDT-S20 exportiert. Neue S-Variable SYSEDT-Sxx werden vom EDT selbst nicht deklariert, können aber im EDT mit der Anweisung @SETVAR deklariert werden.

Abarbeiten einer Start-INPUT-Prozedur

Existiert eine Datei mit dem Namen EDTSTART auf der Benutzerkennung, wird diese als @INPUT-Prozedur beim Aufruf des EDT abgearbeitet. Gibt es keine Datei mit dem Namen EDTSTART, wird die Datei $EDTSTART abgearbeitet. Näheres zu @INPUT-Prozeduren siehe Kapitel „EDT-Prozeduren“ auf Seite 127ff. und Anweisung @INPUT. Eine @HALT-Anweisung in der Prozedur bricht die Abarbeitung der @INPUT-Prozedur ab. Es werden keine Fehlermeldungen ausgegeben.

Beispiel für eine Start-Prozedur EDTSTART

```
@IF ON=5 RETURN   ------------------------------------------------------ (1)
@PAR GLOBAL INFORMATION=ON, EDIT FULL=ON   ---------------------------- (2)
@GETJV '$SYSJV.JOBNAME'=1   ------------------------------------------- (3)
@ON 1:1-5 F 'USER1'   ------------------------------------------------------(4)
@DELETE   ------------------------------------------------------------------(5)
@IF .FALSE. RETURN   -------------------------------------------------------(6)
@SYMBOLS FILLER=' '   ------------------------------------------------------ (7)
```

(1) Wenn Auftragsschalter 5 gesetzt, Prozedur abbrechen.

(2) Einstellungen für alle Arbeitsdateien.

(3) Der Inhalt von $SYSJV.JOBNAME wird in Zeile 1 der Arbeitsdatei geschrieben.

(4) Prüfen, ob Zeile 1 (Inhalt von $SYSJV.JOBNAME) gleich 'USER1' ist. Es wird nicht zwischen 'USER1' und z.B. 'USER11' unterschieden.

(5) Die Arbeitsdatei wird gelöscht (Zeile 1 löschen).

(6) Wenn bei @ON kein Treffer festgestellt wurde (Zeile 1 nicht gleich ’USER1’ ist), Prozedur abbrechen.

(7) Definieren des Füllzeichens zwischen Satzende und Bildschirmzeilenende als Leerzeichen.

Aufruf des EDT als Unterprogramm

Der EDT kann nicht nur als Hauptprogramm, sondern auch von einem Benutzerprogramm aus als Unterprogramm aufgerufen werden.

Der Aufruf des EDT als Unterprogramm ist im Handbuch, „EDT-Unterprogrammschnittstellen“ [9] beschrieben.

3.2 Unterbrechen und Beenden des EDT-Laufs

Unterbrechen des EDT-Laufs

Sowohl im F-Modus als auch im L-Modus kann der EDT-Lauf mit @SYSTEM oder mit K2 unterbrochen werden. In beiden Fällen bleibt der EDT geladen.

Eine Rückkehr in den unterbrochenen Arbeitsmodus des EDT ist mit dem Kommando RESUME-PROGRAM möglich. Das Kommando RESUME-PROGRAM bewirkt, daß der EDT-Lauf an der Stelle fortgesetzt wird, an der er unterbrochen wurde. Wird im F-Modus das Arbeitsfenster, in dem der EDT-Lauf unterbrochen wurde, nach RESUME-PROGRAM nicht oder nur unvollständig ausgegeben, kann der ursprüngliche Inhalt mit K3 wiederhergestellt werden.

Hat man im F-Modus den EDT-Lauf unterbrochen, kann man mit dem Kommando SEND-MESSAGE TO=PROGRAM in den F-Modus zurückkehren. Der Rest der Anweisungszeile wird jedoch nicht mehr ausgeführt.

Hat man im L-Modus den EDT-Lauf unterbrochen, kann man mit dem Kommando SEND-MESSAGE TO=PROGRAM in den L-Modus zurückkehren. Dabei wird die STXIT-Routine des EDT ausgeführt. Die STXIT-Routine schließt u.a. alle eröffneten Dateien (ausgenommen eine durch @OPEN eröffnete Datei). Gibt es keine aktive @DO- oder @INPUT-Prozedur, gibt die STXIT-Routine das aktuelle Anweisungssymbol auf dem Bildschirm aus.

Hat der EDT zum Zeitpunkt der Unterbrechung die Zeilen einer @DO- oder @INPUT-Prozedur oder die Zeilen eines Eingabeblocks (BLOCK-Modus) noch nicht vollständig abgearbeitet, wird bei Rückkehr mit SEND-MESSAGE die Verarbeitung abgebrochen, die restlichen Zeilen werden nicht mehr abgearbeitet.

Werden während der Unterbrechung die Kommandos START-PROGRAM oder LOAD-PROGRAM eingegeben bzw. Prozeduren gestartet, die diese Kommandos enthalten, wird der EDT sowohl im F-Modus als auch im L-Modus entladen.

Beenden des EDT-Laufs

@HALT, @RETURN, @EXEC und @LOAD sowie K1 beenden den EDT. Dabei schließt der EDT alle eröffneten Dateien.

Der EDT kann im Dialog mit @END beendet werden. Im L-Modus wird zuvor eine Meldung ausgegeben.

Mit @HALT ABNORMAL kann im Dialog oder in einer Systemprozedur ein nicht normales Beenden des EDT-Laufs erzwungen werden.

Existieren beim Beenden noch ungesicherte Arbeitsdateien, wird der EDT nicht beendet. Nach der Meldung:

% EDT0900 EDITED FILE(S) NOT SAVED!

werden die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben. Danach folgt die Anfrage an den Benutzer:

% EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)

Bei Antwort N: Das EDT-Arbeitsfenster erscheint wieder. Der Benutzer kann ungesicherte Dateien schließen und zurückschreiben.

Bei Antwort Y: Die ungesicherten, virtuellen Dateien gehen verloren. Der EDT wird beendet.

Ist das Subsystem SDF-P im System verfügbar und existieren S-Variable SYSEDT-S00 bis SYSEDT-S20 mit TYPE=STRING, werden beim Beenden des EDT durch @HALT oder @END die Werte der Zeichenfolgevariablen #S00 bis #S20 in die S-Variablen SYSEDT-S00 bis SYSEDT-S20 exportiert. Neue S-Variable SYSEDT-Sxx werden vom EDT selbst nicht deklariert, können aber im EDT mit der Anweisung @SETVAR deklariert werden.

Tritt das Ereignis „Uberschreitung der Programmlaufzeit“ auf (Laufzeit des EDT ist größer als der im Kommando START-PROGRAM angegebene Wert für CPU-LIMIT), so wird eine Meldung auf SYSOUT ausgegeben und im Stapelbetrieb der EDT abnormal beendet.

Falls das Unterbrechungsereignis PROCHK (Programmüberprüfung) oder ERROR (nicht behebbarer Programmfehler) auftritt und der EDT-Datenbereich noch adressierbar ist, wird die Meldung % EDT8910 ausgegeben, in der der Befehlszähler und das Unterbrechungsgewicht angegeben sind. Im Dialog wird der EDT nicht normal beendet oder versucht (z.B. bei Datenfehler im L-Modus), durch Löschen der aktuellen Arbeitsdatei die fehlerhaften Daten zu entfernen. Ist dies nicht möglich, wird mit der Anforderung eines Speicherabzugs TERM gegeben.

Zur Steuerung von Systemprozeduren, in denen der EDT aufgerufen wird, wird sowohl bei normaler Beendigung des EDT durch @HALT, @RETURN oder im Dialog durch @END, wie auch bei abnormaler Beendigung durch das System oder den Benutzer mit @HALT Information über die Beendigungsursache und den EDT-Lauf zur Verfügung gestellt.

Diese Informationen stehen nicht zur Verfügung, wenn der EDT-Lauf mit @EXEC oder @LOAD abgebrochen wurde.

3.2.1 Kommando-Returncode des EDT

Ab BS2000/OSD-BC V1.0 liefert der EDT einen Kommando-Returncode, der von SDF-P zur Steuerung in S-Prozeduren verwendet werden kann. Durch den Kommando-Returncode besteht die Möglichkeit, auf bestimmte Fehlersituationen gezielt zu reagieren.

Der Kommando-Returncode besteht aus drei Teilen:

– dem Maincode, der einem Meldungsschlüssel entspricht, über den mit dem Kommando HELP-MSG-INFORMATION detailierte Informationen abgefragt werden können
– dem Subcode1 (SC1), der die aufgetretene Fehlersituation in eine Fehlerklasse einordnet, aus der abgeleitet werden kann, wie schwerwiegend ein Fehler ist
– dem Subcode2 (SC2), der Zusatzinformationen (Wert ungleich Null) enthalten kann

(SC2)	SC1	Maincode	Bedeutung
0	0	EDT8000	Normale Beendigung des EDT-Laufs. Es trat keine Meldung auf
2	0	EDT8000	Normale Beendigung des EDT-Laufs. Es traten nur Meldungen der Meldungsstufe* 0, 1, 2 auf. Syntax fehlerfrei. Nur Informationen, Warnungen, Meldungen
5	0	EDT8000	Normale Beendigung des EDT-Laufs. Es trat eine Meldung mit der Meldungsstufe* 4 oder 5 auf. Syntax fehlerfrei. Fehler bei Funktion oder Ausführung
10	0	EDT8000	Normale Beendigung des EDT-Laufs. Es trat ein Syntaxfehler in einer Anweisung auf (Meldungsstufe* 3)
50	64	EDT8100	Abnormale Beendigung durch den Benutzer (@HALT ABNORMAL)
100	64	EDT8200	Abbruch wegen Zeitablauf (RUNOUT)
100	64	EDT8292	Fehler in RDATA. Programm abgebrochen
100	64	EDT8901	Abbruch wegen Datenfehler
100	64	EDT8902	Abbruch wegen Datenfehler (ENTER-Prozeß)
150	64	EDT8910	Programmunterbrechung Abnormaler Abbruch mit DUMP
150	64	EDT8001	Abnormale Beendigung des EDT-Laufs Abnormaler Abbruch mit DUMP
200	64	EDT8002	Fehler beim Nachladen des Modus EDT
200	64	EDT8003	Nicht genügend virtueller Speicher verfügbar
200	64	EDT8005	Fehler bei Initialisierung des EDT
200	64	EDT8006	Installationsfehler bei EDT oder EDTLIB
200	64	EDT8300	Interner EDT-Fehler
200	64	EDT8900	Kein virtueller Adressierungsspeicher

* Die Meldungsstufe ist die Tausenderstelle der Meldungsnummer

Im Fehlerfall können die Komponenten des Returncodes mit den SDF-P-Funktionen SUBCODE1(), SUBCODE2() und MAINCODE() abgefragt werden.

Nach einer fehlerfreien Durchführung kann mit dem Kommando SAVE-RETURNCODE der Returncode sichergestellt und ebenfalls ausgewertet werden. (Nähere Information zu Kommando-Returncodes und zum Abfragen von Returncodes siehe Handbuch „SDF-P“ [8]).

Beispiel zum Abfragen von Returncodes

```
/MODIFY-JOB-SWITCHES ON=5
/START-PROGRAM $EDT
@LOG NONE
@...
@DIALOG
@...
@HALT
/SAVE-RETURNCODE
/IF-BLOCK-ERROR
/    WRITE-TEXT 'FEHLER: &SUBCODE1, &SUBCODE2, &MAINCODE'
/ELSE
/    WRITE-TEXT 'EDT NORMAL BEENDET'
/    IF (&SUBCODE2 > 5)
/        WRITE-TEXT 'SYNTAX FEHLER IST AUFGETRETEN'
/        RAISE-ERROR MAINCODE=EDT3002
/    END-IF
/    ...
/END-IF
/HELP-MSG-INFORMATION &MAINCODE
/MODIFY-JOB-SWITCHES OFF=5
```

3.2.2 Überwachung des EDT-Laufs mit Monitor-Jobvariablen

Der Ablauf des EDTs kann mit einer BS2000-Jobvariablen überwacht werden.

Mit den folgenden Kommandos wird die Monitor-Jobvariable vom Betriebsystem eingerichtet:

START-PROGRAM $EDT,MONJV=jvname oder

START-EDT MONJV=jvname (ab BS2000/OSD-BC V2.0)

Das Betriebsystem bildet in der Jobvariablen zwei Werte ab:

– eine Zustandsanzeige von 3 Byte Länge,

– eine Rückkehrcode-Anzeige von 4 Byte Länge.

Folgende Tabelle zeigt, wie die Jobvariable vom EDT versorgt wird.

Fehlerklasse	**Beendigung**	**Zustands-anzeige**	**Rückkehr-code**	**Spin-off-Mechanismus**
[keine Meldung] [Hinweis] [Funktions-Fehler] [Syntax-Fehler]	normal	$T	0000 1002 1005 1010	nein
[Unterbrechung]	nicht normal durch den Benutzer	$A	2050	ja
[Fatal] [Fatal]+DUMP [Initialisierungs-Fehler]	nicht normal		2100 2150 3200	

Die letzten 3 Stellen des Rückkehrcodes stimmen in Wert und Bedeutung mit dem Subcode2 (SC2) des Kommando-Returncodes überein.

3.3 Ein- und Ausgabe

Eingabe

Die Eingabe für den EDT kann kommen:

– primär über den Bildschirm
– von einer SAM- oder ISAM-Datei
– von einem Bibliothekselement
– von einer POSIX-Datei
– von einer anderen Arbeitsdatei des EDT

Der EDT unterscheidet bei den Eingaben zwischen Daten (Texten) und Anweisungen.

Ausgabe

Die Arbeitsdateien kann der EDT ganz oder teilweise ausgeben:

– primär auf den Bildschirm
– in eine SAM- oder ISAM-Datei auf Platte
– in ein Bibliothekselement
– in eine POSIX-Datei
– in eine weitere Arbeitsdatei des EDT
– auf den Schnelldrucker

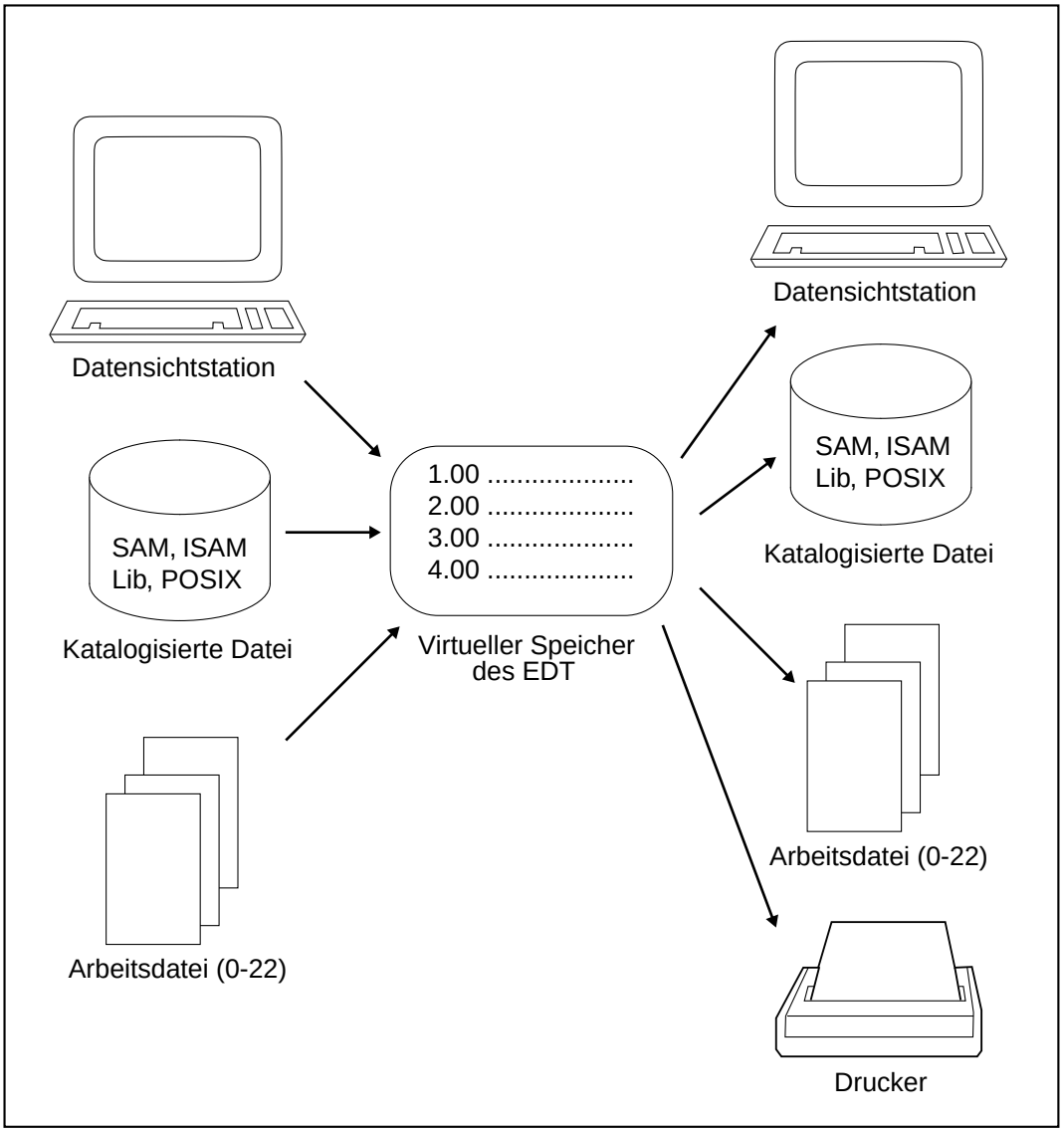


Bild 4: Ein- und Ausgabe des EDT

3.3.1 Eingeben von Daten (Text)

Daten werden beim Erstellen einer Datei in der Reihenfolge, in der sie eingegeben werden, in die Datei übernommen. Dabei wird der Text entsprechend der eventuell enthaltenen Tabulatorzeichen aufbereitet.

Beim Ändern überträgt der EDT die geänderten Zeilen in die entsprechenden Sätze der Datei.

3.3.2 Eingeben von Anweisungen

Anweisungen steuern den EDT-Lauf.

Der EDT interpretiert die Eingabe als Anweisung,

– die im F-Modus aus der Anweisungszeile oder aus der Markierungsspalte (Kurzanweisung) des Arbeitsfensters kommt,

– wenn im L-Modus das erste von einem Leerzeichen verschiedene Zeichen das Anweisungssymbol ist und das erste von einem Leerzeichen verschiedene Zeichen, das dem Anweisungssymbol folgt, kein Anweisungssymbol ist.

  Das Standard-Anweisungssymbol des L-Modus ist @. Beginnt eine Eingabe mit @@ (zwei Anweisungssymbole), interpretiert der EDT diese Eingabe als Text. Das zweite Anweisungssymbol betrachtet er als erstes Zeichen dieses Textes. Alle Zeichen (erstes Anweisungssymbol, Leerzeichen), die vor dem zweiten @ stehen, schneidet der EDT ab.

  Erhält der EDT im L-Modus die Eingabe vom Bildschirm, ist es bei eingeschaltetem BLOCK-Modus (siehe Kapitel „Anweisungen des EDT“ auf Seite 147ff.) möglich, dem EDT mit einem einzigen Eingabeblock mehrere Eingaben zu übergeben.

  Eine einzelne Eingabe darf bis zu 256 Zeichen lang sein.

Bei der Angabe von " bzw. ’ innerhalb einer Zeichenfolge muß "" bzw. ” eingegeben werden.

Beispiel

’Dies ist eine ”beliebige” Zeichenfolge’

Allgemeines Format einer Anweisung:

Operation	Operanden	F-Modus / L-Modus / @PROC
Operation	{ Operanden &str-var }	

Operation — Die Operation entspricht dem Anweisungsnamen, z.B. @OPEN, @COPY, @WRITE...
Sie muß am Anfang der Anweisung stehen. Im F-Modus kann das EDT-Anweisungssymbol (standardmäßig @) auch weggelassen werden.

Operanden — Der Operation folgen, durch ein oder mehrere Leerzeichen getrennt, die Operanden. Die Operanden sind in der vorgegebenen Reihenfolge anzugeben. Vor bzw. nach jedem Operanden können beliebig viele Leerzeichen eingegeben werden.

str-var — Zeichenfolgevariable, die die Operanden enthält (indirekte Angabe der Operanden).

Das Trennzeichen (Leerzeichen) zwischen Operation und Operanden, bzw. zwischen den einzelnen Operanden muß dann angegeben werden, wenn Operation und Operand bzw. zwei aufeinanderfolgende Operanden nicht unterscheidbar sind (Beispiel: @SYMBOLS='?' ist falsch; richtig ist @SYMBOL S='?').

3.3.3 Indirekte Angabe von Operanden

Nach Erkennen des Anweisungsnamens (Operation) wird der Anweisungsrest durch den Inhalt der angegebenen Zeichenfolgevariablen ersetzt und die darin enthaltenen Operanden ausgewertet.

Ist die Protokollierung eingeschaltet (z.B. @LOG ALL oder @LOG COMMANDS), wird zusätzlich zur Originaleingabe auch die durch Ersetzung erzeugte Anweisung ausgegeben. Im Fehlerfall wird bei der Protokollierung nur die Ersetzung ausgegeben.

Wenn die Länge von Anweisungsname und der Ersetzung der Zeichenfolgevariablen größer als 256 ist, wird die Abarbeitung der Anweisung mit der Fehlermeldung abgewiesen:
% EDT1905 INPUT TOO LONG. CORRECT INPUT

Einschränkungen

Bei folgenden Anweisungen ist keine Ersetzung möglich:

– Anweisungen, die keinen Anweisungsnamen haben (z.B. Umdefinieren des Anweisungssymbols und Wertzuweisung zu EDT-Variablen ohne @SET).

– Anweisungen, deren Anweisungsname nicht eindeutig erkennbar ist (z.B. Setzen und Ändern der aktuellen Zeilennummer mit @ln oder @+, @–; innerhalb des Operanden text z.B. bei @IF).

– Anweisung @PARAMS.

Innerhalb von @DO-Prozeduren erfolgt zuerst die Ersetzung der Prozedurparameter, erst dann wird eine eventuelle indirekte Operandenangabe aufgelöst.

3.3.4 Symbolische Zeilennummern

Bei einigen Anweisungen (z.B @ON) kann der Zeilenbereich auch durch symbolische Zeilennummern (%, *, $) oder durch das aktuelle Bereichssymbol (&) angegeben werden.

Zeichen	Bedeutung
%	Zeilennummer des ersten Satzes in der Arbeitsdatei
$	Zeilennummer des letzten Satzes in der Arbeitsdatei
*	aktuelle Zeilennummer
&	eingestellter Zeilenbereich (siehe @RANGE, standardmäßig 0000.0001-9999.9999)
?	Zeilennummer der ersten Trefferzeile nach @ON

Bei der Angabe von Zeilenbereichen müssen symbolische Zeilennummern durch einen Punkt (.) entwertet werden.

Beispiel

%.–10	Zeilenbereich von erster Zeile bis Zeile 10
*–1L	Zeile vor der aktuellen Zeile
10–.$	Zeilenbereich von Zeile 10 bis letzte Zeile
$–1L	vorletzte Zeile

3.4 Arbeitsdateikonzept

Im L-Modus stehen dem Benutzer zur Dateibearbeitung 23 virtuelle Dateien zur Verfügung. Dies sind die sogenannten Arbeitsdateien 0 bis 22. Die Arbeitsdateien sind grundsätzlich virtuelle Dateien. In der Arbeitsdatei 0 kann man auch eine real geöffnete ISAM-Datei bearbeiten (@OPEN).

Im F-Modus stehen dem Benutzer die Arbeitsdateien 0 bis 9 zur direkten Eingabe von Daten zur Verfügung. Im F-Modus kann man das Arbeitsfenster teilen und zwei Arbeitsdateien gleichzeitig darstellen (siehe @PAR SPLIT).

Die Arbeitsdateien 9 und 10 werden von manchen Anweisungen benötigt. Sie sollten deshalb nur als temporäre Hilfsdateien verwendet werden.

– In der Arbeitsdatei 9 werden von einigen Anweisungen Ergebnisse abgelegt (z.B. @COMPARE, Format 2, @FSTAT, @SHOW, @STATUS). Dabei wird evtl. vorhandener Inhalt ohne Warnung gelöscht.

– Die Arbeitsdatei 10 wird von @COMPARE, Format 2 als Hilfsdatei verwendet, falls keine andere Hilfsdatei vom Benutzer explizit angegeben wird.

3.5 Dateibearbeitung

Folgende ISAM- bzw. SAM-Dateien werden vom EDT standardmäßig bearbeitet:

ISAM-Dateien mit

– variabler Satzlänge (RECORD-FORMAT = VARIABLE(...))
– Schlüsselposition 5 (ACCESS-METHOD = ISAM(KEY-POSITION = 5))
– Schlüssellänge 8 (ACCESS-METHOD = ISAM(KEY-LENGTH = 8))
– numerischen Schlüsseln (X'F0'-X'F9', mindestens 1 Zeichen ≠ X'F0'),
– Blockgröße gleich 1 (BUFFER-LENGTH = STD(1)).

Der EDT kann den ISAM-Schlüssel als Zeilennummer interpretieren.

SAM-Dateien mit

– variabler Satzlänge (RECORD-FORMAT = VARIABLE(...))
– Blockgröße gleich 1 (BUFFER-LENGTH = STD(1)).

In Systemen mit der Voreinstellung BLKCTRL=DATA ist der Standardwert für Blockgröße auch NK4-Platten BUFFER-LENGTH=STD(2).

Alle Sätze in einer SAM- oder ISAM-Datei mit variabler Satzlänge, die länger als 256 Zeichen sind, werden beim Zurückschreiben ab Position 257 abgeschnitten.

Die Dateien können mit @GET, @READ, @SAVE, @WRITE, @ELIM und @INPUT (siehe Kapitel „Anweisungen des EDT“ auf Seite 147ff.) bearbeitet werden. Die Dateien können auch mit @OPEN, @WRITE,@COPY Format 2 bearbeitet werden. Die Zugriffsmethode wird durch den Operanden TYPE=SAM|ISAM bestimmt.

Der Name der zu bearbeitenden Datei wird zwischen Hochkomma angegeben ('dateiname'). Anstelle von 'dateiname' kann auch '/' angegeben werden, wenn der Datei vor dem Aufruf des EDT einer der folgenden Dateikettungsnamen fest zugeordnet wurde:

```
/SET-FILE-LINK LINK-NAME = EDTSAM | EDTISAM, FILE-NAME = dateiname
```

Wenn man einer Datei einen Dateikettungsnamen fest zugeordnet hat, kann der EDT keine andere SAM-Datei (EDTSAM) bzw. ISAM-Datei (EDTISAM) mehr bearbeiten, bis die Zuordnung aufgehoben wird.

Die Aufhebung einer festen Zuordnung erfolgt durch:

```
/REMOVE-FILE-LINK LINK-NAME = EDTSAM | EDTISAM
```

Die Angabe von '/' als Dateiname bewirkt, daß der EDT vor dem Öffnen der Datei keine Prüfung der Kataloginformation vornimmt, und daß nach dem Schließen der Datei der überschüssige Speicherplatz nicht freigegeben wird. Es erfolgt keine Sicherungsabfrage, ob eine bestehende Datei überschrieben werden soll. Überschüssiger Speicherplatz wird auch nicht freigegeben, wenn der Dateiname mit einer USERID ungleich TSOS angegeben wird oder der Auftragsschalter 7 gesetzt ist.

3.5.1 Bearbeiten von ISAM-Dateien mit vom Standard abweichenden Attributen

Sollen Dateien bearbeitet werden, die von den Standard-Attributen abweichen, müssen die Datei-Attribute im SET-FILE-LINK oder im CREATE-FILE-Kommando angegeben werden.

Allgemeine Bearbeitungsreihenfolge:

– Der Dateikettungsname EDTISAM bzw. EDTSAM wird der Datei zugeordnet. Die Zuordnung zum Dateikettungsnamen und die Angabe der vom Standard abweichenden Dateiattribute ist bei den einzelnen Punkten beschrieben. Bei der Erstellung einer neuen Datei müssen alle Dateiattribute angegeben werden.

– Bearbeitung mit @GET bzw. @SAVE bei LINK-NAME = EDTISAM und Bearbeitung mit @READ bzw. @WRITE bei LINK-NAME = EDTSAM.

– Es empfiehlt sich, nach der Bearbeitung die Zuordnung des Dateikettungsnamens mit REMOVE-FILE-LINK LINK-NAME = EDTISAM bzw. LINK-NAME = EDTSAM wieder aufzuheben.

ISAM-Dateien mit vom Standard abweichenden Attributen können auch direkt mit @OPEN Format 2, und dem Operanden TYPE=CATALOG bearbeitet werden. Dabei werden die Attribute direkt aus dem Katalog übernommen. Eine Zuweisung eines Dateikettungsnamens ist nicht notwendig.

ISAM-Dateien mit Schlüssellänge < 8 Byte

```
Zuordnung:   / SET-FILE-LINK LINK-NAME = EDTISAM, FILE-NAME = dateiname, -
             /   ACCESS-METHOD = ISAM(KEY-LENGTH = schlüssellänge)
```

Ist eine kürzere Schlüssellänge als 8 vereinbart, wird die vorhandene EDT-Zeilennummer von links her verkürzt. Beispielsweise wird bei einer KEY-LENGTH-Angabe von 4 die Zeilennummer 1234.5678 als ISAM-Schlüssel 5678 interpretiert. Die Eindeutigkeit eines ISAM-Schlüssels ist damit nicht mehr gewährleistet. Der Benutzer ist selbst für die Eindeutigkeit verantwortlich.

ISAM-Datei mit fester Satzlänge

Zuordnung: /SET-FILE-LINK LINK-NAME = EDTISAM, FILE-NAME = dateiname, -
/ RECORD-FORMAT = FIXED(RECORD-SIZE = satzlänge)

Die Länge des Schlüssels (KEY-LENGTH) kann zwischen 1 und 8 liegen. Wird für KEY-POSITION ein Wert ungleich 1 angegeben, wird @GET bzw. @SAVE zurückgewiesen.

Liegt die Satzlänge über 256 Zeichen, wird nach dem Einlesen eine Warnung ausgegeben. Ein Versuch dieselbe Datei zurückzuschreiben, wird mit einer Fehlermeldung abgewiesen, da sonst der Inhalt der Datei ab Spalte 257 verloren gehen würde.

ISAM-Dateien mit Blockgröße > 1

Zuordnung: /CREATE-FILE FILE-NAME = dateiname, SUPPORT = PUBLIC-DISK(-
/ SPACE = RELATIVE(PRIMARY-ALLOCATION = seitenanzahl))
/SET-FILE-LINK LINK-NAME = EDTISAM, FILE-NAME = dateiname, -
/ BUFFER-LENGTH = STD(SIZE = blockgröße)

Für ISAM-Dateien kann die Blockgröße auf ein Mehrfaches der Standardblockgröße gesetzt werden. In diesem Fall muß mit CREATE-FILE die Anzahl der Seiten über eine Primärzuweisung mindestens doppelt so groß festgelegt werden wie die Blockgröße.

ISAM-Dateien, die mit dem Dateikettungsnamen EDTSAM bearbeitbar sind

– ISAM-Dateien mit variablem Satzformat Schlüsselposition ≠ 5
– ISAM-Dateien mit fester Satzlänge und Schlüsselposition > 1
– ISAM-Dateien mit Schlüssellänge > 8
– ISAM-Dateien mit nichtnumerischem ISAM-Schlüssel

Zuordnung: /SET-FILE-LINK LINK-NAME = EDTSAM, FILE-NAME = dateiname, -
/ ACCESS-METHOD = ISAM(KEY-LENGTH = schlüssellänge, -
/ KEY-POSITION = schlüsselposition), -
/ RECORD-FORMAT = format(RECORD-SIZE = satzlänge)

mit *format* FIXED für feste Satzlänge und *format* VARIABLE für variables Satzformat.

Bearbeitung:

@READ '/' oder @WRITE '/'

Die Zeilennummernvergabe erfolgt dann jedoch abhängig von der aktuellen Zeilennummer und der aktuellen Schrittweite. Der ISAM-Schlüssel wird als Bestandteil des Datensatzes in die Arbeitsdatei übernommen. Wird der ISAM-Schlüssel verändert, so ist zu beachten, daß die Satzreihenfolge der Reihenfolge der ISAM-Schlüssel entsprechen muß, da sonst @WRITE '/' mit einer Fehlermeldung abgewiesen wird.

3.5.2 Bearbeiten von SAM-Dateien mit vom Standard abweichenden Attributen

Allgemeine Bearbeitungsreihenfolge:

– Der Dateikettungsname EDTSAM wird der Datei zugeordnet. Die Zuordnung von EDTSAM und die Angabe der vom Standard abweichenden Dateiattribute ist bei den einzelnen Punkten beschrieben. Bei der Erstellung einer neuen Datei müssen alle Dateiattribute angegeben werden.

– Bearbeitung mit @READ bzw. @WRITE

– Es empfiehlt sich, nach der Bearbeitung die Zuordnung des Dateikettungsnamens mit REMOVE-FILE-LINK LINK-NAME = EDTSAM wieder aufzuheben.

i SAM-Dateien mit vom Standard abweichenden Attributen können auch direkt mit @OPEN, Format 2 und dem Operanden TYPE=CATALOG bearbeitet werden. Dabei werden die Attribute direkt aus dem Katalog übernommen. Eine Zuweisung eines Dateikettungsnamens ist nicht notwendig.

SAM-Datei mit fester Satzlänge

```
Zuordnung:  /SET-FILE-LINK LINK-NAME = EDTSAM, FILE-NAME = dateiname, -
            /   RECORD-FORMAT = FIXED(RECORD-SIZE = satzlänge)
```

Liegt die Satzlänge über 256 Zeichen, wird nach dem Einlesen eine Warnung ausgegeben. Ein Versuch dieselbe Datei zurückzuschreiben, wird mit einer Fehlermeldung abgewiesen, da sonst der Inhalt der Datei ab Spalte 257 verloren gehen würde.

SAM-Dateien mit Blockgröße > 1

```
Zuordnung:  /CREATE-FILE FILE-NAME = dateiname, SUPPORT = PUBLIC-DISK( -
            /   SPACE = RELATIVE(PRIMARY-ALLOCATION = seitenanzahl))
            /SET-FILE-LINK LINK-NAME = EDTSAM, FILE-NAME = dateiname, -
            /   BUFFER-LENGTH = STD(SIZE = blockgröße)
```

Für SAM-Dateien kann die Blockgröße auf ein Mehrfaches der Standardblockgröße gesetzt werden. In diesem Fall muß mit CREATE-FILE die Anzahl der Seiten über eine Primärzuweisung mindestens doppelt so groß festgelegt werden wie die Blockgröße.

SAM-Datei auf Magnetband

```
Zuordnung:  /CREATE-FILE FILE-NAME = dateiname, SUPPORT = TAPE( -
            /   VOLUME = datenträgerkennzeichen, DEVICE-TYP = gerätetyp)
            /SET-FILE-LINK LINK-NAME = EDTSAM, FILE-NAME = dateiname
```

3.6 Dateibearbeitung von POSIX-Dateien

Die Funktionen zur Dateibearbeitung von POSIX-Dateien werden ab BS2000/OSD-BC V2.0 unterstützt. POSIX und das zugehörige Laufzeitsystem CRTE müssen als Subsysteme aktiviert sein.

3.6.1 POSIX im BS2000

Die zunehmende Vernetzung unterschiedlicher Rechnersysteme und die verteilte Verarbeitung in diesen Netzen erfordert die Standardisierung und Offenheit der Rechnersysteme und deren Schnittstellen. Diese Schnittstellen müssen den POSIX-/XPG4-Standards entsprechen. Das Betriebssystem BS2000/OSD-BC V2.0 unterstützt die POSIX-/XPG4-Standards mit dem Softwareprodukt „POSIX“.

Unter POSIX (**P**ortable **O**pen **S**ystem **I**nterface for UNI**X**) bzw. XPG4 **(X**/Open **P**ortability **G**uide Version **4**) versteht man eine Reihe von Standards auf UNIX-Basis. POSIX bezeichnet sowohl diese Standards als auch das Softwareprodukt.

Durch das Softwareprodukt POSIX wird das BS2000 zu einem offenen System. Anwendungen, die dem Standard entsprechen, sind portabel zwischen dem BS2000 und anderen Systemen, die POSIX-Schnittstellen unterstützen, besonders UNIX/SINIX.

Das POSIX-Dateisystem ist ein Dateisystem im BS2000 mit der Struktur eines UNIX-Dateisystems (UFS). Es ist hierarchisch aufgebaut und besteht aus Dateien (POSIX-Dateien) und Dateiverzeichnissen. POSIX-Benutzer können POSIX-Dateien erzeugen und bearbeiten. POSIX-Benutzer können vom POSIX-Dateisystem aus auf ferne UNIX-Dateisysteme zugreifen. Umgekehrt kann von einem fernen UNIX-Rechner auf das lokale POSIX-Dateisystem zugegriffen werden.

Der Zugang zu POSIX ist allen BS2000-Benutzern möglich. Auch von einem UNIX-Rechner aus (über rlogin oder Emulation) kann der Zugang zu POSIX auf einem BS2000-Rechner erfolgen. Die Zugangskontrolle wird vollständig über das BS2000 abgewickelt.

Weitere Informationen zu POSIX im BS2000 finden Sie in den Handbüchern „POSIX Grundlagen für Benutzer und Systemverwalter“ [14] und „POSIX Kommandos“ [15].

3.6.2 EDT und POSIX

Dateien, die im POSIX-Dateisystem abgelegt sind, können mit den Anweisungen @XOPEN und @XCOPY in den EDT eingelesen werden und mit den Anweisungen @XWRITE und @CLOSE ins POSIX-Dateisystem zurückgeschrieben werden.

Konventionen für Dateinamen

Der EDT kann nur Dateinamen und Pfadnamen bis zur maximalen Länge von 256 Zeichen verarbeiten. Ist der Pfadname länger, muß vorher innerhalb der POSIX-Shell mit dem cd-Kommando in ein Unterverzeichnis positioniert werden.

Der Name einer POSIX-Datei ist folgendermaßen definiert:

xpath::= chars | .str-var (siehe auch Operandenbeschreibung).
Zeichenfolge mit maximaler Länge von 256 Zeichen.
Zeichenfolge, die den Namen einer POSIX-Datei angibt (eventuell mit Verzeichnis).

Nichtabdruckbare Zeichen, Leerzeichen und andere Trennzeichen innerhalb eines Dateinamens sind nur bei Angabe in str-var möglich.

Enthält der Name Kleinbuchstaben, so muß bei Eingabe von einer Datensichtstation im L-Modus vorher @LOWER ON oder @PAR LOWER=ON eingeschaltet werden.

Der EDT positioniert nicht innerhalb des POSIX-Dateisystems. Die Dateinamen beziehen sich immer auf das aktuelle Verzeichnis, außer der Name beginnt mit /. In diesem Fall bezieht sich der Name auf das Root-Verzeichnis.

Satzlänge

Der EDT liest die Daten zeichenweise ein. Das Satzende wird durch das Satzende-Kennzeichen X’15’ bzw. X’0A’ erkannt.

Erlaubte Satzlänge: 1 bis 256 Zeichen

Zeichenketten, die länger als 256 Zeichen sind, werden ab Zeichen 257 abgeschnitten. Es wird die Fehlermeldung % EDT1253 (SOME) RECORD(S) TRUNCATED ausgegeben.

Zeichenketten der Länge 0 können im EDT-Datenbereich nicht dargestellt werden und müssen extra behandelt werden.

Abhängig vom AUTOFORM-Modus (siehe @BLOCK) wird beim Einlesen folgendermaßen vorgegangen:

– AUTOFORM ausgeschaltet:
Zeichenketten der Länge 0 werden ignoriert. Es wird kein Satz angelegt.

– AUTOFORM eingeschaltet:
Leerzeilen erhalten als Inhalt das Zeilenende-Kennzeichen X'0D' und werden im Datenbereich angelegt.

Analog dazu wird beim Schreiben einer Datenzeile mit Inhalt X'0D' der AUTOFORM-Modus ausgewertet:

– AUTOFORM ausgeschaltet:
Datenzeilen des Inhalts X'0D' werden als solche in die POSIX-Datei geschrieben.

– AUTOFORM eingeschaltet:
Datenzeilen des Inhalts X'0D' werden als Satz der Länge 0 in die POSIX-Datei geschrieben.

Verarbeiten von Daten im ASCII-Code

Dem EDT muß mit dem Operand CODE mitgeteilt werden, ob die Daten im ASCII-Code vorliegen, bzw. ob sie im ASCII-Code in die POSIX-Datei abgelegt werden sollen.

Es wird eine feste Umwandlungstabelle verwendet. Die Tabelle entspricht der Zuordnung von EDF03IRV zu ISO646 internationaler 7-Bit-Code (gleichwertig zur Zuordnung EDF041 zu ISO8859-1).

Die Daten in der Arbeitsdatei können im ASCII-Code in Hexadezimal-Darstellung mit @PAR HEX=ON und Voreinstellung @PAR CODE=ISO angezeigt und im Datenfenster verändert werden.

Im L-Modus ist eine hexadezimale Eingabe in ASCII-Code durch die Voreinstellung @INPUT HEX ISO möglich.

3.6.3 Bearbeiten von POSIX-Dateien mit dem EDT

POSIX-Dateien können mit folgenden Anweisungen bearbeitet werden:

Funktion	**Anweisung**
Erzeugen einer neuen POSIX-Datei Einlesen einer POSIX-Datei in die Arbeitsdatei	@XOPEN
Kopieren von POSIX-Dateien in eine Arbeitsdatei	@XCOPY
Erzeugen einer neuen POSIX-Datei, indem eine erstellte Arbeitsdatei in eine neue POSIX-Datei geschrieben wird Zurückschreiben einer Arbeitsdatei in eine POSIX-Datei	@XWRITE
Zurückschreiben einer Arbeitsdatei in eine POSIX-Datei und schließen der POSIX-Datei	@CLOSE

Die einzelnen Anweisungen sind im Kapitel „Anweisungen des EDT“ auf Seite 147ff. ausführlich beschrieben.

3.7 Bibliotheksbearbeitung mit dem EDT

Eine Bibliothek ist eine Datei mit Unterstruktur. Sie enthält Elemente und ein Inhaltsverzeichnis.

Ein Element ist eine Ablageeinheit, in der eine logisch zusammengehörige Datenmenge wie z.B. eine Datei, eine Prozedur, ein Bindemodul oder ein Quellprogramm abgelegt wird. Jedes Element ist in der Bibliothek einzeln ansprechbar.

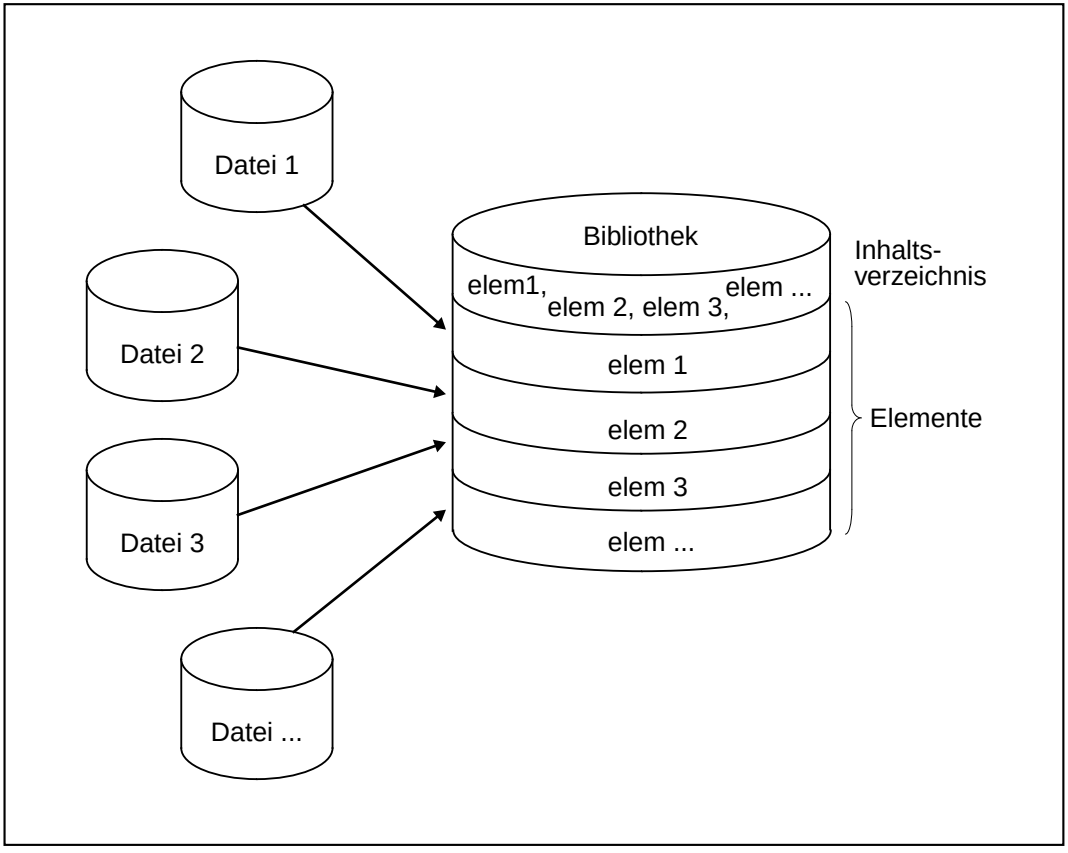


Bild 5: Aufbau einer Bibliothek

Der EDT bearbeitet Programm-Bibliotheken

Programm-Bibliotheken sind PAM-Dateien, die mit der Bibliotheks-Zugriffsmethode PLAM bearbeitet werden. Daher werden sie auch als PLAM-Bibliotheken bezeichnet.

Die Vorteile einer Programmbibliothek bestehen darin, daß

- alle Elementtypen in einer Bibliothek abgelegt werden können. Durch die Eigenschaften der Programmbibliothek können zu einem Projekt sämtliche Daten, vom Quellprogramm, über Binde- und Lademoduln, Übersetzungsprozeduren, Testdaten bis zur Dokumentation, in den entsprechenden Elementen einer Bibliothek gespeichert werden.
- gleichnamige Elemente existieren können, die sich durch Typ- oder Versionsbezeichnung unterscheiden.
- auf die Bibliothek von mehreren Benutzern gleichzeitig auch schreibend zugegriffen werden kann.
- das Speichern mehrerer Dateien als Elemente in einer Bibliothek den Systemkatalog entlastet, da jede Bibliothek nur einen Eintrag hat. Speicherplatz wird gespart, da nur einmal pro Bibliothek eine Standardzuweisung von Speicherplatz vorgenommen wird und die Elemente nur den Platz belegen, den sie wirklich benötigen.

Elementbezeichnung

Elemente sind in Programmbibliotheken über ihre Elementbezeichnung einzeln ansprechbar.

Die Elementbezeichnung setzt sich zusammen aus Name, Version und Elementtyp und wird in folgender Form angegeben:

elemname[(vers)][,elemtyp]

elemname bezeichnet den Namen des Elements der Bibliothek, vers die Versionbezeichnung des Elements. elemtyp bezeichnet den Typ des Elements. Die Angabe der Version und des Elementtyps ist wahlfrei. Wird in einer Anweisung kein Wert für vers eingegeben, wird standardmäßig das Element mit der höchsten Version ausgewählt. Wird in einer Anweisung kein Wert für elemtyp eingegeben, wird standardmäßig der bei @PAR ELEMENT-TYPE angegebene Wert eingesetzt (standardmäßig Typ S).

Aus Kompatibilitätsgründen gelten für die Vergabe der Elementbezeichnung die Namenskonventionen des Softwareprodukts LMS (siehe Handbuch „LMS“ [13]). Sie sind unbedingt einzuhalten, damit Bibliothekselemente, die mit dem EDT erzeugt oder bearbeitet worden sind, auch mit dem Softwareprodukt LMS verwaltet werden können.

3.7.1 Vom EDT unterstützte Elementtypen

Der Elementtyp bestimmt die Art der abgelegten Daten.

Standard- oder vordefinierte Typen sind:

Elemtyp	Inhalt des Elements
S	Quellprogramme.
M	Makros.
J	Prozeduren.
P	Druckaufbereitete Daten.
D	Textdaten.
X	Daten beliebigen Formats.
R	Bindemodule. Wird nur von @DELETE und @SHOW unterstützt.
C	Phasen. Wird nur von @DELETE und @SHOW unterstützt.
H	Compiler-Ergebnisinformationen. Wird nur von @DELETE und @SHOW unterstützt.
L	Bindelagemodule (LLM). Wird nur von @DELETE und @SHOW unterstützt.
U	IFG-Formatmasken.
F	IFG-Benutzerprofile.

Wird in einer Anweisung kein Wert für elemtyp eingegeben, wird standardmäßig der bei @PAR ELEMENT-TYPE angegebene Wert eingesetzt.

Ab EDT V16.5 sind bei den Anweisungen zur Bearbeitung von Bibliothekselementen freie Typnamen (benutzerdefinierte Typen) zugelassen. Es erfolgt keine Prüfung auf den Basistyp.

3.7.2 Bearbeiten von Bibliothekselementen mit dem EDT

Mit dem EDT kann man:

– Elemente erstellen, ändern und lesen (Elementtypen S,M,J,P,D, X und entsprechende freie Typnamen)
– Elemente löschen (alle Elementtypen)
– das Inhaltsverzeichnis einer Bibliothek ausgeben (alle Elementtypen)

Bibliothekselemente können mit folgenden Anweisungen bearbeitet werden:

Funktion	**Anweisung**	**unterstützte Elementtypen**
Erzeugen eines neuen Bibliothekselementes Einlesen eines Bibliothekselementes in die Arbeitsdatei	@OPEN,Format 2	S, M, P, J, D, X, freie Typnamen
Erzeugen eines neuen Bibliothekselementes, indem eine erstellte Arbeitsdatei in ein neues Bibliothekselement geschrieben wird Zurückschreiben einer Arbeitsdatei in ein Bibliothekselement	@WRITE,Format 2	S, M, P, J, D, X, freie Typnamen
Zurückschreiben einer Arbeitsdatei in ein Bibliothekselement und schließen des Bibliothekselements	@CLOSE	S, M, P, J, D, X, freie Typnamen
Kopieren von Bibliothekselementen in eine Arbeitsdatei	@COPY,Format 2	S, M, P, J, D, X, freie Typnamen
Löschen von Bibliothekselementen	@DELETE,Format 2	S, M, P, J, D, X, R, C, H, L, U, F, freie Typnamen
Voreinstellen von Bibliotheken, mit deren Elementen vorwiegend gearbeitet wird	@PAR LIBRARY	S, M, P, J, D, X, freie Typnamen
Voreinstellen des Elementtyps, mit dem vorwiegend gearbeitet wird	@PAR ELEMENT-TYPE	S, M, P, J, D, X, R, C, H, L, U, F, freie Typnamen
Ausgeben des Inhaltsverzeichnisses einer Bibliothek	@SHOW	S, M, P, J, D, X, R, C, H, L, U, F, freie Typnamen

Die einzelnen Anweisungen sind im Kapitel „Anweisungen des EDT“ auf Seite 147ff. ausführlich beschrieben.

Ein Element vom Typ X kann selber wieder eine vollständige Bibliothek sein. Sie kann zwar mit dem EDT bearbeitet werden, doch ihre Struktur wird hierdurch zerstört.

Delta-Elemente können nicht direkt bearbeitet werden. Zum Bearbeiten von Delta-Elementen mit dem LMS (ab LMS V3.0A) geben Sie im EDT folgendes Anweisung ein:
@USE COM='!'(LMSEDT,$LMSLIB)

Weitere Information zu Bibliotheken siehe Handbuch „LMS“ [13].

3.8 SDF-Unterstützung beim Schreiben von Systemprozeduren

In Systemen, in denen SDF ab V3.0 installiert ist, kann eine in einer EDT-Arbeitsdatei erstellte oder eingelesene Systemprozedur auf Syntaxfehler geprüft und gegebenenfalls korrigiert werden, ohne den EDT zu verlassen.

Mit dem EDT können:

– der Inhalt einer Zeile oder eines Zeilenbereiches an SDF zur Syntaxkontrolle übergeben werden und abhängig von der Einstellung der SDF-Optionen fehlerhafte oder fehlende Operanden von Kommandos und Anweisungen im Korrektur-Dialog von SDF korrigiert werden (@SDFTEST und T-Kurzanweisung). Wird der Korrektur-Dialog abgebrochen oder ist kein Korrektur-Dialog möglich, wird die fehlerhafte Zeile überschreibbar an der obersten Fensterposition angezeigt und eine Fehlermeldung ausgegeben. Ist die Syntax korrekt, bzw. wurde sie korrigiert, werden die Kommandos und Anweisungen in die EDT-Arbeitsdatei übernommen.

– der interne Name eine Programms voreingestellt werden (@PAR SDF-PROGRAM). Bei Angabe eines Programmnamens, dem eine SDF-Syntaxdatei zugeordnet ist, werden auch Anweisungen dieses Programmes syntaktisch geprüft.

– Informationen über SDF-Syntaxdateien und eingestellte SDF-Optionen und den eingestellten internen Programmnamen am Bildschirm ausgegeben oder in eine Datei geschrieben werden (@STATUS=SDF)

Bei der Syntaxprüfung unterscheidet der EDT 3 Arten von Zeileninhalten:

1. Zeilen, die mit einem (nur einem) / in Spalte 1 beginnen.
   Sie werden gemäß der SDF-Syntaxdatei-Hierarchie auf Kommando-Syntax geprüft. Die Zulässigkeit bezüglich Privilegien oder Systemumgebung (z.B. Stapelprozess oder Prozedur) ist durch den aktuellen Benutzer und die aktuelle Umgebung bestimmt.
2. Zeilen, die mit // beginnen.
   Sie werden an SDF zur Anweisungsüberprüfung übergeben.
   Der Programmname wird durch die Anweisung @PAR SDF-PROGRAM voreingestellt, bzw. ist durch eine vorangegangene Anweisung @SDFTEST PROGRAM=name bekannt. Der Programmname muß in einer aktuellen SDF-Syntaxdatei bekannt sein.
3. reine Datenzeilen.
   Sie werden nicht geprüft.

Fehlerhafte Operanden bei ISP-Kommandos werden von SDF nicht erkannt.

3.9 Extended Host Code Support (XHCS)

Rechenanlagen und Datensichtstationen arbeiten mit je einem Satz von Buchstaben, Ziffern und Zeichen, aus denen Wörter und andere elementare Bestandteile einer Sprache aufgebaut sind, dem sogenannten Zeichensatz (Character Set).

Durch die Erweiterung dieser Zeichensätze können landesspezifische Zeichendarstellungen, wie z.B. Umlaute (deutsch) oder Akzente (französisch), innerhalb eines Zeichensatzes angeboten werden.

Ein codierter Zeichensatz (Coded Character Set, CCS) ist die eindeutige Darstellung der Zeichen eines Zeichensatzes in binärer Form. Der Inhalt eines codierten Zeichensatzes und seine Regeln, wie z.B. die Sortierreihenfolge und Konvertierungsvorschriften, sind durch internationale Normen festgelegt.

Beispiel: Das Zeichen ä ist im codierten Zeichensatz EBCDIC.DF.03 (deutsche Referenzversion) durch das Byte X'FB', in EBCDIC.DF.04-1 durch X'43' dargestellt.

Jeder codierte Zeichensatz (kurz: Code) wird durch seinen eindeutigen Namen (Coded Character Set Name, CCSN) bestimmt.

Beispiel: der Code EBCDIC.DF.03 (Internationale Referenzversion) hat den Namen EDF03IRV.

Eine Liste der existierenden Codes befindet sich im Handbuch „XHCS“ [6].

3.9.1 XHCS und EDT

Der EDT verwendet Umsetztabellen zur Ermittlung der nichtdarstellbaren Zeichen und zur Klein-/Groß-Umsetzung bei LOWER ON.

Ab BS2000/OSD-BC V1.0 fordert der EDT die Umsetztabellen vom Subsystem XHCS (Extended Host Code Support) an und verwendet diese anstatt der fest definierten EBCDIC.DF.03-Tabellen.

Ist das Subsystem XHCS nicht verfügbar, werden auch ab BS2000/OSD-BC V1.0 wie bisher die Standard-EDT-Umsetztabellen auf der Basis von EBCDIC.DF.03 verwendet.

Bei der Initialisierung des EDT wird der Benutzerstandard-Code eingestellt, wenn dieser durch das Kommando /MODIFY-TERMINAL-OPTIONS aktiviert ist. Im Stapel- oder Prozedurbetrieb wird der CCSN der mit RDATA eingelesenen Prozedurdatei eingestellt. Die Klein-/Groß-Umsetzung erfolgt im Dialogbetrieb durch VTSU (MODE=LINE).

Mit dem EDT können auch Daten bearbeitet werden, die Binärwerte oder gepackte Zahlen enthalten. Bei der Konvertierung von einem CCS zu einem anderen CCS kann es dabei zu unerwünschten Datenverfälschungen kommen. Der EDT macht deshalb grundsätzlich keine Konvertierung. Innerhalb einer aktuellen EDT-Anwendung muß eine homogene Code-Umgebung vorhanden sein, d.h. es kann immer nur ein CCS verwendet werden.

Wird z.B. eine Datei A mit dem Code XC1 bearbeitet, sind alle Arbeitsdateien auf diesen Code eingestellt. Das weitere Einlesen (Mischen) von Dateien ist nur bei gleichem Code zulässig. Bei unterschiedlichem Code werden die Anweisungen @COPY, @GET, @INPUT und @READ abgewiesen. Soll eine Datei B mit dem Code XC2 eingelesen werden, dann muß die Bearbeitung der Datei A zuerst abgeschlossen werden, d.h. die Arbeitsdatei muß mit @DELETE geleert oder (nach Eröffnen mit @OPEN) mit @CLOSE geschlossen werden.

Druckausgaben des EDT (@LIST-Anweisung und Protokollierung auf SYSOUT im Stapelbetrieb) werden wie bisher als Hexadezimal-Zeichenfolgen ohne Code-Merkmal ausgegeben.

Zeichenfolgevariable (#S00 - #S20) des EDT werden ohne Berücksichtigung eines Code-Merkmals behandelt.

Jobvariable und S-Variable werden ohne Berücksichtigung eines Code-Merkmals als Hexadezimal-Zeichenfolgen gelesen und geschrieben.

Erweiterung für die Sprachen Arabisch und Farsi

Der EDT unterstützt die Sprachen Arabisch und Farsi. Der wesentliche Unterschied zu diesen Sprachen besteht in der Schreibrichtung (von rechts nach links). Die Beschreibung der geänderten bzw. erweiterten Funktionen, die der EDT zur Unterstützung der Sprachen Arabisch und Farsi anbietet, sind in folgenden Handbüchern beschrieben:

„Additional Information for Arabic“ [11] (das Handbuch ist nur in englischer Sprache erhältlich).

„Additional Information for Farsi“ [12] (das Handbuch ist nur in englischer Sprache erhältlich).

3.9.2 XHCS im EDT-Dialogbetrieb

Beim Aufruf des EDT wird der Benutzerstandard-Code eingestellt, wenn dieser durch das Kommando /MODIFY-TERMINAL-OPTIONS aktiviert ist. Der CCSN wird im EDT-Datenbereich abgespeichert und gilt global für alle Arbeitsdateien. Dieser CCSN bleibt solange gültig, bis explizit oder implizit auf einen anderen CCSN umgeschaltet wird. Im 7-Bit-Modus (aktueller CCSN=EDF03IRV) erhalten die erzeugten Dateien bzw. Bibliothekselemente das Codemerkmal „Blank“ (CCSN=␣␣␣␣␣␣␣␣).

Vom Subsystem XHCS wird die Tabelle der darstellbaren Zeichen, sowie die Klein-/Groß-Umsetztabelle angefordert. Vor der Ausgabe auf die Datensichtstation wird der Videopuffer mit diesen Tabellen (abhängig von LOWER ON/OFF) umgesetzt.

Bei den Ein-/Ausgaben auf die Datensichtstation (WROUT, WRTRD, RDATA) wird der CCSN über VTSU (im VTSUCB) als Operand mitgegeben.

Mit @WRITE oder @SAVE erzeugte Dateien bzw. Bibliothekselemente bekommen den CCSN als Codemerkmal im Katalog bzw. in der Bibliothek eingetragen. Ein eventueller bereits existierender CCSN wird überschrieben. Im 7-Bit-Modus (aktueller CCSN=EDF03IRV) werden die Standard-EDT-Umsetztabellen verwendet.

Umschalten des Zeichensatzes

Der im EDT eingestellte Zeichensatz kann explizit oder implizit umgeschaltet werden. Damit diese Umschaltung ohne Datenverfälschung möglich ist, müssen folgende Voraussetzungen erfüllt sein:

– In den EDT-Arbeitsdateien sind keine Daten mit einem anderen CCSN (d.h. alle Arbeitsdateien des EDT sind leer)

– Der CCSN ist in der Liste der gültigen CCSN für die Datensichtstation enthalten (d.h. die Datensichtstation kann den Zeichensatz anzeigen)

– Der codierte Zeichensatz (CCS) ist kein ISO-Code und kein anderer 7-Bit-Code als EDF03IRV

Sind diese Bedingungen erfüllt, wird vom Subsystem XHCS die Tabelle der darstellbaren Zeichen, sowie die Klein-/Groß-Umsetztabelle angefordert. Sind diese Bedingungen nicht erfüllt, wird die Umschaltung mit einer Meldung abgewiesen und der eingestellte CCSN bleibt weiterhin gültig.

Das implizite Umschalten des Zeichensatzes erfolgt durch Einlesen einer Datei bzw. eines Bibliothekselementes mit einem anderen CCSN. Enthält eine Datei bzw. ein Bibliothekselement einen CCSN mit dem Wert „Blank“, wird der CCSN EDF03IRV angenommen. Beim Umschalten auf den 7-Bit codierten Zeichensatz (CCS) EDF03IRV werden die Standard-EDT-Umsetztabellen verwendet.

Das explizite Umschalten des Zeichensatzes erfolgt durch die EDT-Anweisung @CODENAME.

3.9.3 XHCS im EDT-Prozedurbetrieb

Im Prozedur- und Stapelbetrieb wird der EDT aus einer BS2000-Prozedur heraus gestartet, wobei die Anweisungen mit RDATA gelesen werden.

Ist der CCSN der mit RDATA gelesenen Datei bzw. des Bibliothekselementes ungleich „blank“, dann wird dieser für den EDT eingestellt, andernfalls wird der CCSN EDF03IRV eingestellt.

Eine Umschaltung auf einen anderen CCSN ist im Prozedur- und Stapelbetrieb nicht möglich.

3.10 Jobvariable

In Systemen, in denen das Subsystem „Jobvariablen Support“ installiert ist, können Jobvariable (JV) genutzt werden.

Im EDT kann man:

- Einträge von Jobvariablen aus dem Katalog löschen (@ERAJV)
- Werte von Jobvariablen
  - am Bildschirm ausgeben
  - in eine Arbeitsdatei schreiben
  - einer Zeichenfolge zuordnen (@GETJV)
- Jobvariable in den Katalog eintragen (@SETJV)
- Jobvariablen Werte zuweisen (@SETJV)
- Informationen über Jobvariable
  - am Bildschirm ausgeben
  - in eine Arbeitsdatei schreiben (@STAJV)

Der EDT-Lauf kann mit einer Monitor-Jobvariablen überwacht werden (siehe Abschnitt „Überwachung des EDT-Laufs mit Monitor-Jobvariablen“ auf Seite 33).

Weitere Information zu Jobvariablen siehe Handbuch „Jobvariablen“ [7].

3.11 SDF-P-Unterstützung

In Systemen, in denen das Subsystem „SDF-P“ installiert ist, können S-Variable genutzt werden.

Im EDT kann man:

- Inhalte von S-Variablen vom Typ STRING und INTEGER
  - am Bildschirm ausgeben
  - einer Zeichenfolge zuordnen (@GETVAR)
- S-Variable deklarieren (@SETVAR)
- S-Variablen Werte zuweisen (@SETVAR)
- Inhalte von zusammengesetzten S-Variablen vom Typ LIST (Listenvariable)
  - löschen (@SETLIST MODE=OVERWRITE)
  - Listenvariable erweitern (@SETLIST]
  - neu bBeschreiben (@SETLIST)
  - einlesen (@GETLIST). Die Elemente der Listenvariablen müssen vom Typ STRING sein

Ab BS2000/OSD-BC V1.0 liefert der EDT bei normalem Beenden durch @HALT, @RETURN oder im Dialog durch @END und bei nicht normalem Beenden einen Kommando-Returncode, der von SDF-P zur Steuerung von S-Prozeduren verwendet werden kann (siehe Abschnitt „Kommando-Returncode des EDT“ auf Seite 31).

Weitere Information zu S-Variablen siehe Handbuch „SDF-P“ [8].

3.12 Auftragsschalter

Es gibt 4 Auftragsschalter, deren Stellung der EDT zur Ablaufsteuerung auswertet. Vor dem EDT-Lauf können die Schalter mit dem Systemkommando MODIFY-JOB-SWITCHES gesetzt oder zurückgesetzt werden. Während des EDT-Laufs kann man dazu auch @SETSW benutzen.

Auftragsschalter 4

Dialogbetrieb, Stapelbetrieb:
Wurde der Auftragsschalter 4 vor dem Laden des EDT gesetzt, werden nach dem Laden die Meldung BLS0500 und nach dem Beenden des EDT die Meldung % EDT8000 EDT NORMAL END unterdrückt. Ebenfalls unterbleiben die Meldungen: % EDT0900 EDITED FILE(S) NOT SAVED! und % EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)

Stapelbetrieb:
Ist der Auftragsschalter 4 vor dem Laden des EDT gesetzt, wird @LOG NONE eingestellt, d.h. während des Ablaufs des EDT wird nichts protokolliert.

Auftragsschalter 5

Wurde der Schalter 5 vor dem Laden des EDT gesetzt, wird der L-Modus eingestellt. Der EDT liest die Eingaben mit RDATA von SYSDTA. Dasselbe (Lesen von SYSDTA mit RDATA) erreicht man durch Eingabe von @EDIT ONLY am Bildschirm. Anstelle der aktuellen Zeilennummer gibt der EDT im Dialogbetrieb * aus.

Bei gesetztem Auftragsschalter 5 ist die kompatible Syntaxkontrolle des L-Modus voreingestellt (siehe @SYNTAX SECURITY=LOW).

Wird die Schalterstellung während des EDT-Laufs geändert, hat dies keine Wirkung.

Mit @EDIT ohne Operanden wird in den L-Modus gewechselt und mit @EDIT FULL SCREEN in den F-Modus.

Auftragsschalter 6

Normalerweise druckt der EDT nicht mehr als 132 Zeichen pro Zeile. Der Rest wird in den folgenden Zeilen gedruckt. Ist der Auftragsschalter 6 gesetzt, druckt der EDT bis zu 160 Zeichen pro Zeile. Der Rest wird nach wie vor in den folgenden Zeilen gedruckt.

Der Auftragsschalter 6 muß vor dem Laden des EDT gesetzt werden.

Auftragsschalter 7

Dieser Schalter kann sowohl vor dem Laden des EDT als auch während des EDT-Laufs gesetzt werden. Er verhindert die automatische Freigabe von vorab zugewiesenem, überschüssigem Speicherplatz durch den EDT, der im Normalfall nicht belegten Speicherplatz durch den FILE-Marco freigibt (siehe Abschnitt „Dateibearbeitung“ auf Seite 40ff.).

4 Arbeitsmodi des EDT

Im EDT stehen zwei Arbeitsmodi für die Bearbeitung von Daten zur Verfügung:

– Im FULL-SCREEN-Modus (F-Modus) steht in 10 Arbeitsdateien (0-9) der gesamte Bildschirm für die Eingabe von Daten und Anweisungen zur Verfügung.

– Im LINE-Modus (L-Modus) wird in 23 Arbeitsdateien (0-22) jeweils nur eine Bildschirmzeile zur Eingabe von Daten und Anweisungen angeboten.
Zur Unterscheidung von Datensätzen und Anweisungen müssen Anweisungen mit einem @ eingegeben werden.

4.1 F-Modus

Im FULL-SCREEN-Modus (F-Modus) bietet der EDT bildschirmorientierte Dateibearbeitung für SAM- und ISAM-Dateien sowie für Elemente aus Programm-Bibliotheken und für POSIX-Dateien an. Insgesamt stehen dem Benutzer dafür 10 Arbeitsdateien (0-9) zur Verfügung.

Bildschirmorientiert heißt, daß im Datenbereich, der am Bildschirm dargestellt wird,

– Daten in beliebiger Reihenfolge überschrieben werden können,
– Text ein- und ausgefügt werden kann, ohne daß auf die Zeilenstruktur geachtet werden muß.

Neben der Möglichkeit, Änderungen direkt am Bildschirm vorzunehmen, kann der Benutzer die Dateibearbeitung steuern durch:

– Anweisungen in der Anweisungszeile
– Kurzanweisungen in der Markierungsspalte
– Anweisungen im Datenfenster
– Satzmarkierungen
– Funktionstasten

Die formatierte Bildschirmausgabe wird als Arbeitsfenster bezeichnet. Im Arbeitsfenster werden die Daten der Arbeitsdatei dargestellt, die durch Eingaben am Bildschirm oder durch Einlesen von SAM-, ISAM-Dateien oder Bibliothekselementen oder POSIX-Dateien in diese Arbeitsdatei geschrieben wurden.

Es besteht die Möglichkeit, vom F-Modus in den L-Modus umzuschalten (siehe @EDIT).

Besonderheiten der Datensichtstation 3270

Der EDT ist für die SIEMENS DSS 8160, 9750 und aufwärtskompatible Geräte und deren Eigenschaften konzipiert.

Die Geräteeigenschaften der DSS 3270 unterscheiden sich davon sehr stark, deshalb kann die Anpassung des EDT aus ergonomischer Sicht nicht perfekt sein.

Bei der Unterstützung der DSS 3270 gibt es wegen der unterschiedlichen Geräteeigenschaften für den EDT einige funktionelle Einschränkungen sowie geringfügige Abweichungen an der Bildschirm-Benutzerschnittstelle.

Dafür sind vor allem folgende Geräte-Eigenschaften ausschlaggebend:

– Die Gerätesteuerzeichen (ASZ, FBZ) belegen ein sichtbares Byte (Leerzeichen) am Bildschirm, dadurch verkleinert sich die mögliche Anzahl der Zeichen einer Zeile.

– Bei der Bildschirmdarstellung gibt es keine NIL-Zeichen (X’00’). Statt dessen werden Leerzeichen (X’40’) angezeigt. Die als Leerzeichen angezeigten NIL-Zeichen werden nicht zum Rechner übertragen.

– Als Ersatzdarstellung für nichtdarstellbare Codierungen wird das Zeichen X’44’ (japanisches Währungssymbol) verwendet.

– Die Tasten Feldmarke und DUP haben für den EDT keine Bedeutung und werden im F-Modus mit dem Fragezeichen ? abgewiesen. Im L-Modus wird dafür das Zeichen X’44’ zum Rechner übertragen.

4.1.1 Das Arbeitsfenster

Das Arbeitsfenster unterteilt den Bildschirm in Felder mit unterschiedlichen Funktionen. Das folgende Bild stellt den Aufbau des Arbeitsfensters schematisch dar.

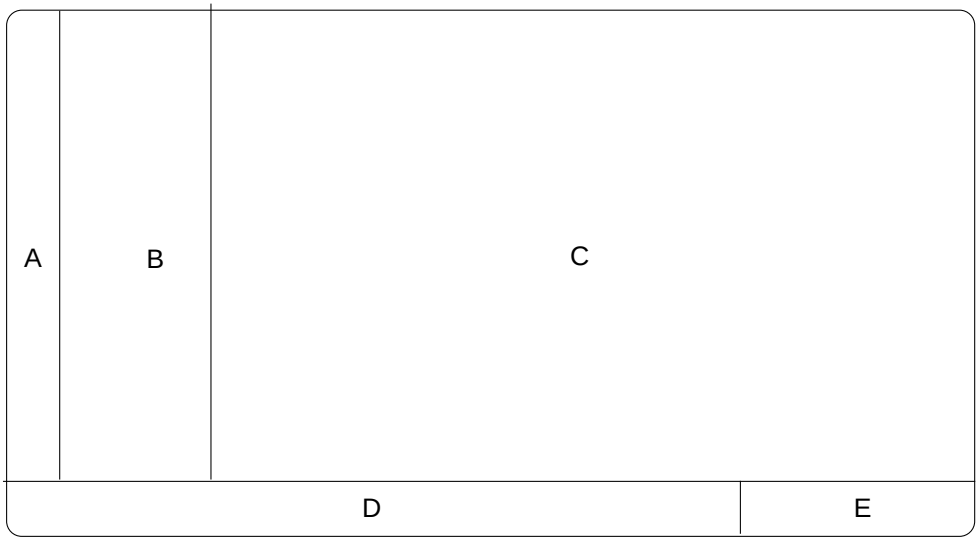


Bild 6: Standard-Arbeitsfensterformat mit eingeschalteter Zeilennummernanzeige

A = Markierungsspalte

B = Zeilennummernanzeige

C = Datenfenster

D = Anweisungszeile

E = Zustandsanzeige

Markierungsspalte

In der Markierungsspalte können durch 1 Zeichen lange Kurzanweisungen Funktionen ausgelöst werden.

Werden am Datenfenster Sätze dargestellt, ist standardmäßig die Markierungsspalte überschreibbar und das Datenfenster vor Überschreiben geschützt. Erst durch Kurzanweisungen in der Markierungsspalte oder durch Datenübertragung mit F2 werden die Datenfensterzeilen auf überschreibbar gestellt. In den überschreibbaren Zeilen können dann keine Kurzanweisungen angegeben werden.

Durch die Anweisung @PAR EDIT FULL=ON können bei eingeschalteter Zeilennummernanzeige (@PAR INDEX=ON) das Datenfenster und die Markierungsspalte auf überschreibbar gestellt werden. Es ist möglich, eine Zeile zu markieren und gleichzeitig Daten in dieser Zeile zu ändern (siehe @PAR EDIT FULL).

Fehleingaben in der Markierungsspalte können durch Überschreiben mit Leerzeichen bzw. NIL-Zeichen gelöscht werden.

Zeilennummernanzeige

Nach Aufruf des EDT wird standardmäßig die Zeilennummernanzeige ausgegeben. Sie kann mit @PAR INDEX=OFF unterdrückt werden.

Abgesehen von der ersten Stelle der Zeilennummernanzeige, die zugleich die Markierungsspalte ist, ist die Zeilennummernanzeige nicht überschreibbar.

Die Zeilennummer wird 6-stellig angezeigt. Vier Stellen stehen vor dem Dezimalpunkt, zwei danach.

Die vollständige Zeilennummer mit ihren insgesamt vier Stellen nach dem Dezimalpunkt wird nur im L-Modus dargestellt.

Datenfenster

Im Datenfenster wird die aktuelle Arbeitsdatei dargestellt. Eine Arbeitsdatei besteht aus Sätzen. Diese Sätze werden in die Zeilen des Datenfensters ausgegeben, wobei ein Satz auch länger sein kann als eine Datenfensterzeile. In diesem Fall ist nur ein Teil des Satzes im Datenfenster sichtbar. Das Datenfenster stellt einen Ausschnitt der Arbeitsdatei dar. Es kann durch Positionieren verschoben werden.

Sätze, die länger als die Datenfensterzeile sind, können im EDIT LONG-Modus vollständig dargestellt werden (siehe @PAR EDIT LONG).

Enthält die Datei weniger Sätze als das Datenfenster Zeilen hat, werden die restlichen Zeilen mit Füllzeichen (standardmäßig NIL-Zeichen) aufgefüllt und auf überschreibbar gestellt.

Nach Aufruf des EDT erscheint die leere Arbeitsdatei 0 auf dem Bildschirm.

Standardmäßig sind die Sätze im Datenfenster nicht überschreibbar. Zum Ändern müssen die Sätze in der Markierungsspalte markiert oder der Bildschirm mit F2 auf überschreibbar gestellt werden. Im EDIT-FULL-Modus, der mit @PAR EDIT FULL = ON eingestellt wird, sind alle Sätze des Datenfensters immer überschreibbar. Gleichzeitig können auch Kurzanweisungen in der Markierungsspalte angegeben werden (siehe @PAR EDIT FULL).

Der F-Modus benötigt zum Kennzeichnen des Satzendes kein Satzendezeichen. Satzende ist das letzte von NIL- oder Füllzeichen verschiedene Zeichen eines Satzes. NIL- oder Füllzeichen am Ende eines Satzes werden ignoriert.

Unabhängig davon, ob der Bildschirm mit F2, DUE bzw. DUE1 abgeschickt wird, werden Eingaben im Datenfenster in die Arbeitsdatei übertragen.

Füllzeichen

Über die Anweisung @SYMBOLS FILLER = kann das Füllzeichen definiert werden, das zwischen Satzende und Bildschirmzeilenende eingesetzt wird. Standardmäßig ist das Füllzeichen ein NIL-Zeichen (siehe @SYMBOLS).

Behandlung von Füllzeichen im Datenfenster

Bei der Erfassung eines Satzes (durch Eintippen in eine leere Datei, Weiterschreiben am Dateiende, Einfügen in Zeilen, die nach Markieren zum Einfügen angeboten werden), werden NIL-Zeichen vor oder zwischen sonstigen Zeichen in Leerzeichen umgesetzt.

Beim Ändern bereits existierender Sätze werden Füllzeichen innerhalb eines Satzes als Leerzeichen in die Datei übernommen.

Im EDIT LONG- bzw. im HEX-Modus werden Füllzeichen innerhalb einer Zeile in die Datei übernommen. Füllzeichen am Ende einer Zeile werden ignoriert.

Bildschirmzeilen, die nur aus Füllzeichen ungleich '␣' bestehen, werden nicht in die Datei aufgenommen. Ein Satz kann also gelöscht werden, indem der am Datenfenster sichtbare Teil mit Füllzeichen überschrieben wird.

Bildschirmzeilen, die nur aus Füllzeichen = '␣' bestehen, werden als Datensätze bestehend aus zwei Leerzeichen angelegt.

Behandlung von NIL-Zeichen im Datenfenster

Enthält eine Zeile nur NIL-Zeichen, wird sie nicht als Satz angelegt.

Bei der Erfassung eines Satzes (durch Eintippen in eine leere Datei, Weiterschreiben am Dateiende, Einfügen in Zeilen, die nach Markieren zum Einfügen angeboten werden), werden NIL-Zeichen vor oder zwischen sonstigen Zeichen in Leerzeichen umgesetzt.

Falls das Füllzeichen von NIL verschieden ist, werden beim Ändern bereits existierender Sätze NIL-Zeichen innerhalb eines Satzes als editierbare Zeichen behandelt, d.h. X'00' wird als NIL-Zeichen am Bildschirm dargestellt und beim Übertragen unverändert in die Datei übernommen. Im EDIT LONG- bzw. im HEX-Modus werden NIL-Zeichen auch bei der Ersterfassung in die Datei übernommen.

NIL-Zeichen am Ende einer Zeile werden ignoriert. Ist ein Satz länger als der am Bildschirm dargestellte Teil, bewirken NIL-Zeichen am Ende der Bildschirmzeile, daß der restliche Text an die erste Stelle ungleich NIL-Zeichen herangezogen wird, der Satz also verkürzt wird.

Zum Löschen eines ganzen Datensatzes können LZE und LZF nur bedingt verwendet werden.

– LZE löscht alle Zeichen des Datensatzes ab der eingegebenen Position.

– LZF löscht nur den Zeilenrest, etwaige Zeichen des Datensatzes außerhalb des Datenfensters werden nachgezogen.

Ein ganzer Datensatz bei Spaltenposition ungleich 1 muß explizit mit @DELETE oder mit der Kurzanweisung D gelöscht werden.

Durch @SYMBOLS FILLER = '␣' wird die bis zu EDT V16.2 entsprechende Darstellungsform eingestellt.

Nicht darstellbare Zeichen im Text

Enthält eine Datei am Bildschirm nicht darstellbare Zeichen, werden diese Zeichen als Schmierzeichen ausgegeben.

Wird ein solcher Satz geändert, wird anstelle des Schmierzeichens das ursprüngliche Zeichen in die Datei eingesetzt. Verschiebt sich durch Einfügen oder Ausfügen (EFG/AFG) die Position des Schmierzeichens im Satz, dann wird an die Stelle des Schmierzeichens ein Fragezeichen ? gesetzt und die Zeile geschützt dargestellt mit einem '?' in der Markierungsspalte. Der ursprüngliche Inhalt des Satzes bleibt erhalten.

i Im LOWER OFF-Modus werden Kleinbuchstaben in der Datei als Schmierzeichen ausgegeben. Auf diese Weise soll der Benutzer aufmerksam gemacht werden, daß er den „falschen" Modus eingeschaltet hat. Texte, die nicht abbildbare Zeichen enthalten, sollten im Hexadezimal-Modus (siehe @PAR HEX) oder Codier-Modus (siehe @CODE) erfaßt werden.

Anweisungszeile

Eingaben in der Anweisungszeile werden als Anweisungen interpretiert. Eine Übersicht über die Anweisungen des F-Modus wird im Kapitel „Anweisungen des EDT" auf Seite 147ff. gegeben. Das EDT-Anweisungsfluchtsymbol (@) muß nicht angegeben werden.

Der Benutzer kann eine oder mehrere Anweisungen (Anweisungsfolge) in der Anweisungszeile eingeben. Die einzelnen Anweisungen sind durch ein Semikolon (;) zu trennen. Tritt ein Fehler auf, wird die Abarbeitung abgebrochen. Es werden eine Fehlermeldung und der nicht bearbeitete Teil der Anweisungseingabe ausgegeben, einschließlich der fehlerhaften Anweisung.

Nach korrekter Abarbeitung einer Eingabe wird die Anweisungszeile bei der Bildschirmausgabe gelöscht. Die zuletzt eingegebene Anweisung kann jedoch durch Eingabe von # wieder sichtbar gemacht werden, um sie erneut, verändert oder unverändert, absetzen zu können. In diesem Fall muß jedoch mindestens ein Zeichen überschrieben, geändert bzw. hinzugefügt werden. Der Inhalt einer Anweisungszeile bzw. ein nicht benötigter Zeilenrest kann mit LZF gelöscht werden.

Innerhalb einer Zeichenfolge zwischen Hochkomma wird ein Semikolon (;) nicht als Anweisungstrennzeichen interpretiert.

Bei einem Wechsel mit @EDIT in den L-Modus innerhalb einer Anweisungsfolge wird ein evtl. Rest der Anweisungsfolge nicht abgearbeitet.

Anweisungszeile-Fortsetzung

Ist nach Abschicken des Bildschirms das letzte Zeichen der Anweisungszeile kein NIL-Zeichen oder Leerzeichen, wird angenommen, daß der Benutzer für die Eingabe einen Fortsetzungsbereich benötigt. In diesem Fall wird ihm eine zweite Zeile angeboten. Der Inhalt der Anweisungszeile wird vom EDT in die Zeile davor gebracht und die nun leere Anweisungszeile wird als Fortsetzungszeile angeboten. Maximal werden zwei Fortsetzungszeilen angeboten, d.h. die maximale Eingabelänge beträgt 198 Zeichen.

Behandlung von Leerzeichen in der Anweisungszeile

Führende Leerzeichen vor Anweisungen und Leerzeichen zwischen Schlüsselwörtern (Operanden) werden ignoriert. Leerzeichen innerhalb von Schlüsselwörtern sind nicht erlaubt.

Behandlung von NIL-Zeichen in der Anweisungszeile

Ein NIL-Zeichen an der letzten Stelle der Anweisungszeile signalisiert das Ende der Anweisungseingabe. Vor der Analyse der Eingabe werden NIL-Zeichen innerhalb der Anweisungsfolge in Leerzeichen umgesetzt.

Zustandsanzeige

Die Zustandsanzeige zeigt, von links nach rechts gelesen:

– die Zeilennummer der ersten Zeile des Arbeitsfensters (6stellig)

– die Spaltennummer, ab der die Sätze im Datenfenster dargestellt werden (3stellig)
– die Nummer der dargestellten Arbeitsdatei (in runden Klammern)

Die Zustandsanzeige ist nicht überschreibbar.

Beispiel

```
                                                          0008.00:001(3)
```

Zeile 8.00 ist die erste Zeile des Arbeitsfensters 3.

Verändern des Arbeitsfensters

Der Benutzer kann das Format des Arbeitsfensters verändern, indem er

– die Zeilennummernanzeige unterdrückt
– den Bildschirm in zwei Arbeitsfenster aufteilt

Unterdrücken der Zeilennummernanzeige

Mit @PAR INDEX=OFF wird das Datenfenster auf alle 80 Zeichen (DSS 3270: 77 Zeichen) der Bildschirmzeile erweitert. Das erste Zeichen jeder Zeile ist dabei überschreibbar und entspricht der Markierungsspalte.

Aufteilen des Bildschirms

Wird der Bildschirm in zwei Arbeitsfenster aufgeteilt (siehe @PAR SPLIT), spricht man von einem gesplitteten Bildschirm. Das obere Arbeitsfenster wird Arbeitsfenster 1 oder erstes Arbeitsfenster genannt, das untere Arbeitsfenster 2 oder zweites Arbeitsfenster. Ein Arbeitsfenster umfaßt mindestens zwei Zeilen, von denen eine die Anweisungszeile ist.

Beispiel

```
par split 10 $3 ............................................ 0000.00:001(0)
```

Der Bildschirm soll in zwei Arbeitsfenster aufgeteilt werden. Dazu wird in der Anweisungszeile @PAR SPLIT 10 $3 eingegeben und mit DUE abgeschickt.

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..................................................................
   7.00 ..................................................................
   8.00 ..................................................................
   9.00 ..................................................................
  10.00 ..................................................................
  11.00 ..................................................................
  12.00 ..................................................................
  13.00 ..................................................................
.....................................................................0001.00:001(0)
   1.00 SIE KOENNEN JETZT..................................................
   2.00 ABWECHSELND ODER GLEICHZEITIG.....................................
   3.00 DIE ARBEITSDATEIEN 0 UND 3.........................................
   4.00 BEARBEITEN.........................................................
   5.00 ..................................................................
   6.00 ..................................................................
   7.00 ..................................................................
   8.00 ..................................................................
   9.00 ..................................................................
.....................................................................0001.00:001(3)
```

Das obere Arbeitsfenster (Arbeitsfenster 1) ist das verkleinerte Arbeitsfenster, in dem @PAR SPLIT eingegeben wurde.

Das untere Arbeitsfenster (Arbeitsfenster 2) wird zusätzlich ausgegeben. Es umfaßt 10 Zeilen einschließlich der Anweisungszeile und gibt die Arbeitsdatei 3 aus (@PAR SPLIT 10 $3).

Abarbeitungsreihenfolge

Abarbeitungsreihenfolge bei einem Arbeitsfenster:

1. Datenfensterauswertung
2. Kurzanweisungen in der Markierungsspalte
3. Anweisung in der Anweisungszeile

Solange in der Markierungsspalte Einfüge- bzw. Änderungsmarkierungen vorhanden sind oder die Dauereinfügefunktion eingeschaltet ist (siehe Abschnitt „n/I Einfügen von Zeilen“ auf Seite 89ff.), werden nur Datenfenster und Markierungsspalte ausgewertet. Dabei werden zuerst die Sätze aus dem Datenfenster in die Datei übernommen. Anschließend wird die Markierungsspalte ausgewertet. Der Inhalt der Anweisungszeile bleibt unverändert und wird erst ausgewertet, wenn keine Einfüge- bzw. Änderungsmarkierungen mehr angegeben werden oder die Dauereinfügefunktion ausgeschaltet wird.

Abarbeitungsreihenfolge bei zwei Arbeitsfenstern:

1. Datenfensterauswertung des oberen Arbeitsfensters
2. Kurzanweisungen in der Markierungsspalte des oberen Arbeitsfensters
3. Datenfensterauswertung des unteren Arbeitsfensters
4. Kurzanweisungen in der Markierungsspalte des unteren Arbeitsfensters
5. Anweisung in der oberen Anweisungszeile
6. Anweisung in der unteren Anweisungszeile

Solange in der Markierungsspalte Einfüge- bzw. Änderungsmarkierungen vorhanden sind oder die Dauereinfügefunktion eingeschaltet ist, wird die oben genannte Reihenfolge unterbrochen und nur Datenfenster und Markierungsspalte des betroffenen Arbeitsfensters werden abgearbeitet. Werden keine Änderungs- bzw. Einfügemarkierungen mehr angegeben bzw. wird die Dauereinfügefunktion ausgeschaltet, werden anschließend Datenfenster und Markierungsspalte des anderen Arbeitsfensters abgearbeitet.

i Wird in der oberen Anweisungszeile mit @PAR SPLIT=OFF das Bildschirmformat auf ein Arbeitsfenster zurückgesetzt, werden Anweisungen in der unteren Anweisungszeile nicht mehr ausgeführt.

4.1.2 Die F-Tasten

Tasten / Verwendung	F1	F2	F3	F4	F5
Datenzeilen übernehmen	x	x	x	x	x
Kurzanweisungen		x		x	x
Marken setzen			x		
überschreibbar stellen		x		x	x
zu Sätzen mit gleicher Strukturtiefe positionieren	x				
zu Sätzen mit Satzmarkierungen positionieren			x		

F1 Positionieren zu Sätzen mit gleicher Strukturtiefe

Mit F1 kann zum nächsten Satz mit derselben Strukturtiefe wie der markierte positioniert werden (siehe Abschnitt „+ / – Positionieren des Arbeitsfensters nach der Strukturtiefe“ auf Seite 102ff.).

F2 Ändern aller Zeilen

Wird der Bildschirm mit F2 abgeschickt, wird das Datenfenster bzw. werden beide Datenfenster bei gesplittetem Bildschirm bei der nächsten Ausgabe auf überschreibbar gestellt.

Wurden Änderungen im Datenfenster vorgenommen oder Kurzanweisungen in der Markierungsspalte angegeben, werden zunächst nur die Änderungen im Datenfenster und die Kurzanweisungen ausgeführt. Danach wird das Datenfenster auf überschreibbar gestellt. Die Anweisungszeile wird noch nicht ausgewertet.

Erfolgte keine Eingabe in der Markierungsspalte, wird die Anweisungszeile abgearbeitet und anschließend das Datenfenster auf überschreibbar gestellt.

Tritt bei der Abarbeitung der Anweisungszeile ein Fehler auf oder wird vom EDT eine Meldung ausgegeben, wird das Datenfenster anschließend nicht auf überschreibbar gestellt.

F3 Bearbeiten von Satzmarkierungen

Mit F3 werden folgende Funktionen ausgelöst:

– Setzen von Satzmarkierungen
– Löschen von Satzmarkierungen
– Positionieren zu Sätzen mit Satzmarkierungen

4.1.3 Die K-Tasten

Tasten Verwendung	K1	K2	K3
F-Modus Bildschirmdialog und @CODE SHOW beenden	x		
EDT-Lauf unterbrechen		x	
Bildschirm wiederherstellen			x

K4 bis K15 haben keine Funktion.

K1 Beenden des F-Modus Bildschirmdialogs

Mit K1 wird der Bildschirmdialog abgebrochen und folgende Meldung ausgegeben: % EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO). Bei Eingabe von Y wird der Bildschirmdialog beendet. Bei N wird der Bildschirmdialog fortgesetzt.

Außerdem wird mit K1 nach @CODE SHOW die Anzeige der Codiertabelle beendet.

K2 Unterbrechung des EDT-Laufes

Unterbrechungen des EDT-Laufs mit Wechsel in den Systemmodus können neben @SYSTEM auch mit K2 erreicht werden.

Mit dem Kommando RESUME-PROGRAM gelangt man in den F-Modus zurück. Danach wird der gesamte Bildschirm neu ausgegeben.

Wird das Arbeitsfenster, in dem der EDT-Lauf unterbrochen wurde, nach RESUME-PROGRAM nicht oder nur unvollständig ausgegeben, kann der ursprüngliche Inhalt mit K3 wiederhergestellt werden.

Wird während der Unterbrechung eines der Kommandos START-PROGRAM oder LOAD-PROGRAM eingegeben oder werden Prozeduren gestartet, wird der EDT entladen.

K3 Wiederherstellen des Bildschirminhaltes

Wurde der Bildschirminhalt verschoben (z.B. durch eine Broadcast-Meldung), kann mit K3 der ursprüngliche Zustand wiederhergestellt werden.

4.1.4 Kurzanweisungen im F-Modus

Die Kurzanweisungen (Markierungen), sind Anweisungen von der Länge eines Zeichens. Sie werden in der Markierungsspalte als Groß- oder Kleinbuchstaben eingegeben.

Syntax- und Semantikprüfung

Vor der Bildschirmbearbeitung wird eine Syntax- und Semantikprüfung für die Kurzkommandospalte durchgeführt. Werden ungültige Kurzkommandos oder ungültige Kombinationen (z.B. M gefolgt von C) erkannt, erfolgt keine Eingabebearbeitung. Anstelle der fehlerhaften Kurzkommandos wird ein Fragezeichen ? ausgegeben. Die Schreibmarke wird auf die erste fehlerhafte Kurzanweisung positioniert.

Reihenfolge der Abarbeitung in der Markierungsspalte

In Abhängigkeit der verwendeten Funktionstaste bzw. der Kurzanweisungen werden bei der Abarbeitung der Markierungsspalte folgende Fälle unterschieden:

1. Wird F3 verwendet, dann wertet der EDT lediglich Kurzanweisungen aus, die mit F3 gesendet werden dürfen (Anweisungen zum Setzen und Löschen von Satzmarkierungen).
2. Wird eine der Zielmarkierungen A (after), B (before) oder O (on) angegeben, dann werden die Kurzanweisungen in folgender Reihenfolge ausgewertet:
   - die K-Markierung
   - alle D-Markierungen
   - die *-Markierung zum Löschen des Kopierpuffers
   - alle C, R, M zum Kopieren
   - alle A, B, O als Zielmarkierung beim Kopieren

   Andere Kurzanweisungen werden anschließend ausgeführt.
3. Wird weder ein S noch ein A, B oder O angegeben, ist folgende Abarbeitungsreihenfolge gültig:
   - die K-Markierung
   - alle D-Markierungen
   - die *-Markierung zum Löschen des Kopierpuffers
   - alle C, R, M-Markierungen zum Kopieren
   - alle T-Markierungen zum Testen von SDF-Syntax
   - alle X (Ändern), E (Einfügen von Zeichen), n und I (Einfügen von Zeilen)

Die Auswertung von X, E und I und n hängt von der Eingabereihenfolge ab. X, E hinter I bzw. n können verloren gehen, wenn aufgrund des Einfügebereichs die Zeilen nicht mehr am Bildschirm darstellbar sind. Eine Warnung erfolgt jedoch nicht.

Die Anweisungszeile wird nach der Abarbeitung der Kurzanweisungen ausgewertet.

Übersicht der Kurzanweisungen des EDT

Kurzanweisung	Funktion
*	Löschen des Kopierpuffers
A, B, O	Markieren einer Zeile als Zielort beim Kopieren
C	Markieren zum Kopieren
D	Löschen von Sätzen
E	Einfügen von Zeichen
J	Zusammenketten zweier Sätze
K	Kopieren einer Zeile in die Anweisungszeile
M	Kopieren und Löschen markierter Zeilen
n/I	Einfügen von Zeilen
R	Markieren zum Kopieren (ohne Löschen des Kopierpuffers)
S	Positionieren des Arbeitsfensters (horizontal und vertikal)
T	Syntaxtest durch SDF
+ / –	Positionieren des Arbeitsfensters (horizontal)
+ / – F1	Positionieren des Arbeitsfensters nach der Strukturtiefe
X	Ändern von Zeilen
D F3	Löschen einer Satzmarkierung
m F3	Setzen einer Satzmarkierung

* Löschen des Kopierpuffers

* löscht einen durch C, M oder R erzeugten Kopierpuffer.

Kurzanweisung	Taste
*	DUE oder F2

* wird vor A, B, O, C, M oder R ausgewertet, unabhängig davon, in welcher Zeile * angegeben wird. Dies bedeutet, daß auf jeden Fall zuerst der Kopierpuffer gelöscht wird, wenn eine Zeile mit * markiert wird.

Wird am gleichen Arbeitsfenster neben * auch eine C-, M- oder R-Markierung (ohne Angabe des Zielortes) angegeben, unterbleibt die Meldung % EDT0292 COPYBUFFER CLEARED, da Zeilen in den Kopierpuffer aufgenommen wurden.

Beispiel

Siehe Beispiel im Abschnitt „R Markieren zum Kopieren (ohne Löschen des Kopierpuffers)“ auf Seite 93.

A,B,O Markieren einer Zeile als Zielort

Diese Kurzanweisungen markieren den Zielort für die mit C, M und R markierten Zeilen.

Kurzanweisung	Taste
A B O	DÜE oder F2

A Hinter die mit A markierte Zeile werden die zu kopierenden Zeilen eingefügt.

B Vor die mit B markierte Zeile werden die zu kopierenden Zeilen eingefügt.

O Die mit dem Buchstaben O markierte Zeile wird mit den Zeilen überschrieben, die kopiert oder übertragen werden.
Dabei sind zwei Fälle zu unterscheiden:

- Die erste Spalte der Datei ist die erste Spalte im Datenfenster.
  Der gesamte Inhalt der mit O markierten Zeile wird mit der zu kopierenden Zeile überschrieben.
- Die erste Spalte der Datei wird nicht in der ersten Spalte im Datenfenster ausgegeben.
  Ist die Länge der zu kopierenden Zeile kleiner als die Länge der mit O markierten Zeile, bleiben die restlichen Teile der zu überschreibenden Zeile erhalten. Auf diese Weise kann ein Text einer Zeile in eine andere Zeile eingefügt oder angefügt werden. Dabei wird ab der eingeschalteten Spaltenposition überschrieben oder angefügt.

Werden mehrere zu kopierende Zeilen übertragen, werden sie entsprechend ihrer Anzahl auf die dem Zielort folgenden Zeilen übertragen. Wird dabei das Ende der Arbeitsdatei überschritten, werden ab dort neue Zeilen angelegt.

Die Zeilennummern der kopierten Zeilen vergibt der EDT nach 3 Prinzipien (siehe @COPY, Format 2):

1. Standardnumerierung mit der Schrittweite 1.0000, wenn durch @PAR INCREMENT nicht anders eingestellt.
2. Numerierung mit festgelegter Schrittweite (@PAR INCREMENT).
3. Automatische Numerierung und Umnumerierung (nur bei @PAR RENUMBER=ON). Der EDT numeriert automatisch um, wenn die Schrittweite zu groß ist, um alle kopierten Zeilen einzufügen. Näheres siehe @COPY, Format 2 „Berechnung der Zeilennummern“.

Der Kopiervorgang wird erst nach Abarbeitung von C-, M- und R-Markierungen ausgeführt, so daß in einem Arbeitsfenster in einem Dialogschritt die Zielmarkierung auch vor den zu kopierenden Zeilen angegeben werden kann.

Beispiel

```
c  1.00 DIE KURZANWEISUNGEN A, B UND O MARKIEREN DEN ZIELORT.....
a  2.00 FUER DIE MIT C, M UND R MARKIERTEN ZEILEN...............
   3.00 ...........................................................
```

Die Zeile 1.00 soll hinter die Zeile 2.00 kopiert werden. Dazu wird die Zeile 1.00 in der Markierungsspalte mit C und die Zeile 2.00 mit A markiert.

```
   1.00 DIE KURZANWEISUNGEN A, B UND O MARKIEREN DEN ZIELORT.....
m  2.00 FUER DIE MIT C, M UND R MARKIERTEN ZEILEN...............
o  3.00 DIE KURZANWEISUNGEN A, B UND O MARKIEREN DEN ZIELORT.....
   4.00 ...........................................................
```

Die Zeile 2.00 soll in die Zeile 3.00 übertragen werden. Dazu wird die Zeile 2.00 in der Markierungsspalte mit M und die Zeile 3.00 mit O markiert.

```
   1.00 DIE KURZANWEISUNGEN A, B UND O MARKIEREN DEN ZIELORT.....
   3.00 FUER DIE MIT C, M UND R MARKIERTEN ZEILEN...............
   4.00 ...........................................................
```

Die Zeile 3.00 wurde mit dem Inhalt der Zeile 2.00 überschrieben, und die Zeile 2.00 gelöscht.

C Markieren zum Kopieren

Mit C werden Zeilen zum Kopieren an einem durch A, B oder O angegebenen Zielort markiert. Die Zeilennummern (maximal 255) werden in einem Kopierpuffer zwischengespeichert. Sobald der Zielort angegeben ist, wird die Kopieranweisung ausgeführt und der Inhalt des Kopierpuffers gelöscht.

Kurzanweisung	Taste
C	DUE oder F2

Die Zeilennummern der kopierten Zeilen vergibt der EDT nach 3 Prinzipien (siehe @COPY, Format 2):

1. Standardnumerierung mit der Schrittweite 1.0000
2. Numerierung mit festgelegter Schrittweite (@PAR INCREMENT).
3. Automatische Numerierung und Umnumerierung (nur bei @PAR RENUMBER=ON). Der EDT numeriert automatisch um, wenn die Schrittweite zu groß ist, um alle kopierten Zeilen einzufügen. Näheres siehe @COPY, Format 2 „Berechnung der Zeilennummern“.

In einem Dialogschritt werden C, M und R nicht gleichzeitig durchgeführt. Die gemischte Angabe von C, M und R in der Markierungsspalte eines Arbeitsfensters wird durch eine Semantikprüfung abgewiesen. Anstelle der fehlerhaften Kurzanweisung wird ein Fragezeichen ? ausgegeben. Die Schreibmarke wird zum Korrigieren auf die fehlerhafte Kurzanweisung positioniert.

A, B, O und * löschen den Inhalt des Kopierpuffers. Ein mit C angelegter Kopierpuffer wird durch nachfolgende M oder R Kurzanweisungen gelöscht.

Bei einem gesplitteten Bildschirm können in einem Dialogschritt Zeilen vom ersten Arbeitsfenster ins zweite Arbeitsfenster kopiert werden.

Beim Kopieren vom zweiten Arbeitsfenster ins erste Arbeitsfenster sind, bedingt durch die Abarbeitungsreihenfolge, zwei Dialogschritte notwendig.

Der Kopierpuffer kann auch durch Markieren von Zeilen in verschiedenen Arbeitsdateien aufgebaut werden. Die Zielmarkierung kann in jeder Arbeitsdatei angegeben werden.

Der Kopierpuffer enthält Arbeitsdatei- und Zeilennummer der mit C markierten Sätze. Die Zeilennummern dürfen daher zwischen Markieren zum Kopieren mit C und der Ausführung des Kopiervorgangs durch Eingabe der Kurzanweisungen A, B oder O nicht verändert werden.

Beispiel

```
   1.00 C    MARKIEREN ZUM KOPIEREN.....................................
   2.00 ==========================.....................................
c  3.00  .....................................................................
   4.00 MIT C MARKIERTE ZEILEN WERDEN AN EINEN ANGEGEBENEN ZIELORT (A, B, O)....
   5.00 KOPIERT. DIE ZEILENNUMMERN (MAXIMAL 255) WERDEN IN EINEM.................
   6.00 KOPIERPUFFER ZWISCHENGESPEICHERT. .....................................
b  7.00 SOBALD DER ZIELORT ANGEGEBEN IST, WIRD DIE KOPIERANWEISUNG AUSGEFUEHRT,.
a  8.00 DER INHALT DES KOPIERPUFFERS GELOESCHT. ...............................
   9.00 .....................................................................
```

Die Zeile 3.00 soll vor die Zeile 7.00 und hinter die Zeile 8.00 kopiert werden. Dazu werden die Zeile 3.00 mit C, die Zeile 7.00 mit B und die Zeile 8.00 mit A markiert.

```
   1.00 C    MARKIEREN ZUM KOPIEREN.....................................
   2.00 ==========================.....................................
   3.00  .....................................................................
   4.00 MIT C MARKIERTE ZEILEN WERDEN AN EINEN ANGEGEBENEN ZIELORT (A, B, O)....
   5.00 KOPIERT. DIE ZEILENNUMMERN (MAXIMAL 255) WERDEN IN EINEM.................
   6.00 KOPIERPUFFER ZWISCHENGESPEICHERT. .....................................
   6.10  .....................................................................
   7.00 SOBALD DER ZIELORT ANGEGEBEN IST, WIRD DIE KOPIERANWEISUNG AUSGEFUEHRT,.
   8.00 DER INHALT DES KOPIERPUFFERS GELOESCHT. ...............................
   9.00 .....................................................................
  10.00 .....................................................................
  11.00 .....................................................................
  12.00 .....................................................................
  13.00 .....................................................................
  14.00 .....................................................................
  15.00 .....................................................................
  16.00 .....................................................................
  17.00 .....................................................................
  18.00 .....................................................................
  19.00 .....................................................................
  20.00 .....................................................................
  21.00 .....................................................................
%  EDT5360 NO COPY. BUFFER EMPTY
.............................................................. 0001.00:001(0)
```

Die Zeile 3.00 wurde vor die Zeile 7.00 kopiert, nicht jedoch hinter die Zeile 8.00. Stattdessen gibt der EDT eine Fehlermeldung aus.

Mit C markierte Zeilen können nur zu einem Zielort kopiert werden, da nach dem ersten Kopieren der Kopierpuffer gelöscht wird. Die zweite Zielortangabe verursachte die Fehlermeldung. Um Zeilen mehrmals kopieren zu können, sind diese Zeilen mit R zu markieren.

D Löschen von Sätzen

Die mit D markierten Sätze werden gelöscht.

Kurzanweisung	Taste
D	DÜE oder F2

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
d  2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................
```

Die Zeile 2.00 soll gelöscht werden. Dazu wird sie in der Markierungsspalte mit D markiert.

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
  6.00 ...........................................................................
```

E Einfügen von Zeichen

Mit E können Zeichen in eine Zeile eingefügt werden. Die Zeile wird auf überschreibbar gestellt.

Kurzanweisung	Taste
E	DUE oder F2

Enthält die mit E markierte Zeile nicht mindestens 20 NIL- oder Füllzeichen am Zeilenende, stellt der EDT 20 NIL-Zeichen am Zeilenende zur Verfügung. Die Zeichen der Zeile, die durch die 20 NIL-Zeichen verschoben wurden, sind zwar im Datenfenster nicht mehr sichtbar, bleiben jedoch erhalten.

Der Benutzer kann nun an beliebiger Stelle der Zeile bis zu 20 Zeichen einfügen (EFG). Werden weniger als 20 Zeichen eingefügt, wird der verschobene Satzrest ins Datenfenster nachgerückt.

Im EDIT LONG-Modus wird durch die Markierung mit E zur üblichen Darstellung des Datensatzes zusätzlich eine Zeile mit 80 NIL-Zeichen angeboten.

Beispiel

```
   1.00 E      EINFUEGEN VON ZEICHEN.....................................
   2.00  ....................................................................
   3.00  ....................................................................
   4.00 MIT E KOENNEN ZEICHEN IN EINE ZEILE EINGEFUEGT WERDEN. DIE ZEILE.....
   5.00 WIRD AUF UEBERSCHREIBBAR GESTELLT. ..................................
   6.00  ....................................................................
x  7.00 ENTHAELT DIE MIT E MARKIERTE ZEILE NICHT MINDESTENS 20 NILZEICHEN.....
e  8.00 AM ZEILENENDE, SO STELLT DER EDT 20 NIL-ZEICHEN ZUR VERFUEGUNG. .....
   9.00 DIE ZEICHEN DER ZEILE, DIE DURCH DIE 20 NIL-ZEICHEN VERSCHOBEN WURDEN,..
  10.00 SIND ZWAR IM DATENFENSTER NICHT MEHR SICHTBAR, BLEIBEN JEDOCH ERHALTEN..
  11.00 ....................................................................
```

Zeile 8.00 wird mit E zum Einfügen und Zeile 7.00 mit X zum Ändern markiert.

```
   1.00 E      EINFUEGEN VON ZEICHEN .....................................
   2.00  ....................................................................
   3.00  ....................................................................
   4.00 MIT E KOENNEN ZEICHEN IN EINE ZEILE EINGEFUEGT WERDEN. DIE ZEILE.....
   5.00 WIRD AUF UEBERSCHREIBBAR GESTELLT. ..................................
   6.00  ....................................................................
   7.00 ENTHAELT DIE MIT E MARKIERTE ZEILE NICHT MINDESTENS 20 NILZEICHEN.....
   8.00 AM ZEILENENDE, SO STELLT DER EDT 20 NIL-ZEICHEN ZUR ..................
   9.00 DIE ZEICHEN DER ZEILE, DIE DURCH DIE 20 NIL-ZEICHEN VERSCHOBEN WURDEN,..
  10.00 SIND ZWAR IM DATENFENSTER NICHT MEHR SICHTBAR, BLEIBEN JEDOCH ERHALTEN..
  11.00 ....................................................................
```

Da am Ende von Zeile 8.00 kein Platz für 20 Zeichen vorhanden war, schiebt der EDT den Zeilenrest aus dem Datenfenster und stellt 20 NIL-Zeichen zur Verfügung.

```
  1.00 E      EINFUEGEN VON ZEICHEN.....................................
  2.00  .....................................................................
  3.00  .....................................................................
  4.00 MIT E KOENNEN ZEICHEN IN EINE ZEILE EINGEFUEGT WERDEN. DIE ZEILE.....
  5.00 WIRD AUF UEBERSCHREIBBAR GESTELLT....................................
  6.00  .....................................................................
  7.00 ENTHAELT DIE MIT E MARKIERTE ZEILE NICHT MINDESTENS 20 NIL- oder.....
  8.00 Fuellzeichen AM ZEILENENDE, SO STELLT DER EDT 20 NIL-ZEICHEN ZUR.....
  9.00 DIE ZEICHEN DER ZEILE, DIE DURCH DIE 20 NIL-ZEICHEN VERSCHOBEN WURDEN...
 10.00 SIND ZWAR IM DATENFENSTER NICHT MEHR SICHTBAR, BLEIBEN JEDOCH ERHALTEN..
 11.00 .....................................................................
```

Zeile 7.00 wurde geändert und in Zeile 8.00 mit EFG neuer Text eingefügt.

```
  1.00 E      EINFUEGEN VON ZEICHEN.....................................
  2.00  .....................................................................
  3.00  .....................................................................
  4.00 MIT E KOENNEN ZEICHEN IN EINE ZEILE EINGEFUEGT WERDEN. DIE ZEILE.....
  5.00 WIRD AUF UEBERSCHREIBBAR GESTELLT....................................
  6.00  .....................................................................
  7.00 ENTHAELT DIE MIT E MARKIERTE ZEILE NICHT MINDESTENS 20 NIL- ODER.....
  8.00 FUELLZEICHEN AM ZEILENENDE, SO STELLT DER EDT 20 NIL-ZEICHEN ZUR VERFUEG
  9.00 DIE ZEICHEN DER ZEILE, DIE DURCH DIE 20 NIL-ZEICHEN VERSCHOBEN WURDEN,..
 10.00 SIND ZWAR IM DATENFENSTER NICHT MEHR SICHTBAR, BLEIBEN JEDOCH ERHALTEN..
 11.00 .....................................................................
```

Da in Zeile 8.00 weniger als 20 neue Zeichen eingegeben wurden, rückt der EDT den verschobenen Zeilenrest ins Datenfenster nach.

J Zusammenketten zweier Sätze

Mit J muß ein Satz markiert werden, der an den davorliegenden Satz angefügt werden soll. Der markierte Satz wird anschließend gelöscht.

Kurzanweisung	Taste
J	DUE oder F2

Übersteigt die Summe der Satzlängen der verketteten Sätze die maximale Satzlänge von 256 Zeichen, wird auf maximale Satzlänge abgeschnitten. Der mit J markierte Satz wird nicht gelöscht. Es wird folgende Meldung ausgegeben: % EDT2267 LINE TRUNCATED AFTER 256 CHARACTERS. Ein Fehlerschalter wird nicht gesetzt.

Wird der erste Satz in der Datei mit J markiert, bleibt dieser unverändert in der Datei erhalten.

Auftrennen eines Datensatzes (siehe Abschnitt „Anweisung im Datenfenster - Auftrennen eines Datensatzes“ auf Seite 107).

Beispiel

```
   1.00 MIT J MUSS EIN SATZ MARKIERT.....................................
j  2.00  WERDEN, DER AN DEN DAVORLIEGENDEN.................................
   3.00  SATZ ANGEFUEGT WERDEN.........................................
j  4.00  SOLL. DER MARKIERTE.............................................
j  5.00 SATZ WIRD.....................................................
   6.00 ANSCHLIESSEND.....................................................
j  7.00  GELOESCHT. ....................................................
   8.00 ..........................................................
```

```
   1.00 MIT J MUSS EIN SATZ MARKIERT WERDEN, DER AN DEN DAVORLIEGENDEN.....
   3.00 SATZ ANGEFUEGT WERDEN SOLL. DER MARKIERTE SATZ WIRD................
   6.00 ANSCHLIESSEND GELOESCHT. ..........................................
   7.00 ..........................................................
```

K Kopieren einer Zeile in die Anweisungszeile

Der Inhalt einer mit K markierten Zeile wird in die Anweisungszeile übernommen (max. 65 Zeichen), wobei ein etwa vorhandener Inhalt in der Anweisungszeile überschrieben wird.

Kurzanweisung	Taste
K	DUE oder F2

Pro Arbeitsfenster ist nur eine K-Markierung erlaubt. Werden mehrere Zeilen in einem Arbeitsfenster mit K markiert, erfolgt keine Eingabebearbeitung. Die überflüssigen K-Markierungen werden mit einem Fragezeichen ? überschrieben.

Mindestens ein Zeichen der mit K kopierten Zeile muß in der Anweisungszeile überschrieben, geändert bzw. hinzugefügt werden, wenn sie als Anweisung abgeschickt werden soll.

Beispiel

```
   1.00 DER INHALT EINER MIT K MARKIERTEN ZEILE WIRD IN DIE ANWEISUNGSZEILE.....
   2.00 UEBERNOMMEN (MAX. 65 ZEICHEN), WOBEI EIN ETWA VORHANDENER..............
   3.00 INHALT IN DER ANWEISUNGSZEILE UEBERSCHRIEBEN WIRD. .....................
   4.00  .......................................................................
k  5.00 SCALE ON..................................................................
```

```
   1.00 DER INHALT EINER MIT K MARKIERTEN ZEILE WIRD IN DIE ANWEISUNGSZEILE.....
   2.00 UEBERNOMMEN (MAX. 65 ZEICHEN), WOBEI EIN ETWA VORHANDENER..............
   3.00 INHALT IN DER ANWEISUNGSZEILE UEBERSCHRIEBEN WIRD. .....................
   4.00  .......................................................................
   5.00 SCALE ON..................................................................
   6.00 .......................................................................









sCALE ON ..................................................0001.00:001(0)
```

```
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 DER INHALT EINER MIT K MARKIERTEN ZEILE WIRD IN DIE ANWEISUNGSZEILE.....
   2.00 UEBERNOMMEN (MAX. 65 ZEICHEN), WOBEI EIN ETWA VORHANDENER..............
   3.00 INHALT IN DER ANWEISUNGSZEILE UEBERSCHRIEBEN WIRD. .....................
   4.00  .......................................................................
   5.00 SCALE ON..................................................................
   6.00 .......................................................................
```

M Kopieren und Löschen markierter Zeilen

M kennzeichnet Zeilen, die an einen Zielort (A, B, O) übertragen werden. Anschließend werden die mit M markierten Zeilen gelöscht. Die Zeilennummern (maximal 255) werden in einem Kopierpuffer zwischengespeichert. Sobald der Zielort angegeben ist, wird die Kopieranweisung ausgeführt und die mit M markierten Zeilen sowie der Inhalt des Kopierpuffers gelöscht.

Kurzanweisung	Taste
M	DUE oder F2

Die Zeilennummern der kopierten Zeilen vergibt der EDT nach drei Prinzipien (siehe @COPY, Format 2):

1. Standardnumerierung mit Schrittweite 1.0000
2. Numerierung mit festgelegter Schrittweite (@PAR INCREMENT).
3. Automatische Numerierung und Umnumerierung (nur bei @PAR RENUMBER=ON). Der EDT numeriert automatisch um, wenn die Schrittweite zu groß ist, um alle kopierten Zeilen einzufügen. Näheres siehe @COPY, Format 2 „Berechnung der Zeilennummern“.

In einem Dialogschritt werden C, M und R nicht gleichzeitig durchgeführt. Die gemischte Angabe von C, M und R in der Markierungsspalte eines Arbeitsfensters wird durch eine Semantikprüfung abgewiesen. Anstelle der fehlerhaften Kurzanweisung wird ein ’?’ ausgegeben. Die Schreibmarke wird zum Korrigieren auf die fehlerhafte Kurzanweisung positioniert.

A, B, O und * löschen den Inhalt des Kopierpuffers.

Bei einem gesplitteten Bildschirm können in einem Dialogschritt Zeilen vom ersten Arbeitsfenster ins zweite Arbeitsfenster kopiert werden.

Beim Kopieren vom zweiten Arbeitsfenster ins erste Arbeitsfenster sind, bedingt durch die Abarbeitungsreihenfolge, zwei Dialogschritte notwendig.

Der Kopierpuffer kann auch durch Markieren von Zeilen in verschiedenen Arbeitsdateien aufgebaut werden. Die Zielmarkierung kann in jeder Arbeitsdatei angegeben werden. Ein mit M angelegter Kopierpuffer wird durch nachfolgendes C oder R gelöscht.

Der Kopierpuffer enthält Arbeitsdatei- und Zeilennummer der mit M markierten Sätze. Die Zeilennummern dürfen daher zwischen M-Markierung und Ausführung des Übertragungsvorgangs mit A, B oder O nicht verändert werden.

Wird eine mit M markierte Zeile anschließend mit O markiert, wird diese Zeile gelöscht.

Beispiel

```
   1.00 M KENNZEICHNET ZEILEN, DIE AN EINEN ZIELORT (A, B, O).................
   2.00 UEBERTRAGEN WERDEN. ...................................................
m  3.00  ..........................................................................
m  4.00 AAA........................................................................
a  5.00    BBB.....................................................................
   6.00       CCC..................................................................
   7.00 ..........................................................................
```

```
   1.00 M KENNZEICHNET ZEILEN, DIE AN EINEN ZIELORT (A, B, O).................
   2.00 UEBERTRAGEN WERDEN. ...................................................
   5.00    BBB.....................................................................
   5.10  ..........................................................................
   5.20 AAA........................................................................
   6.00       CCC..................................................................
   7.00 ..........................................................................
```

n/I Einfügen von Zeilen

Mit Hilfe von n/I können Sätze in eine Arbeitsdatei eingefügt werden.

Kurzanweisung	Taste
n I	DÜE oder F2

n Anzahl der einzufügenden Zeilen.
$1 \leq n \leq 9$
Die Zeilen werden vor der mit n markierten Zeile eingefügt.

I Mit der Markierung I (für Insert) wird eine Dauereinfügefunktion aktiviert, bei der - sofern das Arbeitsfenster groß genug ist - ein 9 Zeilen umfassender Einfügebereich aufgebaut wird.

Nachdem die bereitgestellten Leerzeilen beschrieben und das Arbeitsfenster mit DÜE übertragen wurde, wird erneut ein Einfügebereich angeboten. Dies geschieht solange, bis

- in der letzten ausgegebenen Einfügezeile keine Eingabe erfolgt oder
- S eingegeben wird oder
- der Einfügebereich durch Positionieren oder Arbeitsdateiwechsel nicht mehr am Datenfenster dargestellt werden kann.

Es ist nur eine I-Markierung pro Arbeitsfenster erlaubt. Während einer Dauereinfügefunktion wird durch ein weiteres I der erste Einfügebereich geschlossen und eine neue Einfügestelle definiert.

Dem gewählten n entsprechend werden vor der markierten Zeile Leerzeilen angeboten. Für diese Leerzeilen werden bereits Zeilennummern gebildet. Diese werden bei eingeschalteter Zeilennummernanzeige mit angezeigt. Die Zeilennummern werden in Abhängigkeit von der Zeilennummer der Zeile vor dem Einfügebereich und der eingestellten Schrittweite gebildet. Die dritte und vierte Stelle der Zeilennummer der Zeile vor dem Einfügebereich wird dabei nicht berücksichtigt. Bei einer ursprünglich in Einerschritten numerierten Arbeitsdatei können somit an einer Stelle 99 neue Zeilen eingefügt werden, ohne die ursprünglichen Zeilennummern zu ändern.

Ist die Differenz zwischen zwei Zeilennummern nicht groß genug, um die angegebene Zahl von Zeilen mit Schrittweite 0.01 einzufügen, werden die nachfolgenden Zeilen umnumeriert. Dabei werden die Zeilen mit Schrittweite 0.01 weiternumeriert, bis die aufsteigende Reihenfolge wiederhergestellt ist. Diese Neunumerierung wird beibehalten, auch wenn nicht alle angebotenen Leerzeilen mit Text belegt wurden.

Wird in einer angebotenen Leerzeile kein Text eingetragen, wird auch kein Satz in der Arbeitsdatei angelegt.

Die I-Markierung kann auch mit F2 abgeschickt werden, dabei werden die Zeilen des Arbeitsfensters auf überschreibbar gestellt.

Beispiel

```
   1.00 n/I     EINFUEGEN VON ZEILEN.....................................
   2.00  ..................................................................
   3.00  ..................................................................
   4.00 Mit n/I koennen Saetze in eine Arbeitsdatei eingefuegt werden. .....
   5.00  ..................................................................
   6.00 111................................................................
3  7.00    222.............................................................
   8.00       333..........................................................
```

Vor Zeile 7.00 sollen 3 Zeilen eingefügt werden. Dazu wird die Zeile 7.00 mit „3“ markiert.

```
   1.00 n/I     EINFUEGEN VON ZEILEN.....................................
   2.00  ..................................................................
   3.00  ..................................................................
   4.00 Mit n/I koennen Saetze in eine Arbeitsdatei eingefuegt werden. .....
   5.00  ..................................................................
   6.00 111................................................................
   6.10 ...................................................................
   6.20 ...................................................................
   6.30 ...................................................................
   7.00    222.............................................................
i  8.00       333..........................................................
   9.00 ...................................................................
```

```
   1.00 n/I     EINFUEGEN VON ZEILEN.....................................
   2.00  ..................................................................
   3.00  ..................................................................
   4.00 Mit n/I koennen Saetze in eine Arbeitsdatei eingefuegt werden. .....
   5.00  ..................................................................
   6.00 111................................................................
   6.10 112................................................................
   6.20 113................................................................
   6.30 114................................................................
   7.00    222.............................................................
i  8.00       333..........................................................
   9.00 ...................................................................
```

Vor Zeile 8.00 sollen mehr als 9 Zeilen eingefügt werden. Dazu wird Zeile 8.00 mit ’I’ (Daueinfügefunktion) markiert.

```
  1.00 n/I     EINFUEGEN VON ZEILEN.....................................
  2.00  ...................................................................
  3.00  ...................................................................
  4.00 Mit n/I koennen Saetze in eine Arbeitsdatei eingefuegt werden. .....
  5.00  ...................................................................
  6.00 111.................................................................
  6.10 112.................................................................
  6.20 113.................................................................
  6.30 114.................................................................
  7.00    222..............................................................
  7.10       1.............................................................
  7.20       2.............................................................
  7.30       3.............................................................
  7.40       4.............................................................
  7.50       5.............................................................
  7.60       6.............................................................
  7.70       7.............................................................
  7.80       8.............................................................
  7.90       9.............................................................
  8.00       333...........................................................
  9.00 .....................................................................
 10.00 .....................................................................
 11.00 .....................................................................
.....................................................................0001.00:001(0)
```

Alle 9 Zeilen des Einfügebereichs wurden mit Daten gefüllt.

```
  7.90       9.............................................................
  7.91 .....................................................................
  7.92 .....................................................................
  7.93 .....................................................................
  7.94 .....................................................................
  7.95 .....................................................................
  7.96 .....................................................................
  7.97 .....................................................................
  7.98 .....................................................................
  7.99 .....................................................................
  8.00       333...........................................................
  9.00 .....................................................................
```

Da der Einfügebereich gefüllt war, wird ein weiterer 9 Zeilen umfassender Einfügebereich mit Zeilennummernabstand 0.01 angeboten.

R Markieren zum Kopieren (ohne Löschen des Kopierpuffers)

Mit R werden Zeilen zum Kopieren an einen angegebenen Zielort (A, B, O) bereitgestellt. Beim Kopieren wird der Kopierpuffer nicht gelöscht.

Die Zeilennummern (maximal 255) werden in einem Kopierpuffer zwischengespeichert. Der Inhalt des Kopierpuffers bleibt so lange erhalten, bis eine C-, M- oder *-Markierung angegeben wird. Dadurch ist es möglich, die mit R markierten Zeilen an mehrere Zielorte zu kopieren.

Kurzanweisung	Taste
R	DUE oder F2

Die Zeilennummern der kopierten Zeilen vergibt der EDT nach drei Prinzipien (siehe @COPY, Format 2):

1. Standardnumerierung mit der Schrittweite 1.0000
2. Numerierung mit festgelegter Schrittweite (@PAR INCREMENT)
3. Automatische Numerierung und Umnumerierung (nur bei @PAR RENUMBER=ON). Der EDT numeriert automatisch um, wenn die Schrittweite zu groß ist, um alle kopierten Zeilen einzufügen. Näheres siehe @COPY, Format 2 „Berechnung der Zeilennummern“.

In einem Dialogschritt werden C, M und R nicht gleichzeitig durchgeführt. Die gemischte Angabe von C, M und R in der Markierungsspalte eines Arbeitsfensters wird durch eine Semantikprüfung abgewiesen. Anstelle der fehlerhaften Kurzanweisung wird ein ’?’ ausgegeben. Die Schreibmarke wird zum Korrigieren auf die fehlerhafte Kurzanweisung positioniert.

Bei einem gesplitteten Bildschirm können in einem Dialogschritt Zeilen vom ersten Arbeitsfenster ins zweite Arbeitsfenster kopiert werden.

Beim Kopieren vom zweite Arbeitsfenster ins erste Arbeitsfenster sind, bedingt durch die Abarbeitungsreihenfolge, zwei Dialogschritte notwendig.

Der Kopierpuffer kann auch durch Markieren von Zeilen in verschiedenen Arbeitsdateien aufgebaut werden. Die Zielmarkierung kann in jeder Arbeitsdatei angegeben werden. Ein mit R angelegter Kopierpuffer wird durch nachfolgendes C oder M gelöscht.

Der Kopierpuffer enthält Arbeitsdatei- und Zeilennummer der mit R markierten Sätze. Die Zeilennummern dürfen daher zwischen R-Markierung und der Ausführung des Kopiervorgangs durch Eingabe der Kurzanweisungen A, B oder O nicht verändert werden.

Beispiel

```
   1.00 R    MARKIEREN ZUM KOPIEREN (OHNE LOESCHEN DES KOPIERPUFFERS)...........
r  2.00  ..................................................................
b  3.00 Mit R werden Zeilen an einen angegebenen Zielort (A, B, O) kopiert. ....
b  4.00 Die Zeilennummern werden in einem Kopierpuffer (maximal.................
   5.00 255) zwischengespeichert. Der Inhalt des Kopierpuffers bleibt so lange..
   6.00 erhalten, bis eine C-, M- oder *-Markierung angegeben wird. .............
   7.00 Dadurch ist es moeglich, die mit R markierten Zeilen an mehrere Zielorte
   8.00 zu kopieren. ..........................................................
   9.00 ..................................................................
```

```
   1.00 R    MARKIEREN ZUM KOPIEREN (OHNE LOESCHEN DES KOPIERPUFFERS)...........
   2.00  ..................................................................
   2.10  ..................................................................
   3.00 Mit R werden Zeilen an einen angegebenen Zielort (A, B, O) kopiert. ....
   3.10  ..................................................................
   4.00 Die Zeilennummern werden in einem Kopierpuffer (maximal.................
   5.00 255) zwischengespeichert. Der Inhalt des Kopierpuffers bleibt so lange..
   6.00 erhalten, bis eine C-, M- oder *-Markierung angegeben wird. .............
   7.00 Dadurch ist es moeglich, die mit R markierten Zeilen an mehrere Zielorte
   8.00 zu kopieren. ..........................................................
   9.00 ..................................................................
```

S Positionieren des Arbeitsfensters (horizontal und vertikal)

Mit S wird eine Zeile auf überschreibbar gestellt und in die zweite Zeile des Arbeitsfensters positioniert. In der ersten Zeile des Arbeitsfensters wird ein Spaltenzähler ausgegeben.

Schreibt der Benutzer in der markierten Zeile bis vor die gewünschte Spaltenposition Leerzeichen, dann positioniert der EDT

– die erste mit S markierte Zeile in die erste Zeile des Arbeitsfensters,

– das Arbeitsfenster auf die erste Spalte ungleich Leerzeichen in der mit S markierten Zeile.

Wird die Zeile nicht verändert, erfolgt keine Spaltenpositionierung. Leerzeichen und evtl. andere eingegebene Zeichen ändern den ursprünglichen Zeileninhalt nicht.
Soll innerhalb eines Leerzeichenbereiches positioniert werden, muß an die entsprechende Stelle ein Zeichen ungleich dem Leerzeichen eingegeben werden.

Kurzanweisung	Taste
S	DUE

Überschreibt der Benutzer den gesamten Text einer mit S markierten Zeile mit Leerzeichen, dann positioniert der EDT das Arbeitsfenster auf Spalte 73 bzw. 81 (entsprechend INDEX=ON/OFF).

S ist nur mit K und D kombinierbar. Dabei müssen diese Markierungen im Datenfenster vor einer S-Markierung stehen.

Stehen andere Kurzanweisungen (X, E) in der Markierungsspalte unterhalb von S, werden sie mit ? überschrieben und abgewiesen. Stehen andere Kurzanweisungen oberhalb von S, wird S mit ? überschrieben und abgewiesen.

Mit der Anweisung << wird das Arbeitsfenster auf Spalte 1 zurückpositioniert (siehe Abschnitt „> / < Horizontales Positionieren in der Arbeitsdatei“ auf Seite 110ff.).

Beispiel

```
   1.00 LFD.NR   ART.NR. ART.NAME       BESTAND   BESTELLT.....................
   2.00 1        0024    SEIFE          3000      150..........................
   3.00 2        0015    DEODORANT      2500      600..........................
s  4.00 3        0048    PARFUEM        400       60...........................
   5.00 4        0003    CREME          987       555..........................
   6.00 5        0091    RASIERSCHAUM   350       30...........................
   7.00 6        0090    RASIERWASSER   340       30...........................
   8.00 7        0092    RASIERPINSEL   200       30...........................
   9.00 ...........................................................................
```

Das Arbeitsfenster soll auf Zeile 4.00 positioniert und die Zeile 4.00 überschreibbar gestellt werden.

```
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   4.00                 PARFUEM        400       60...........................
   5.00 4        0003    CREME          987       555..........................
   6.00 5        0091    RASIERSCHAUM   350       30...........................
   7.00 6        0090    RASIERWASSER   340       30...........................
   8.00 7        0092    RASIERPINSEL   200       30...........................
   9.00 ...........................................................................
```

Das Arbeitsfenster soll auf Spalte 18 positioniert werden. Dazu wird die Zeile 4.00 bis zur gewünschten Spaltenposition mit Leerzeichen überschrieben.

```
   4.00 PARFUEM        400       60.....................................
   5.00 CREME          987       555....................................
   6.00 RASIERSCHAUM   350       30.....................................
   7.00 RASIERWASSER   340       30.....................................
   8.00 RASIERPINSEL   200       30.....................................
   9.00 ...........................................................................
```

Der EDT positionierte das Arbeitsfenster auf Spalte 18.

Mit der Anweisung << in der Anweisungszeile wird auf die Spalte 1 zurückpositioniert (siehe auch Abschnitt „> / < Horizontales Positionieren in der Arbeitsdatei“ auf Seite 110).

T Syntaxtest durch SDF

Eine mit T gekennzeichnete Zeile wird gemeinsam mit ihren Folgezeilen an SDF zur Kontrolle auf Kommando- oder Anweisungssyntax übergeben.

Abhängig vom GUIDANCE-Modus der SDF-Einstellung (GUIDANCE=MIN|MED|MAX) wird bei einer fehlerhaften SDF-Syntax in den geführten Korrektur-Dialog von SDF übergegangen.

Bricht der Anwender den Korrektur-Dialog ab oder ist ein solcher nicht möglich, wird die Zeile überschreibbar an der obersten Fensterposition angezeigt und wie bei der Anweisung @SDFTEST eine Fehlermeldung ausgegeben.

Ist die SDF-Syntax korrekt, bzw. wurde sie korrigiert, wird der Text in die Arbeitsdatei aufgenommen.
Das Format wird durch die SDF-Einstellung LOGGING bestimmt (siehe Beschreibung des Kommandos MODIFY-SDF-OPTIONS).

Es gelten die aktuellen SDF-Einstellungen, die mit MODIFY-SDF-OPTIONS verändert werden können.

Kurzanweisung	Taste
T	DUE oder F2

Der EDT unterscheidet 3 Arten von Zeileninhalten:

1. Zeilen, die mit einem (nur einem) / in Spalte 1 beginnen.
   Sie werden gemäß der SDF-Syntaxdatei-Hierarchie auf Kommando-Syntax geprüft. Die Zulässigkeit bezüglich Privilegien oder Systemumgebung (z.B. Stapelprozess oder Prozedur) ist durch den aktuellen Benutzer und die aktuelle Umgebung bestimmt.
2. Zeilen, die mit // beginnen.
   Diese werden an SDF zur Anweisungsüberprüfung übergeben.
   Der Programmname wird durch die Anweisung @PAR SDF-PROGRAM voreingestellt, bzw. ist durch eine vorangegangene Anweisung @SDFTEST PROGRAM=name bekannt.
   Der Programmname muß in einer aktuellen SDF-Syntaxdatei bekannt sein.
3. reine Datenzeilen
   Die Angabe von t wird ignoriert.

Fortsetzungszeilen bei der Eingabe

Ist das letzte Zeichen einer Zeile, die mit / oder // beginnt ein Fortsetzungszeichen (d.h. -) und beginnt die nächste ebenfalls mit / oder // , so ist diese eine Fortsetzungszeile und wird als Zeichenkette verkettet an SDF mitübergeben. Eine eventuelle t-Markierung wird ignoriert.

Ausgabe des geprüften Textes und Fortsetzungszeilen bei der Ausgabe

Der Text wird beginnend ab der markierten Zeilennummer - gegebenenfalls in Teilstücken - in die Datei geschrieben.

Das Fortsetzungszeichen wird an die 72. Spalte gesetzt. Wenn nötig, werden die nachfolgenden Zeilen umnumeriert.

Die Zeilennummernvergabe erfolgt wie bei @COPY Format 2.

i

- Die T-Markierung darf nicht gleichzeitig mit Positionier-Markierungen (+, - oder S) und Einfüge-Markierungen (1-9 oder I) verwendet werden.
- Kennwörter und andere Operanden, die mit OUTPUT=SECRET-PROMPT definiert wurden, werden bei GUIDANCE-Einstellung MIN, MED oder MAX durch P ersetzt.
- Fehlerhafte Operanden bei ISP-Kommandos werden von SDF nicht erkannt.
- Es sind maximal 255 Fortsetzungszeilen (sowohl bei der Eingfabe, als auch bei der Ausgabe) möglich.
- Eine &-Ersetzung wird nur in Operandenwerten akzeptiert, nicht jedoch in Kommando-, Anweisungs- und Operandennamen oder Teilen von diesen.

Beispiel 1

```
/MODIFY-SDF-OPTIONS GUIDANCE=EXPERT,LOGGING=INPUT-FORM
```

Es wird für SDF die EXPERT-Form des ungeführten Dialogs eingestellt.

```
t  1.00 /SET-JOB-STEP.....................................................
t  2.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE2, -.........................
t  3.00 /                      NEW-NAME=FILE3, -..........................
t  4.00 /                      PROT=*PARAMETERS(UCCESS=*READ).............
t  5.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE1, -.........................
t  6.00 /                      NEW-NAME=FILE2, -..........................
t  7.00 /                      PROT=*PARAMETERS(ACCESS=*READ).............
t  8.00 ..................................................................
```

Die Zeilen 1-8 sollen durch SDF auf richtige SDF-Syntax geprüft werden.

```
   1.00 /SET-JOB-STEP.....................................................
   2.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE2, -.........................
   3.00 /                      NEW-NAME=FILE3, -..........................
   4.00 /                      PROT=*PARAMETERS(UCCESS=*READ).............
   5.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE1, -.........................
   6.00 /                      NEW-NAME=FILE2, -..........................
   7.00 /                      PROT=*PARAMETERS(ACCESS=*READ).............
   8.00 ..................................................................




% EDT4310 SDF: SYNTAX ERROR IN LINE 0002.000 .....................0002.00:001(0)
```

Es wird auf die erste Zeile des fehlerhaften Kommandos positioniert und die Zeilen des Kommandos werden überschreibar ausgegeben.

Beispiel 2

```
/MODIFY-SDF-OPTIONS GUIDANCE=MINIMUM,LOGGING=INPUT-FORM
```

Es wird für SDF der geführte Dialog mit minimaler Hilfe eingestellt.

```
t  1.00 /SET-JOB-STEP.....................................................
t  2.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE2, -.........................
t  3.00 /                      NEW-NAME=FILE3, -..........................
t  4.00 /                      PROT=*PARAMETERS(UCCESS=*READ).............
t  5.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE1, -.........................
t  6.00 /                      NEW-NAME=FILE2, -..........................
t  7.00 /                      PROT=*PARAMETERS(ACCESS=*READ).............
t  8.00 ..................................................................
```

Die Zeilen 1-8 sollen durch SDF auf richtige SDF-Syntax geprüft werden.

```
SITUATION: ERROR IN PROG/S-PROC          COMMAND: MODIFY-FILE-ATTRIBUTES


------------------------------------------------------------------------------
FILE-NAME            = FILE3
NEW-NAME             = FILE4
SUPPORT              = *UNCHANGED
PROTECTION           = *PARAMETERS(ACCESS=*UNCHANGED,USER-ACCESS=*UNCHANGED,BASI
                       C-ACL=*UNCHANGED,GUARDS=*UNCHANGED,WRITE-PASSWORD=*UNCHAN
                       GED,READ-PASSWORD=*UNCHANGED,EXEC-PASSWORD=*UNCHANGED,DES
                       TROY-BY-DELETE=*UNCHANGED,AUDIT=*UNCHANGED,SPACE-RELEASE-
                       LOCK=*UNCHANGED,RETENTION-PERIOD=*UNCHANGED)
SAVE                 = *UNCHANGED
MIGRATE              = *UNCHANGED
CODED-CHARACTER-SET  = *UNCHANGED


------------------------------------------------------------------------------
NEXT = *CONTINUE
       *EXECUTE“F3“ OR + OR *EXIT“K1“ OR *EXIT-ALL“F1“



ERROR:  CMD0185 OPERAND NAME ‚UCCESS‘ COULD NOT BE IDENTIFIED
```

Es wird in den geführten Fehler-Dialog von SDF übergegangen.

```
SITUATION: ERROR IN PROG/S-PROC          COMMAND: MODIFY-FILE-ATTRIBUTES


------------------------------------------------------------------------------
FILE-NAME             = FILE3
NEW-NAME              = FILE4
SUPPORT               = *UNCHANGED
PROTECTION            = *PARAMETERS(ACCESS=R)




SAVE                  = *UNCHANGED
MIGRATE               = *UNCHANGED
CODED-CHARACTER-SET   = *UNCHANGED


------------------------------------------------------------------------------
NEXT = *CONTINUE
       *EXECUTE“F3“ OR + OR *EXIT“K1“ OR *EXIT-ALL“F1“



ERROR:  CMD0185 OPERAND NAME ‚UCCESS‘ COULD NOT BE IDENTIFIED
```

Der Fehler wird korrigiert.

```
   1.00 /SET-JOB-STEP.....................................................
   2.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE3,NEW-NAME=FILE4,PROTECTION=     -
   3.00 /*PARAMETERS(ACCESS=*READ)........................................
   5.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE1, -.........................
   6.00 /                       NEW-NAME=FILE2, -.........................
   7.00 /                       PROT=*PARAMETERS(ACCESS=*READ)............
   8.00 ..................................................................
```

Die Zeilen 2-4 werden durch die Zeilen 2-3 ersetzt.

+ / – Positionieren des Arbeitsfensters

Mit + wird eine markierte Zeile zum ersten Satz im Arbeitsfenster.

Mit – wird die markierte Zeile zum letzten Satz im Arbeitsfenster.

Kurzanweisung	Taste
+ –	DUE oder F2

+ bzw. – darf innerhalb der Markierungsspalte eines Arbeitsfensters

– nur einmal vorkommen
– nicht zusammen mit + bzw. – oder S angegeben werden
– nicht nach X, E, n oder I angegeben werden

Die Spaltenposition wird durch diese Kurzanweisungen nicht verändert.

Das Zeichen + ist nicht wirksam, wenn nur noch die letzte Datenzeile am Bildschirm angezeigt wird.

Das Zeichen – ist im ersten Bildschirm (am Anfang der Daten) einer Arbeitsdatei nicht wirksam.

+ / – Positionieren des Arbeitsfensters nach der Strukturtiefe

Mit diesen Anweisungen kann man zum nächsten bzw. vorherigen Satz mit derselben Strukturtiefe positionieren.

Strukturtiefe ist der Abstand des ersten Zeichens ungleich eines Leerzeichens vom Satzbeginn.

Ist ein Struktursymbol definiert ungleich Leerzeichen (siehe @PAR STRUCTURE), werden nur die Sätze ausgewertet, die mindestens dieses Struktursymbol enthalten. Wird für das Struktursymbol ein Leerzeichen angegeben, so werden alle Sätze ausgewertet. Der Standardwert des Struktursymbols ist @.

Kurzanweisung	Taste
+ –	F1

Wird kein Satz mit derselben Strukturtiefe gefunden, bleibt die Position unverändert.

Enthält der markierte Satz kein Struktursymbol, wird die Kurzanweisung mit folgender Meldung abgewiesen: % EDT5354 STRUCTURE SYMBOL 'symbol' NOT FOUND

Beispiel

```
   1.00 STRUKTURTIEFE 1.....................................................
   2.00   STRUKTURTIEFE  2..................................................
   3.00     STRUKTURTIEFE  3................................................
   4.00  ...................................................................
   5.00   JETZT NOCH MAL STRUKTURTIEFE 2....................................
   6.00 UND WIEDER STRUKTURTIEFE 1..........................................
   7.00 ....................................................................




par structure=' '.......................................0001.00:001(0)
```

Ein Leerzeichen wird als Struktursymbol definiert.

```
   1.00 STRUKTURTIEFE 1.....................................................
+  2.00   STRUKTURTIEFE  2.....................................................
   3.00     STRUKTURTIEFE  3.....................................................
   4.00  .....................................................................
   5.00   JETZT NOCH MAL STRUKTURTIEFE 2.....................................
   6.00 UND WIEDER STRUKTURTIEFE 1.....................................
   7.00 .....................................................................
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
.....................................................................0001.00:001(0)
```

Zeile 2 wird als Strukturtiefe markiert.

```
   5.00   JETZT NOCH MAL STRUKTURTIEFE 2.....................................
   6.00 UND WIEDER STRUKTURTIEFE 1.....................................
   7.00 .....................................................................
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
.....................................................................0005.00:001(0)
```

Mit F1 wird auf den nächsten Datensatz mit gleicher Strukturtiefe positioniert.

X Ändern von Zeilen

Mit X werden Zeilen markiert, die geändert werden können. Der EDT stellt die so markierten Zeilen auf überschreibbar und kennzeichnet sie zusätzlich durch Hell-Steuerung. Die Schreibmarke wird auf den Zeilenanfang der ersten überschreibbaren Zeile innerhalb des Datenfensters positioniert.

Kurzanweisung	Taste
X	DUE oder F2

Änderungen sind in dem im Datenfenster dargestellten Teil der Sätze wirksam. Die anderen Teile der Sätze bleiben unverändert. Beim Einfügen mit EFG ist jedoch zu berücksichtigen, daß Teile der Sätze, die über den Datenfensterrand hinausgeschoben werden, verloren gehen. Dies kann vermieden werden, wenn mit E eingefügt wird.

Zeilen, die nach dem Ändern nur NIL-Zeichen enthalten (erreichbar z.B durch LZE in Spaltenposition 1) werden aus der Arbeitsdatei gelöscht.

Bei Anweisungen in der Anweisungszeile zum Blättern (+, –, ...) werden Änderungsmarkierungen, die noch in der Markierungsspalte stehen, zuerst ausgeführt.

Änderungen des Arbeitsdateiinhalts sind auch durch Eingabe von Anweisungen innerhalb der Anweisungszeile möglich.

Das gesamte Datenfenster kann durch F2 auf überschreibbar gestellt werden (siehe Abschnitt „Die F-Tasten“ auf Seite 73ff.).

Im neuen Arbeitsmodus sind alle Sätze des Datenfensters immer überschreibbar (siehe @PAR EDIT FULL).

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
x  2.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND...........................
   3.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG.......................
   4.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ...........................................................................
```

Zeile 2 wird zum Ändern markiert.

```
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
   2.00 DUCK     DONALD    WALTSTREET 8   33333 DISNEYLAND.....................
   3.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG.......................
   4.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ...........................................................................
```

Die zu ändernde Zeile wird hell dargestellt. Vor DISNEYLAND wird die Postleitzahl eingefügt.

D Löschen einer Satzmarkierung

D F3 löscht eine eventuell vorhandene Satzmarkierung (siehe Löschen von Satzmarkierungen, Seite 123).

Kurzanweisung	Taste
D	F3

m Setzen einer Satzmarkierung

Die Satzmarkierung m wird im angegebenen Satz gesetzt. Zu diesen Satzmarkierungen kann das Datenfenster positioniert werden (siehe Abschnitt „+ / – Positionieren in der Arbeitsdatei“ auf Seite 109).

Kurzanweisung	Taste
m	F3

m Nummer der Satzmarkierung, wobei $1 \leq m \leq 9$ (siehe Setzen von Satzmarkierungen, Seite 123).

m kann nicht als Ganzzahlvariable angegeben werden.

Durch @ON, Format 4 können ebenfalls Satzmarkierungen gesetzt werden.

4.1.5 Anweisung im Datenfenster - Auftrennen eines Datensatzes

Mit @PAR kann ein beliebiges Satztrennzeichen definiert werden (z.B. @PAR SEPARATOR = ';', siehe @PAR).

Wird dieses Satztrennzeichen im Datenfenster in einer Zeile eingegeben, wird der Satz an dieser Stelle aufgetrennt.

Es können mehrere Auftrennstellen in einem Satz angegeben werden. Der erste Satzteil erhält die ursprünglich vergebene Zeilennummer. Die folgenden Satzteile werden als neue Sätze eingefügt. Die Zeilennummernvergabe erfolgt analog zu @COPY, Format 2. Beim Einfügen des Satztrennzeichens ist darauf zu achten, daß am Ende der Datenfensterzeile keine Zeichen verloren gehen.

Das Auftrennen eines Datensatzes erfolgt nur bei der Neueingabe oder Änderung von Datensätzen bzw. des Auftrennzeichens. Wird z.B. ein Datensatz duch kopieren eingefügt, muß mindestens ein Zeichen überschrieben, geändert oder eingefügt werden, um das Auftrennen des Datensatzes mittels Satztrennzeichen zu erreichen.

Die Zeichen NIL oder AM können nicht als Satztrennzeichen definiert werden.

Für Satztrennzeichen und Tabulatorzeichen sind unterschiedliche Zeichen zu wählen.

Bei eingeschalteter Codierfunktion darf das Satztrennzeichen nicht umdefiniert werden.

Leersätze (Satzlänge=0) können nicht erzeugt werden. Es wird jedoch eine Zeilennummer reserviert.

Als Satztrennzeichen sollte ein Zeichen verwendet werden, das nicht in den Daten vorkommt.

4.1.6 Anweisungen in der Anweisungszeile

Im F-Modus werden Anweisungen in der Anweisungszeile (siehe Abschnitt „Das Arbeitsfenster“ auf Seite 65ff.) eingegeben. In der Anweisungszeile können Anweisungen mit oder ohne das Anweisungssymbol @ eingegeben werden, da der EDT hier alle Eingaben als Anweisungen interpretiert.

Auf den folgenden Seiten sind nur die Anweisungen beschrieben, die ausschließlich im F-Modus-Dialog eingegeben werden dürfen. Anweisungen, die sowohl im F-Modus, als auch im L-Modus und in EDT-Prozeduren verwendet werden können, werden im Kapitel „Anweisungen des EDT“ auf Seite 147ff. beschrieben.

+ / – Positionieren in der Arbeitsdatei

Die +/– Anweisungen bieten in zwei Formaten folgende Funktionen:

– innerhalb der aktuellen Arbeitsdatei um ein Datenfenster oder um beliebig viele Zeilen vorwärts oder rückwärts positionieren (Format 1).

– innnerhalb der aktuellen Arbeitsdatei zu Satzmarkierungen positionieren (Format 2).

+/– (Format 1) Positionieren in der Arbeitsdatei

Mit diesen Anweisungen wird in der Arbeitsdatei positioniert.

Operation	Operanden	F-Modus
+ –	[n]	
++ – –		

+ Mit dieser Anweisung wird in der aktuellen Arbeitsdatei um 1 Datenfenster vorwärts (in Richtung Dateiende) positioniert. Der erste Satz im neuen Datenfenster ist der Satz, der auf den letzten des alten Datenfensters folgt.

++ Mit dieser Anweisung wird an das Ende der aktuellen Arbeitsdatei positioniert. Die letzte Zeile des Datenfensters enthält dann den letzten Satz der Arbeitsdatei, wenn die aktuelle Arbeitsdatei mehr Zeilen als das Arbeitsfenster enthält.

– Mit dieser Anweisung wird in der aktuellen Arbeitsdatei um 1 Datenfenster rückwärts (in Richtung Dateianfang) positioniert. In der letzten Zeile des neuen Datenfensters steht der Satz, der vor dem ersten Satz des alten Datenfensters stand.

– – Mit dieser Anweisung wird an den Anfang der aktuellen Arbeitsdatei positioniert. Die erste Zeile des Datenfensters enthält dann den ersten Satz der Arbeitsdatei.

n Beliebige ganze Zahl > 0, mit der bei + oder – eine Distanz angegeben wird.

+n zeigt, ausgehend von der ersten Zeile des Datenfensters, den n-ten Satz danach als ersten Satz des neuen Datenfensters.

–n zeigt, ausgehend von der ersten Zeile des Datenfensters, den n-ten Satz davor als ersten Satz des neuen Datenfensters.

Werden gleichzeitig mit einer Positionieranweisung in der Markierungsspalte die Kurzanweisungen zum Einfügen (I, n) und Ändern von Zeilen (X, E) eingegeben, werden vor dem Positionieren die Kurzanweisungen abgearbeitet.

+/– (Format 2) Positionieren zu Satzmarkierungen

Mit diesem Format von +/– wird zu Sätzen mit Satzmarkierungen positioniert.
Die 1. Zeile des Datenfensters enthält den ersten gefundenen Satz. Beim Weiterblättern wird der nächste markierte Satz zum ersten Satz im Datenfenster.

Wird keine Markierung gefunden, bleibt die Position in der Arbeitsdatei unverändert.

Operation	Operanden	F-Modus
+ ++ – – –	[([m, ...])]	

Die Anweisung wird mit F3 abgeschickt.

Wird die Anweisung mit DUE oder F2 abgeschickt, müssen die Klammern angegeben werden.

+ Es wird zum nächsten Satz in Richtung Dateiende positioniert, der eine Satzmarkierung enthält.

++ Es wird zum letzten Satz der Arbeitsdatei positioniert, der eine Satzmarkierung enthält.

– Es wird zum nächsten Satz in Richtung Dateianfang positioniert, der eine Satzmarkierung enthält.

– – Es wird zum ersten Satz der Arbeitsdatei positioniert, der eine Satzmarkierung enthält.

m Ist eine der 9 möglichen Satzmarkierungen, zu der positioniert werden soll. Es können mehrere Satzmarkierungen angegeben werden.

Markierungen mit Sonderfunktionen (z.B. Markierung 15 für Schreibschutz) werden hier nicht ausgewertet.

Ist m nicht angegeben, wird jede beliebige Satzmarkierung zum Positionieren verwendet.

Beispiel

Siehe Beispiel zu @ON, Format 4

> / < Horizontales Positionieren in der Arbeitsdatei

Mit diesen Anweisungen wird in der Arbeitsdatei horizontal positioniert, d.h. das Datenfenster kann spaltenweise nach rechts bzw. nach links (in Richtung Datensatzende bzw. -anfang) verschoben werden.

Die Spaltennummer, ab der die Sätze im Datenfenster dargestellt werden, wird in der Zustandsanzeige des Arbeitsfensters ausgegeben.

Operation	Operanden	F-Modus
> <	[n]	
<<		

> In der aktuellen Arbeitsdatei wird das Arbeitsfenster um 1 Datenfensterbreite nach rechts verschoben.

< In der aktuellen Arbeitsdatei wird das Arbeitsfenster um 1 Datenfensterbreite nach links verschoben.

<< Es wird auf die Spalte 1 zurückpositioniert.

n Anzahl der Spalten, um die das Arbeitsfenster bei < oder > verschoben werden soll.

$1 \leq n \leq 184$ (bei INDEX ON)

$1 \leq n \leq 176$ (bei INDEX OFF)

>n positioniert um n Spalten nach rechts.

<n positioniert um n Spalten nach links.

<<n ist nicht erlaubt und wird mit einer Fehlermeldung abgewiesen.

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
   2.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND...........................
   3.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG.......................
   4.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..............................................................................









>9.....................................................................0001.00:001(0)
```

Das Datenfenster wird um 9 Spalten nach rechts verschoben.

```
   1.00 ADALBERT  HOCHSTR.10     81234 MUENCHEN ..............................
   2.00 DONALD    WALTSTREET 8   DISNEYLAND...................................
   3.00 GUNDULA   HAFERSTR.16    89123 AUGSBURG...............................
   4.00 LUDWIG    GANGGASSE 3A   80123 MUENCHEN ..............................
   5.00 MANUELA   POSTWEG 3      80123 MUENCHEN ..............................
   6.00 ..............................................................................









.....................................................................0001.00:010(0)
```

Durch das Verschieben beginnt das Datenfenster auf Spalte 10.

Ausgeben der letzten Anweisungen

Mit # werden die letzten vom EDT bereits ausgeführte Anweisungen in der Anweisungszeile erneut ausgegeben. Ausgenommen sind alle Blätteranweisungen sowie die Anweisungen zum Wechseln der Arbeitsdatei.

Operation	Operanden	F-Modus
[n]#		

n Mit n wird die Tiefe angegeben, d.h. die wievielte vorhergegangene Anweisung gezeigt werden soll.

$1 \leq n \leq 256$

oder 1# gibt die letzte vom EDT bereits ausgeführte Anweisung in der Anweisungszeile erneut aus. Mit 2# wird die zweite davorliegende gezeigt. Die #-Anweisung kann in aufeinanderfolgenden Dialogschritten eingegeben werden, dann wird jedesmal im Anweisungspuffer um die angegebenen Stellen vorpositioniert.
Ist der Anfang des Puffers erreicht, bleibt die Anweisungszeile leer. Folgt daraufhin wieder eine #-Anweisung, wird auf die zuletzt gespeicherte Anweisung (Ende des Puffers) zurückpositioniert.
Nach jedem Dialogschritt, der nicht mit # endet, wird wieder am Ende des Puffers aufgesetzt.

Der Anweisungspuffer hat eine feste Größe. Die maximale Anzahl der Rückpositionierungen ist von der Größe der eingegebenen Anweisungen abhängig.

Es wird nicht berücksichtigt, ob eine Anweisung im oberen oder unteren Teil eines geteilten Bildschirmes eingegeben wurde. Die Speicherung im #-Puffer erfolgt nach der Reihenfolge der Eingabe.

Wird # innerhalb einer Anweisungsfolge eingegeben, wird nach der Abarbeitung von # die Verarbeitung abgebrochen und die zuletzt ausgeführte Anweisung ausgegeben.

Nach # stehende Anweisungen werden nicht mehr ausgeführt.

fwkfnr/fwkfv Wechseln der Arbeitsdatei

Mit dieser Anweisung wechselt der EDT in eine andere Arbeitsdatei.

Operation	Operanden	F-Modus
fwkfnr fwkfv		

fwkfnr Nummer (0,...,9) der Arbeitsdatei, in die gewechselt werden soll.

fwkfv Die Arbeitsdateivariable ($0,...,$9) gibt die Arbeitsdatei (0,...,9) an, in die gewechselt werden soll.

Zwischen fwkfnr und fwkfv besteht funktionell kein Unterschied.

Beispiel

```
   1.00 Mit dieser Anweisung wechselt der EDT in eine.....................
   2.00 andere Arbeitsdatei. ..............................................
   3.00 ...................................................................
 7.....................................................................0000.00:001(0)
```

Mit **7** wechselt der EDT in die Arbeitsdatei 7.

```
.....................................................................0000.00:001(7)
```

EDIT LONG Ausgeben von Datensätzen größer 80 Zeichen

Mit EDIT LONG kann die Ausgabe am Bildschirm verändert werden. Für Datensätze, die größer als 80 Zeichen sind, kann bestimmt werden, daß

– die Sätze vollständig im Datenfenster dargestellt werden,

– ein Ausschnitt von 80 Zeichen eines Satzes im Datenfenster dargestellt wird.

Operation	Operanden	F-Modus
EDIT **L**ONG	[**ON**] | **O**FF	

ON Die Sätze werden vollständig im Datenfenster dargestellt.

OFF Von längeren Sätzen ist jeweils nur ein Ausschnitt von 80 Zeichen im Datenfenster sichtbar.

Der EDIT LONG-Modus arbeitet ohne Zeilennummernanzeige. Ein Satz wird fortlaufend über mehrere Zeilen geschrieben. Leerzeichen am Ende eines Satzes werden ignoriert. Das letzte von Leerzeichen und Nilzeichen verschiedene Zeichen des Satzes bestimmt bei Eingabe am Bildschirm das Satzende.

Endet ein Satz am Ende einer Bildschirmzeile, wird als nächste Zeile eine ganze NIL-Zeile ausgegeben.

Wird mit F2 das gesamte Datenfenster auf überschreibbar gestellt, bleibt der letzte im Datenfenster nicht vollständig gezeigte Satz nicht überschreibbar. Wird dieser Satz mit der Kurzanweisung X markiert, so wird die Anweisung ignoriert. Damit dieser Satz geändert werden kann, muß das Datenfenster so positioniert werden, daß der vollständige Satz im Datenfenster dargestellt wird.

Die Markierungsspalte ist die erste Spalte des Bildschirmes. Mehrzeilige Sätze sind in ihrer ersten Zeile zu markieren. Wird beim Erweitern eine Zeile mit E markiert, die kleiner als 240 Zeichen ist, wird eine ganze Nil-Zeile zusätzlich ausgegeben.

Nach EDIT LONG OFF ist die Zeilennummernanzeige ausgeschaltet.

Der EDIT LONG-Modus wird auch durch INDEX ON, INDEX OFF und HEX ON ausgeschaltet.

i Die Anweisungen zum horizontalen Blättern (>,<) werden akzeptiert, jedoch erst nach dem Ausschalten des EDIT LONG-Modus wirksam.

Beispiel

```
   1.00 LFD.NR   ART.NR. ART.NAME       BESTAND   BESTELLT        VERKAUFSPREIS.
   2.00 1        0024    SEIFE          3000      150              1.59 DM......
   3.00 2        0015    DEODORANT      2500      600              4.38 DM......
   4.00 3        0048    PARFUEM        400       60              18.60 DM......
   5.00 4        0003    CREME          987       555              2.83 DM......
   6.00 5        0091    RASIERSCHAUM   350       30               4.39 DM......
   7.00 6        0090    RASIERWASSER   340       30              10.55 DM......
   8.00 7        0092    RASIERPINSEL   200       30              11.50 DM......




edit long on ..................................................0001.00:001(0)
```

Die Datensätze sollen vollständig im Datenfenster dargestellt werden.

```
LFD.NR   ART.NR. ART.NAME       BESTAND   BESTELLT         VERKAUFSPREIS      EI
NKAUFSPREIS.....................................................................
1        0024    SEIFE          3000      150               1.59 DM           1
.10 DM..........................................................................
2        0015    DEODORANT      2500      600               4.38 DM           3
.25 DM..........................................................................
3        0048    PARFUEM        400       60               18.60 DM          14
.00 DM..........................................................................
4        0003    CREME          987       555               2.83 DM           2
.20 DM..........................................................................
5        0091    RASIERSCHAUM   350       30                4.39 DM           3
.70 DM..........................................................................
6        0090    RASIERWASSER   340       30               10.55 DM           9
.80 DM..........................................................................
   7        0092    RASIERPINSEL   200       30              11.50 DM           9
.00 DM..........................................................................
................................................................................
```

Die Datensätze enthalten eine weitere Rubrik EINKAUFSPREIS.

HEX Hexadezimal-Modus einschalten

Mit HEX wird der Hexadezimal-Modus eingeschaltet. Es werden alle Sätze in abdruckbarer und hexadezimaler Form am Bildschirm dargestellt:

1.Zeile: Datenzeile in abdruckbarer Form

2.,3.Zeile: Datenzeile in hexadezimaler Form (vertikale Darstellung)

4.Zeile: Spaltenzähler wie bei SCALE ON. Der mit SCALE eingeschaltete Spaltenzähler wird im HEX-Modus nicht ausgegeben.

Die Hexadezimaldarstellung gilt für die aktuelle Arbeitsdatei, unabhängig vom Arbeitsfenster.

Operation	Operanden	F-Modus
HEX	[**ON**] | **O**FF	

ON Schaltet den Hexadezimal-Modus ein.

OFF Schaltet den Hexadezimal-Modus aus.

Der Hexadezimalmodus wird auch durch EDIT LONG OFF ausgeschaltet.

Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

Ändern von Datenzeilen im Hexadezimalmodus

Die Betätigung von ⇥ bewirkt im Hex-Modus ein Weiterpositionieren zur nächsten Hexadezimal-Zeile (nicht Datenzeile!).

Änderungen können sowohl in der ersten Zeile (abbildbare Form) als auch in der Hexadezimaldarstellung vorgenommen werden. Wurden in einem Dialogschritt sowohl in der ersten Zeile als auch in der Hexadezimaldarstellung Zeichen eingegeben, wird nur die Hexadezimaldarstellung berücksichtigt.

Werden in den Hexadezimal-Zeilen ungültige Hexadezimal-Zeichen (Zeichen ungleich 0...9, A...F) eingegeben, werden die fehlerhaften Zeilen überschreibbar ausgegeben und die fehlerhaften Zeichen mit einem Fragezeichen überschrieben. Die Schreibmarke steht an der ersten fehlerhaften Zeile. Die restlichen, fehlerfreien Zeilen des Datenfensters sind auf nicht überschreibbar gestellt.

Der angezeigt Hexadezimal-Code ist von der CODE-Einstellung (siehe @PAR CODE) abhängig.

Darstellung bei gesplittetem Bildschirm

Bei gesplittetem Bildschirm (@PAR SPLIT) ist die Darstellung im Hexadezimal-Modus von der Anzahl der Datenzeilen (Zeilen im Datenfenster) des jeweiligen Bildschirmes abhängig:

Anzahl der Datenzeilen		**Darstellung im Bildschirm**
1	nur die	Zeichendarstellung des Datensatzes
2	1. Zeile : 2. Zeile :	Zeichendarstellung des Datensatzes Spaltenzählerzeile
3	1. Zeile : 2. u. 3. Zeile:	Zeichendarstellung des Datensatzes Hexadezimaldarstellung (Spaltenzählerzeile entfällt)

Beachte: Anzahl Bildschirmzeilen = Anzahl (Datenzeilen + Anweisungszeile)

Ist die Anzahl der Datenzeilen ein Vielfaches von 4, werden jeweils 4 Bildschirmzeilen pro Satz ausgegeben.

Ist die Anzahl der Datenzeilen ungleich einem Vielfachen von 4, werden die Restzeilen (1, 2 oder 3) gemäß der Übersichtstabelle behandelt.

INDEX Auswählen des Arbeitsfensterformats

Mit INDEX kann das Format des Arbeitsfensters ausgewählt werden. Standardmäßig wird das Format mit 72 Zeichen je Zeile und 6-stelliger Zeilennummernanzeige eingestellt. Die erste Spalte jeder Zeile bildet die Markierungsspalte.

Operation	Operanden	F-Modus
INDEX	[**ON**] | **O**FF	

ON Schaltet das Arbeitsfenster mit dem Standardformat ein.

OFF Schaltet auf das Arbeitsfenster mit 80 Zeichen pro Zeile ohne Zeilennummernanzeige um.

INDEX wirkt bei geteiltem Bildschirm (siehe @PAR SPLIT) auf das Arbeitsfenster, in dem INDEX eingegeben wurde.

Durch INDEX wird der EDIT LONG-Modus ausgeschaltet.

Beispiel

```
   1.00 Mit INDEX kann das Format des Arbeitsfensters ausgewaehlt.............
   2.00 werden. Standardmaessig wird das Format mit 72 Zeichen je Zeile und.....
   3.00 6-stelliger Zeilennummernanzeige eingestellt. .........................
   4.00 Die 1. Spalte jeder Zeile bildet die Markierungsspalte. ...............
   5.00 .....................................................................
index off .....................................................0000.00:001(0)
```

Die Zeilennummernanzeige wird ausgeschaltet.

```
Mit INDEX kann das Format des Arbeitsfensters ausgewaehlt.....................
werden. Standardmaessig wird das Format mit 72 Zeichen je Zeile und.............
6-stelliger Zeilennummernanzeige eingestellt. ................................
Die 1. Spalte jeder Zeile bildet die Markierungsspalte. .......................
.............................................................................
```

SCALE Spaltenzähler ausgeben

SCALE gibt einen Spaltenzähler (Zeilenlineal) im Arbeitsfenster aus. Der Spaltenzähler erscheint als 1. Zeile im Arbeitsfenster (nicht im EDIT-LONG-Modus).

Operation	Operanden	F-Modus
SCALE	[**ON**] | **O**FF	

ON Der Spaltenzähler erscheint als erste Zeile nach einer eventuell vorhandenen Informationszeile und gibt die aktuellen Spaltennummern des Arbeitsfensters an (z.B. nach horizontalem Verschieben des Arbeitsfensters).

Fall ein Tabulator definiert ist (siehe @TABS), wird eine weitere Bildschirmzeile ausgegeben, in der die aktuellen Positionen des Tabulators mit „I“ angezeigt werden.

Das Tabulatorzeichen wird in der Markierungsspalten-Position abgebildet.

OFF Ausschalten des Spaltenzählers und des eventuell vorhandenen Tabulator-Anzeigelineals.

Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

SCALE wirkt bei geteiltem Bildschirm (siehe @PAR SPLIT) nur auf das Arbeitsfenster, in dem SCALE eingegeben wurde.

Im EDIT LONG-Modus wird der Spaltenzähler nicht ausgegeben.

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................









scale on ..................................................0001.00:001(0)
```

Zum Überprüfen der Spaltennummern wird ein Spaltenzähler angefordert.

```
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................
```

SPLIT Ausgeben von 2 Arbeitsfenstern

Mit SPLIT wird ein zweites Arbeitsfenster am Bildschirm ausgegeben. Jedes Arbeitsfenster hat eine eigene Anweisungszeile.

Die Schreibmarke wird nach der Teilung des Bildschirms auf die obere Anweisungszeile positioniert. Nach jeder weiteren Ausgabe wird sie auf jene Anweisungszeile positioniert, in der die letzte Anweisung bzw. Anweisungsfolge eingegeben wurde.

Wird in beiden Anweisungszeilen eine Anweisung eingegeben, wird auf die obere Anweisungszeile positioniert. Tritt bei der Abarbeitung einer Anweisung ein Fehler auf, wird auf jene Anweisungszeile positioniert, in der die fehlerhafte Anweisung eingegeben wurde.

Wird bei geteiltem Bildschirm in der oberen Anweisungszeile SPLIT OFF und in der unteren Anweisungszeile eine Anweisung eingegeben, wird SPLIT OFF mit einer Fehlermeldung abgewiesen.

Nach dem Aufruf des EDT ist SPLIT OFF voreingestellt.

Operation	Operanden	F-Modus
SPLIT	n(fwkfnr) | 0 | **O**FF	

n — Zeilenanzahl des unteren Arbeitsfensters inklusive Anweisungszeile:

$2 \leq n \leq 22$

fwkfnr — Nummer der Arbeitsdatei, die im unteren Arbeitsfenster gezeigt werden soll. Im oberen Arbeitsfenster wird die Arbeitsdatei gezeigt, in der die Anweisung gegeben wurde.

0 — Das Arbeitsfenster wird vollständig (24 Zeilen) gezeigt, in dessen

OFF — Anweisungszeile die Anweisung eingegeben wurde.

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..................................................................


split 10(2) ..................................................0001.00:001(0)
```

Mit SPLIT 10(2) wird ein zweites Arbeitsfenster angefordert, das 10 Zeilen inklusive Anweisungszeile umfaßt. In diesem Arbeitsfenster soll die Arbeitsdatei 2 ausgegeben werden.

```
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..................................................................
   7.00 ..................................................................
   8.00 ..................................................................
   9.00 ..................................................................
  10.00 ..................................................................
  11.00 ..................................................................
  12.00 ..................................................................
  13.00 ..................................................................
.....................................................................0001.00:001(0)
   1.00 SIE KOENNEN JETZT..................................................
   2.00 ABWECHSELND ODER GLEICHZEITIG.....................................
   3.00 DIE ARBEITSDATEIEN 0 UND 2.........................................
   4.00 BEARBEITEN.........................................................
   5.00 ..................................................................
   6.00 ..................................................................
   7.00 ..................................................................
   8.00 ..................................................................
   9.00 ..................................................................
.....................................................................0001.00:001(2)
```

4.1.7 Beschreibung der Satzmarkierungen des F-Modus

Jeder Satz in der virtuellen Arbeitsdatei des EDT kann mit mehreren Satzmarkierungen gekennzeichnet werden. Die Satzmarkierungen werden im virtuellen Datenbereich des EDT vermerkt und sind für den Benutzer nicht sichtbar. Sie werden nicht in die reale Datei mitübernommen.

Die Satzmarkierungen können beim Bearbeiten der Arbeitsdatei benutzt werden (siehe @ON, +(m), usw.).

Durch @OPEN, Format 1 real eröffnete ISAM-Dateien können nicht markiert werden.

Setzen von Satzmarkierungen

Satzmarkierungen können gesetzt werden durch

– @ON in der Anweisungszeile (siehe @ON, Format 4).

– die Funktionen IEDTPUT und IEDTPTM bei Aufruf des EDT als Unterprogramm (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]; Schreiben eines Satzes und Markieren eines Satzes)

– die Kurzanweisung m F3 in der Markierungsspalte. Die 1 Zeichen langen Satzmarkierungen (1 bis 9) sind in der Markierungsspalte anzugeben und mit F3 zu übertragen.

Löschen von Satzmarkierungen

Die Satzmarkierungen einer Arbeitsdatei werden durch @DELETE MARK gelöscht (siehe @DELETE MARK).

Durch @COMPARE werden die Satzmarkierungen der verglichenen Arbeitsdateien gelöscht.

Die Kurzanweisung D F3 in der Markierungsspalte löscht eventuell vorhandene Satzmarkierungen des jeweiligen Datensatzes.

i In den vorangegangenen Abschnitten wurde bereits der Begriff „markieren“ im Zusammenhang mit der Markierungsspalte und den Kurzanweisungen, die dort eingegeben werden können, gebraucht. Diese Form der Markierung ist als Funktionskennzeichen zu verstehen. Sie ist nach Ausführung der Kurzanweisung nicht mehr vorhanden.

4.1.8 Anweisungen im F-Modus

Folgende Anweisungen sind im F-Modus erlaubt:

– – [(m)]	@GETLIST	@SETLIST
+ [(m)]	@GETVAR	@SETSW
– [(m)]	@HALT	@SETVAR
<	@INPUT ***)	@SHOW
<<	@LIMITS	@SORT
>	@LIST	@STAJV
#	@LOAD	@STATUS
@:	@LOG	@SUFFIX
@AUTOSAVE	@LOWER	@SYMBOLS
@BLOCK	@MOVE	@SYNTAX
@CLOSE	@ON **)	@SYSTEM
@CODE	@OPEN	@TABS **)
@CODENAME	@P-KEYS *)	@TMODE
@COLUMN	@PAGE	@UNLOAD
@COMPARE	@PAR	@UNSAVE
@COPY	@PREFIX	@USE
@CREATE (Format 1)	@PRINT	@VDT
@DELETE	@QUOTE	@VTCSET
@DELIMIT	@RANGE	@WRITE
@DO (Format 1)	@READ	@XCOPY
@DROP	@RENUMBER	@XOPEN
@EDIT	@RESET	@XWRITE
@ELIM	@RETURN	EDIT LONG
@END	@RUN	HEX
@ERAJV	@SAVE	INDEX
@EXEC	@SDFTEST	SCALE
@FILE	@SEQUENCE	SPLIT
@FSTAT	@SET	fwkfnr
@GET	@SETF	fwkfv
@GETJV	@SETJV	

*) Wird von der DSS 3270 nicht unterstützt.

**) Nicht @ON, Format 3.

***) Nicht @INPUT, Format 3.

Im F-Modus kann das EDT-Anweisungssymbol (standardmäßig @) auch weggelassen werden.

Beschreibung der Anweisungen siehe Kapitel „Anweisungen des EDT“ auf Seite 147ff.

4.2 L-Modus

Im LINE-Modus (L-Modus) erfolgt die Dateibearbeitung zeilenorientiert, d.h. der EDT bietet nur eine Zeile an. In diese Zeile können sowohl Datensätze als auch Anweisungen geschrieben werden. Datensätze werden in die jeweils aktuelle Zeile geschrieben. Anweisungen werden sofort ausgeführt. Zur Unterscheidung dient das Anweisungssymbol @ (siehe weiter unten).

Der L-Modus ist im Dialog- und Stapelbetrieb möglich.

Nur im L-Modus unterstützt werden:

– EDT-Prozeduren
– INPUT-Dateien
– das Lesen von SYSDTA mit RDATA (Systemprozeduren, Stapelbetrieb)

Anweisungen aus dem F-Modus werden im L-Modus als Datensätze interpretiert.

Das Umschalten in den L-Modus erfolgt mit @EDIT.

4.2.1 Eingabe im L-Modus

Der EDT interpretiert eine Eingabe im L-Modus als Anweisung, wenn:

– das erste Zeichen das Anweisungssymbol (standardmäßig @) ist
– und das folgende Zeichen kein Anweisungssymbol ist.

Ansonsten wird Eingabe als Text in die aktuelle Zeile geschrieben.

Führende und folgende Leerzeichen werden vom EDT ignoriert. So interpretiert der EDT eine Eingabe mit mehr als einem Anweisungssymbol (@@, bzw. @ @, ...), nicht als Anweisung. In diesem Fall wird die Eingabe als Text in der aktuellen Zeile gespeichert. Dabei werden die Zeichen (erstes Anweisungssymbol und Leerzeichen), die vor dem zweiten Anweisungssymbol stehen abgeschnitten.
Eingaben mit mehr als einem Anweisungssymbol dienen zum Erstellen von Prozeduren (siehe auch Kapitel „EDT-Prozeduren“ auf Seite 127ff.). Die Eingabe wird nicht sofort als Anweisung ausgeführt, sondern sozusagen als Anweisung abgespeichert und kann somit später mehrfach zur Ausführung gebracht werden.

Alle Eingaben ohne Anweisungssymbol werden grundsätzlich als Datensatz interpretiert und sofort in der aktuellen Zeile abgelegt.

4.2.2 Anweisungen im L-Modus

Folgende Anweisungen sind im L-Modus erlaubt:

@	@GETLIST	@SDFTEST
@+	@GETVAR	@SEQUENCE
@–	@HALT	@SET
@:	@IF (Format 1)	@SETF
@AUTOSAVE	@INPUT	@SETJV
@BLOCK	@LIMITS	@SETLIST
@CHECK	@LIST	@SETSW
@CLOSE	@LOAD	@SETVAR
@CODE	@LOG	@SHOW
@CODENAME	@LOWER	@SORT
@COLUMN	@MOVE	@STAJV
@COMPARE	@NOTE	@STATUS
@CONTINUE	@ON	@SUFFIX
@COPY	@OPEN	@SYMBOLS
@CREATE	@P-KEYS	@SYNTAX
@DELETE	@PAGE	@SYSTEM
@DELIMIT	@PAR	@TABS
@DIALOG	@PREFIX	@TMODE
@DO (Format 1)	@PRINT	@UNLOAD
@DROP	@PROC	@UNSAVE
@EDIT	@QUOTE	@UPDATE
@ELIM	@RANGE	@USE
@END	@READ	@VDT
@ERAJV	@RENUMBER	@VTCSET
@EXEC	@RESET	@WRITE
@FILE	@RETURN	@XCOPY
@FSTAT	@RUN	@XOPEN
@GET	@SAVE	@XWRITE
@GETJV		

Folgende Anweisungen sind nicht in EDT-Prozeduren erlaubt:

@CODENAME, @DROP

Folgende Anweisungen sind nur in EDT-Prozeduren erlaubt:

@GOTO, @DO (Format 2), @IF (Format 2, 3, 4), @PARAMS

Das EDT-Anweisungssymbol (standardmäßig @) muß im L-Modus angegeben werden.

Beschreibung der Anweisungen siehe Kapitel „Anweisungen des EDT“ auf Seite 147ff.

5 EDT-Prozeduren

Während der Arbeit mit dem EDT kann es vorkommen, daß Bearbeitungen von Dateien mit gleichen oder ähnlichen Anweisungsfolgen vorgenommen werden müssen. Diese mehrfach benötigten Anweisungsfolgen können in sogenannten EDT-Prozeduren zusammengefaßt und beliebig oft und jederzeit zur Ausführung gebracht werden.

Neben den EDT-Anweisungen, die nur in EDT-Prozeduren Anwendung finden, sind alle EDT-Anweisungen erlaubt, die im L-Modus auch direkt eingeben werden können.

EDT-Prozeduren können ausgeführt werden:

– aus Arbeitsdateien (temporär während einer EDT-Sitzung) oder
– aus katalogisierten Dateien (SAM- oder ISAM-Datei).

5.1 Eingabequellen des EDT

Der EDT erwartet die Eingabe der Anweisungen entweder:

– direkt vom Bildschirm (F- oder L-Modus-Dialog),
– von einer katalogisierten Datei oder von einem Bibliothekselement als @INPUT-Prozedur oder
– von einer Arbeitsdatei als @DO-Prozedur.

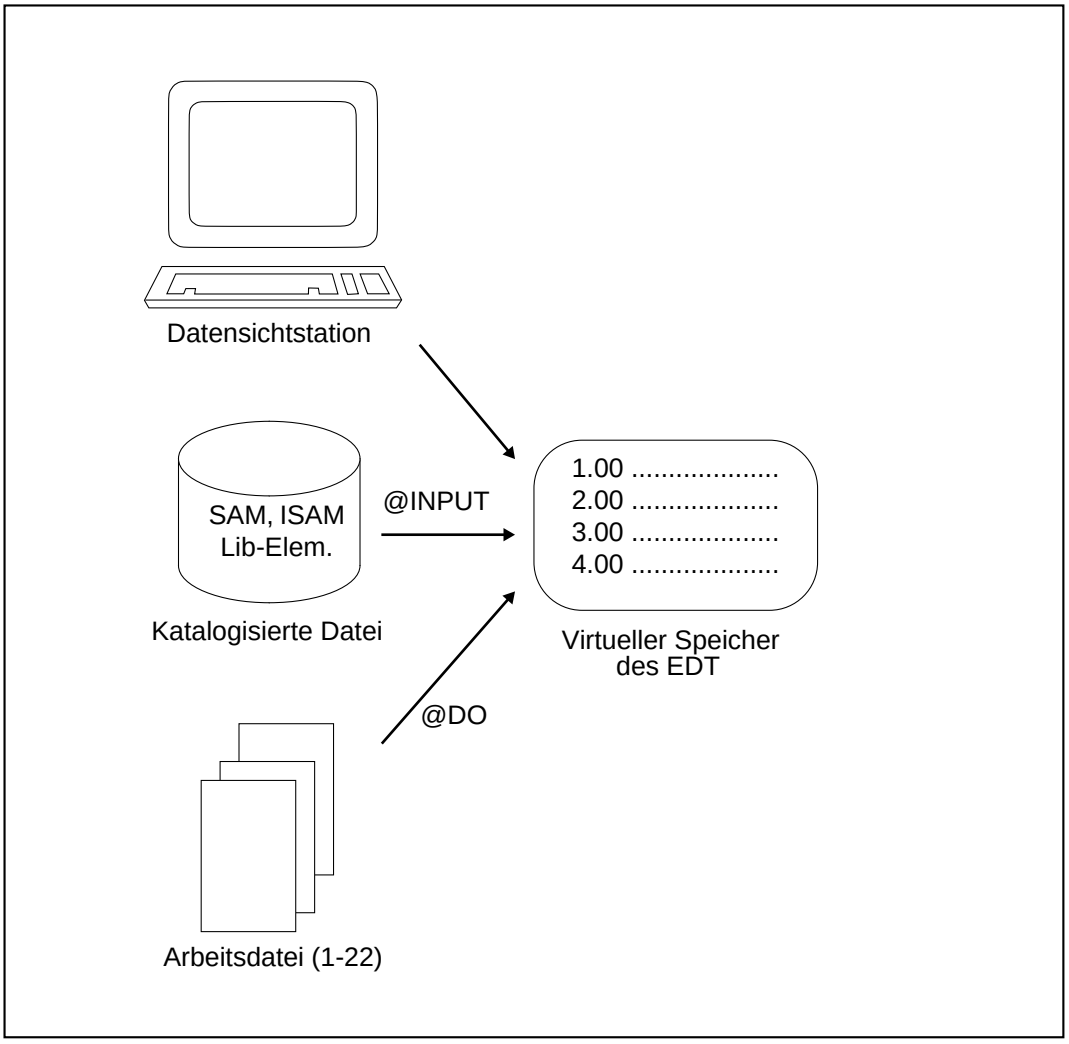


Bild 7: Anweisungen an den EDT

Durch @DO, @INPUT, @EDIT, @DIALOG, @RETURN und @HALT kann die Eingabequelle verändert werden (siehe auch entsprechende Anweisungen).

5.2 EDT-Variablen

Die EDT-Variablen dienen zum Ablegen von Werten. Diese Werte können sein: ganze Zahlen, Zeichenfolgen oder Zeilennummern. Die vielfältigen Anwendungsbereiche von Variablen liegen z.B. im Zwischenspeichern von Werten, Festlegen von Schleifenzählern und Abbruchbedingungen, zur Eingabe von Zeichenfolgen (Dateinamen, Suchbegriffe, u.ä.) oder um einfache Berechnungen durchzuführen.

EDT-Variablen sind nur während des EDT-Laufs gültig. Sie können von allen Arbeitsdateien aus belegt, benutzt oder abgefragt werden. D.h. wird einer Variablen in einer Arbeitsdatei ein Wert zugewiesen, so steht die Variable mit dem gleichen Wert auch in einer anderen Arbeitsdatei zur Verfügung.

Der EDT bietet drei Arten von Variablen, die mit entsprechenden Werten versorgt werden können. Von jeder Variablenart stehen 21 Variablen mit dem Index 0 bis 20 zur Verfügung.

– Ganzzahlvariablen (#I0-#I20)
– Zeichenfolgevariablen (#S0-#S20)
– Zeilennummervariablen (#L0-#L20)

Die EDT-Variablen werden mit den Formaten der @SET-Anweisung oder über @CREATE mit Werten versorgt (siehe @SET, Format 1-5 und @CREATE).

Die Zeilennummervariable #L0 und die Ganzzahlvariablen #I0 und #I1 sollten nicht verwendet werden, da sie im Fall eines Treffers bei der @ON-Anweisung mit Werten überschrieben werden.

Ganzzahlvariablen

In den Ganzzahlvariablen (#I0-#I20) können positive bzw. negative Ganzzahlen abgelegt werden. Die größtmögliche Zahl ist 2.147.483.647 ($2^{31} - 1$).

Die Ganzzahlvariablen können über die @SET-Anweisung, Format 1 mit Werten versorgt werden. Mit @STATUS kann der Inhalt der Ganzzahlvariablen am Bildschirm ausgegeben werden.

Zeichenfolgevariablen

In den Zeichenfolgevariablen (#S0-#S20) können Zeichenfolgen im EBCDI-Code abgelegt werden. Eine Zeichenfolge kann bis zu 256 Zeichen lang sein

Die Zeichenfolgevariablen können über die @SET-Anweisung, Format 2 und 5 oder über @CREATE mit Werten versorgt werden. Mit @PRINT kann der Inhalt der Zeichenfolgevariablen am Bildschirm ausgegeben werden.

Zeichenfolgevariablen werden beim Aufruf des EDT mit einem Leerzeichen vorbelegt, wenn sie nicht die Werte der eventuell vorhandenen S-Variablen SYSEDT-S00, ... SYSEDT-S20 übernommen haben (siehe Abschnitt „Starten des EDT“ auf Seite 25ff.).

Zeilennummervariablen

In den Zeilennummervariablen (#L0-#L20) können Zeilennummern abgelegt werden. Der Wertebereich liegt zwischen 0.0001 und 9999.9999.

Die Zeilennummervariablen können über die @SET-Anweisung, Format 3 mit Werten versorgt werden. Mit @STATUS kann der Inhalt der Zeilennummervariablen am Bildschirm ausgegeben werden.

Jobvariablen

In Systemen, in denen das Subsystem ’Jobvariablen Support’ installiert ist, können im EDT Jobvariablen (JV) benutzt werden. Im Gegensatz zu den Ganzzahl-, Zeichenfolge- und Zeilennummervariablen bleiben Jobvariablen auch nach Beenden des EDT bestehen, bzw. kann im EDT auf bestehende Jobvariablen zugegriffen werden (siehe Abschnitt „Jobvariable“ auf Seite 58).

S-Variablen

In Systemen, in denen das Subsystem SDF-P installiert ist, können im EDT S-Variablen benutzt werden. Im Gegensatz zu den Ganzzahl-, Zeichenfolge- und Zeilennummervariablen bleiben S-Variablen auch nach Beenden des EDT bestehen, bzw. kann im EDT auf bestehende S-Variablen zugegriffen werden (siehe Abschnitt „SDF-P-Unterstützung“ auf Seite 59).

Inhalte von S-Variablen können Zeichenfolgevariablen bzw. Inhalte von Zeichenfolgevariablen können S-Variablen zugeordnet werden und zwar:

– implizit beim Starten bzw. Beenden des EDT (siehe Abschnitt „Starten des EDT“ auf Seite 25ff.).
– explizit über @GETVAR bzw. @SETVAR.

5.3 Erstellen, Aufruf und Ablauf von EDT-Prozeduren

Eingabe in einer EDT-Prozedur

Die Eingabe, die der EDT aus einer Prozedur abzuarbeiten hat, kann sein:

– eine Zeichenfolge (beliebiger Text).

  Sie beginnt nicht mit dem Anweisungssymbol (standardmäßig @). Der EDT übernimmt die Zeichenfolge als Daten in die aktuelle Ausgabedatei.

  Beginnt die Zeichenfolge mit mehr als einem Anweisungssymbol (z.B @@...) oder mehr als einem Benutzerfluchtsymbol, übernimmt der EDT die Zeichenfolge mit einem Anweisungssymbol oder Benutzerfluchtsymbol weniger als Daten in die aktuelle Ausgabedatei.

– eine Anweisung.

  Sie beginnt mit genau einem Anweisungssymbol (standardmäßig @). Der EDT führt die Anweisung sofort aus.

– eine externe Anweisung.

  Sie beginnt mit dem Benutzerfluchtsymbol (siehe @USE). Der EDT führt die externe Anweisung sofort aus.

Anweisungs- und Textfolgen können in jede Arbeitsdatei geschrieben werden.

Erstellen von EDT-Prozeduren

EDT-Prozeduren sollten im F-Modus erstellt werden. So können die vielfältigen Möglichkeiten für die Bearbeitung von Daten auch für das Erstellen von EDT-Prozeduren verwendet werden.

Im F-Modus stehen zum Erstellen von EDT-Prozeduren insgesamt 10 Arbeitsdateien (0-9), im L-Modus 23 Arbeitsdateien (0-22) zur Verfügung. Für den späteren Ablauf der Prozeduren ist jedoch folgendes zu beachten:

– In der Arbeitsdatei 0 können EDT-Prozeduren nur erstellt werden. Sie können hier jedoch nicht zum Ablauf gebracht werden.

– Die Arbeitsdateien 9 und 10 dienen einigen Anweisungen (z.B. @COMPARE oder @FSTAT) als Ausgabedateien. Dadurch kann eine bestehende EDT-Prozedur in diesen Arbeitsdateien ggf. gelöscht werden.

Aufruf einer EDT-Prozedur

Es gibt zwei Arten von EDT-Prozeduren

– @DO-Prozeduren (Aufruf mit @DO)

– @INPUT-Prozeduren (Aufruf mit @INPUT)

Unterschiede zwischen @DO- und @INPUT-Prozeduren

@DO-Prozedur	**@INPUT-Prozedur**
steht in einer Arbeitsdatei	steht in einer katalogisierten Datei auf Platte bzw. in einem Bibliothekselement
es kann nur die ganze Prozedur ausgeführt werden	es können auch Teile der Prozedur ausgeführt werden
mehrere @DO-Prozeduren können ineinander geschachtelt werden, weiterer @DO-Aufruf möglich Aufruf aus einer @INPUT-Prozedur möglich	kann nur mit einer @DO-Prozedur geschachtelt werden, d.h. kein weiterer @INPUT-Aufruf möglich
Sprunganweisungen innerhalb einer @DO-Prozedur	keine Sprunganweisungen
Parameterübergabe möglich	keine Parameter

Ablauf von Prozeduren

Prozeduren laufen immer im L-Modus ab.

Beim Aufruf einer Prozedur im F-Modus wird

– automatisch in den L-Modus verzweigt,

– die Prozedur abgearbeitet und am Ende der Prozedur wieder

– automatisch in den F-Modus gewechselt.

5.4 @DO-Prozeduren

Anweisungen, die in einer der Arbeitsdateien 1-9 (L-Modus 1-22) stehen, können direkt aus einer anderen Arbeitsdatei mit @DO aufgerufen werden.

Die Vorteile einer @DO-Prozedur bestehen in der Möglichkeit:

– die Prozeduren direkt aus einer Arbeitsdatei aufzurufen

– Parameter zu übergeben (siehe @PARAMS)

– @DO-Prozeduren zu schachteln (weitere @DO-Aufrufe innerhalb einer Prozedur).

@DO-Prozeduren stehen während der gesamten EDT-Sitzung in den Arbeitsdateien des EDT zur Verfügung, d.h. bei Beenden des EDT müssen sie in eine katalogisierte Datei auf Platte geschrieben werden, ansonsten werden sie gelöscht.

Beispiel : Aufruf einer Prozedur im F-Modus als @DO-Prozedur

```
    1.00 ..........................................................................
    2.00 ..........................................................................
    3.00 ..........................................................................
    4.00 ..........................................................................
    5.00 ..........................................................................
    6.00 ..........................................................................
    7.00 ..........................................................................









 23.00 ..........................................................................
1.....................................................0001.00:001(0)
```

Es wird in die Arbeitsdatei 1 gewechselt. In ihr wird die Prozedur erstellt.

```
    1.00 @READ 'TESTDATEI'..........................................................
    2.00 @PAR LOWER=ON...............................................................
    3.00 ..........................................................................
    4.00 ..........................................................................
    5.00 ..........................................................................
    6.00 ..........................................................................
    7.00 ..........................................................................









 23.00 ..........................................................................
0.....................................................0001.00:001(1)
```

Die Anweisungen werden mit dem Anweisungssymbol in der Arbeitsdatei 1 erstellt. Anschließend wird wieder in die Arbeitsdatei 0 zurückgekehrt.

```
    1.00 ..........................................................................
    2.00 ..........................................................................
    3.00 ..........................................................................
    4.00 ..........................................................................
    5.00 ..........................................................................
    6.00 ..........................................................................
    7.00 ..........................................................................









 23.00 ..........................................................................
do 1.....................................................0001.00:001(0)
```

Aus der Arbeitsdatei 0 wird die Prozedur in Arbeitsdatei 1 mit @DO aufgerufen.

```
    1.00 Dies ist eine Testdatei,...................................................
    2.00 die mit einer Prozedur.....................................................
    3.00 ins Arbeitsfenster 0 eingelesen wurde. .....................................
    4.00 ..........................................................................
    5.00 ..........................................................................
    6.00 ..........................................................................
    7.00 ..........................................................................









 23.00 ..........................................................................
.....................................................0001.00:001(0)
```

Ergebnis des Prozedurlaufes.

Beispiel : Aufruf einer Prozedur im L-Modus als @DO-Prozedur

```
1.     @PROC 1 ------------------------------------------------------ (01)
1.     @ @CREATE 1: 'DIES IST EIN BEISPIEL'
2.     @ @CREATE 2: 'FUER EINE PROZEDUR IM L-MODUS'
3.     @ @COPY 1 TO 3                                 ------------------ (02)
4.     @ @DELETE 1:1-9
5.     @ @PRINT
6.     @END --------------------------------------------------------- (03)
1.     @DO 1 -------------------------------------------------------- (04)
1.0000 EIN BEISPIEL
2.0000 FUER EINE PROZEDUR IM L-MODUS
3.0000 DIES IST EIN BEISPIEL
4.
```

(01) Es wird in die Arbeitsdatei 1 umgeschaltet.

(02) 5 EDT-Anweisungen werden in die Arbeitsdatei 1 geschrieben (@DO-Prozedur).

(03) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(04) Aufruf der @DO-Prozedur. Die Anweisungen in der Arbeitsdatei werden ausgeführt.

Weitere Beipiele siehe auch Kapitel „Anweisungen des EDT“ auf Seite 147ff. (z.B. @DO).

5.5 @INPUT-Prozeduren

EDT-Anweisungen können als @INPUT-Prozeduren in eine SAM- oder ISAM-Datei auf Platte geschrieben werden.

Die Vorteile von @INPUT-Prozeduren bestehen in der Möglichkeit:

– Prozeduren jederzeit während einer beliebigen EDT-Sitzung aufrufen zu können,

– nur Teile der Prozedur ausführen zu lassen,

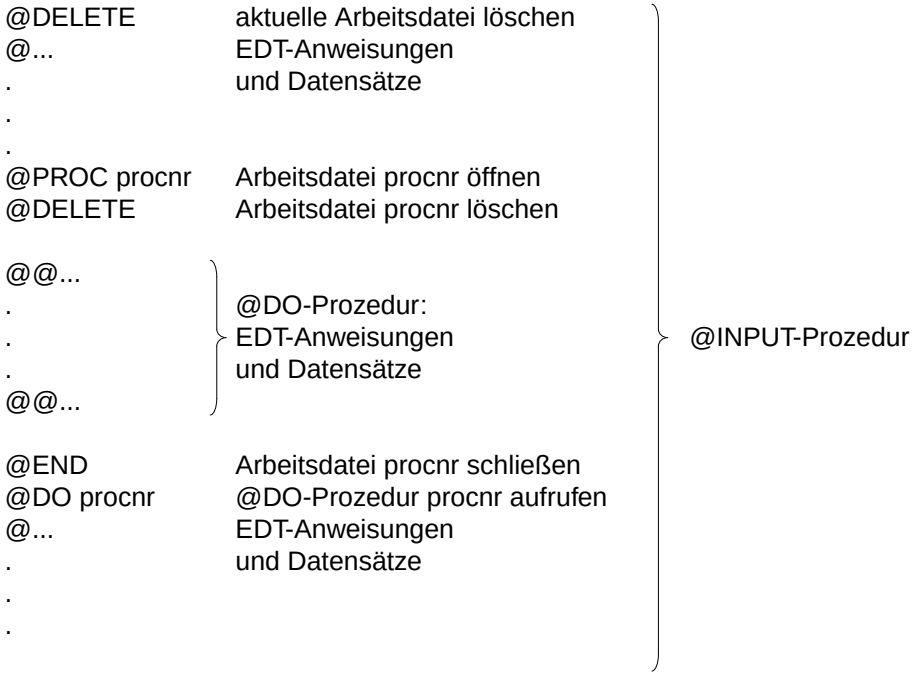
– @INPUT-Prozeduren mit @DO-Prozeduren zu schachteln (@DO-Aufruf innerhalb einer @INPUT-Prozedur),

@INPUT-Prozeduren können nicht ineinander geschachtelt werden. In einer reinen @INPUT-Prozedur (ohne @DO) dürfen keine Sprunganweisungen und Parameter definiert werden.

Um alle Vorteile von @INPUT-Prozeduren und @DO-Prozeduren zu nutzen, können @INPUT-Prozeduren mit @DO-Prozeduren geschachtelt werden.

Aufbau einer geschachtelten @INPUT-Prozedur

(@DO-Prozedur innerhalb einer @INPUT-Prozedur)



Beispiel für eine geschachtelte @INPUT-Prozedur

```
   1.00  @NOTE *** DATEINAME EINLESEN ***.....................................
   2.00 @CREATE #S00 READ 'DATEINAME: '.......................................
   3.00  @NOTE *** ARBEITSDATEI 1 AUFKLAPPEN ***..............................
   4.00 @PROC 1...............................................................
   5.00 @DELETE...............................................................
   6.00  @NOTE *** DATUM IN DER FORM jjmmtt IN #S01 ABLEGEN ***.................
   7.00 @ @SET    #S01 = DATE ISO.............................................
   8.00 @ @SET    #S01 = #S01:1-8.............................................
   9.00 @ @ON     #S01 CHANGE ALL '-' TO ''...................................
  10.00  @NOTE *** UHRZEIT IN DER FORM hhmmss IN #S02 ABLEGEN ***..............
  11.00 @ @SET    #S02 = TIME................................................
  12.00  @NOTE *** COPY-FILE-KOMMANDO ZUSAMMENSETZEN UND IN #S03 ABLEGEN........
  13.00 @ @CREATE #S03 : '/COPY-FILE ',#S00,',',#S00,'.',#S01,'.',#S02.........
  14.00  @NOTE *** COPY-FILE-KOMMANDO ALS SYSTEM-KOMMANDO ABSETZEN ***.........
  15.00 @ @SYSTEM #S03.......................................................
  16.00  @NOTE *** ARBEITDATEI 1 ZUKLAPPEN ***.................................
  17.00 @END.....................................................................
  18.00 @DO 1....................................................................
  19.00 .....................................................................
  20.00 .....................................................................
  21.00 .....................................................................
  22.00 .....................................................................
  23.00 .....................................................................
write 'sicherungskopie.input'.....................................0001.00:001(1)
```

Die Prozedur wird im F-Modus in der Arbeitsdatei 1 erstellt und als SAM-Datei unter dem Namen SICHERUNGSKOPIE.INPUT gespeichert.

Die @DO-Prozedur besteht aus allen Anweisungen mit @@ (siehe weiter unten).

```
   1.00  @NOTE *** DATEINAME EINLESEN ***.....................................
   2.00 @CREATE #S00 READ 'DATEINAME: '.......................................
   3.00  @NOTE *** ARBEITSDATEI 1 AUFKLAPPEN ***..............................
   4.00 @PROC 1...............................................................
   5.00 @DELETE...............................................................
   6.00  @NOTE *** DATUM IN DER FORM jjmmtt IN #S01 ABLEGEN ***.................
   7.00 @ @SET    #S01 = DATE ISO.............................................
   8.00 @ @SET    #S01 = #S01:1-8.............................................
   9.00 @ @ON     #S01 CHANGE ALL '-' TO ''...................................
  10.00  @NOTE *** UHRZEIT IN DER FORM hhmmss IN #S02 ABLEGEN ***..............
  11.00 @ @SET    #S02 = TIME................................................
  12.00  @NOTE *** COPY-FILE-KOMMANDO ZUSAMMENSETZEN UND IN #S03 ABLEGEN........
  13.00 @ @CREATE #S03 : '/COPY-FILE ',#S00,',',#S00,'.',#S01,'.',#S02.........
  14.00  @NOTE *** COPY-FILE-KOMMANDO ALS SYSTEM-KOMMANDO ABSETZEN ***.........
  15.00 @ @SYSTEM #S03.......................................................
  16.00  @NOTE *** ARBEITDATEI 1 ZUKLAPPEN ***.................................
  17.00 @END.....................................................................
  18.00 @DO 1....................................................................
  19.00 .....................................................................
  20.00 .....................................................................
  21.00 .....................................................................
  22.00 .....................................................................
%  EDT0160 FILE 'SICHERUNGSKOPIE.INPUT' WRITTEN
input 'sicherungskopie.input';1...................................0001.00:001(0)
```

Durch @INPUT wird die Prozedur aufgerufen. Die Anweisungen der SAM-Datei SICHERUNGSKOPIE.INPUT werden abgearbeitet. Dabei wird die geschachtelte @DO-Prozedur (Zeilen 6.00 bis 16.00, EDT-Anweisungen mit mehr als einem Anweisungssymbol @) in der Arbeitsdatei 1 abgelegt (siehe weiter unten). Anschließend wird in die Arbeitsdatei 1 gewechselt.

```
   DATEINAME: testdatei
```

Bei Abarbeitung der Anweisung @CREATE ... READ wird die Eingabeaufforderung ausgegeben. Nach Eingabe des Dateinamens TESTDATEI wird eine Sicherungskopie mit dem Namen TESTDATEI.jjmmtt.hhmmss erstellt.

```
   1.00 @SET    #S01 = DATE ISO.............................................
   2.00 @SET    #S01 = #S01:1-8.............................................
   3.00 @ON     #S01 CHANGE ALL '-' TO ''...................................
   4.00 @SET    #S02 = TIME.................................................
   5.00 @CREATE #S03 : '/COPY-FILE ',#S00,',',#S00,'.',#S01,'.',#S02........
   6.00 @SYSTEM #S03.......................................................
   7.00 ...................................................................




.....................................................................0001.00:001(1)
```

Die Anweisungen der @INPUT-Prozedur mit mehr als einem Anweisungssymbol wurden in der Arbeitsdatei 1 als @DO-Prozedur mit einem Anweisungssymbol weniger abgelegt.

5.6 Aufruf einer EDT-Prozedur in einer BS2000-Systemprozedur

Eine EDT-Prozedur kann aus einer BS2000-Systemprozedur aufgerufen werden.

Näheres zu BS2000-Systemprozeduren siehe auch Handbuch „Benutzer-Kommandos (SDF-Format)“ [2] BEGIN-PROCEDURE.

Beispiel für eine EDT-Prozedur in einer BS2000-Systemprozedur

```
/BEGIN-PROCEDURE -
/  PARAMETER=YES (PROCEDURE-PARAMETERS=(&DATEI),ESCAPE-CHARACTER='&')
/SHOW-FILE-ATTRIBUT &DATEI ------------------------------------------------ (01)
/ASSIGN-SYSDTA TO-FILE=*SYSCMD -------------------------------------------- (02)
/MODIFY-JOB-SWITCHES ON=(4,5) --------------------------------------------- (03)
/START-PROGRAM $EDT ------------------------------------------------------- (04)
@DELETE ------------------------------------------------------------------- (05)
@PROC 1 ------------------------------------------------------------------- (06)
@DELETE ------------------------------------------------------------------- (07)
@ @READ '&DATEI'
@ @PAR LOWER=ON
@ @PAR SCALE=ON          ---------------------------------------------------- (08)
@ @PAR INFORMATION=ON
@ @PAR EDIT FULL=ON
@END ---------------------------------------------------------------------- (09)
@DO 1 --------------------------------------------------------------------- (10)
@DIALOG ------------------------------------------------------------------- (11)
@HALT --------------------------------------------------------------------- (12)
/ASSIGN-SYSDTA TO-FILE=*PRIMARY ------------------------------------------- (13)
/MODIFY-JOB-SWITCHES OFF=(4,5) -------------------------------------------- (14)
/SET-JOB-STEP
/END-PROCEDURE
```

(01) Prüfen, ob Datei vorhanden ist. Wenn nicht, wird zu SET-JOB-STEP verzweigt.

(02) Eingabequelle zuordnen. Das System liest über SYSCMD nicht nur die Kommandos, sondern auch Daten ein.

(03) Auftragsschalter 4 und 5 setzen (siehe Abschnitt „Auftragsschalter“ auf Seite 60ff.).

(04) EDT aufrufen.

(05) Der Inhalt der Arbeitsdatei 0 wird gelöscht.

(06) Es wird in die Arbeitsdatei 1 gewechselt.

(07) Der Inhalt der Arbeitsdatei 1 wird gelöscht.

(08) Die EDT-Anweisungen werden in der Arbeitsdatei 1 abgelegt.

(09) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(10) Aufruf der @DO-Prozedur, die im Arbeitsfenster 1 steht (Einlesen einer Datei, Unterscheidung zwischen Klein- und Großbuchstaben, Ausgeben eines Spaltenzählers, Ausgeben einer Informationszeile, Datenfenster und Markierungsspalte auf überschreibbar stellen).

(11) Umschalten in den F-Modus-Dialog. Nach Beenden des Dialogbetriebs mit @HALT oder @RETURN wird der Lauf der Prozedur an der Stelle fortgesetzt, wo er unterbrochen wurde.

(12) Beenden des EDT.

(13) Rücksetzen der Eingabequellen (Wahlweise, wird von END-PROCEDURE rückgesetzt).

(14) Rücksetzen der Auftragsschalter.

5.7 Unbedingter und bedingter Sprung

Mit der @GOTO-Anweisung wird in einer @DO-Prozedur auf eine Zeile verzweigt. Die Zeilennummer wird in @GOTO angegeben. Die Zeile muß existieren und darf nicht außerhalb der Prozedur liegen. Ein Sprung kann nur auf eine Zeile erfolgen, nicht auf eine Marke.

Mit der @IF-Anweisung wird in einer @DO-Prozedur in Abhängigkeit von einer Bedingung verzweigt. Ist die Bedingung erfüllt, so wird auf die Zeile verzweigt, die in der @IF-Anweisung angegeben ist. Ist die Bedingung nicht erfüllt, wird die Prozedur mit der Anweisung, die unmittelbar auf die @IF-Anweisung folgt, fortgesetzt.

Um mögliche Verschiebungen der Zeilen bei Änderungen der Prozedur zu vermeiden, sollten vor den Sprungzielen die Zeilennummern mit @SET, Format 6 (wegen besserer Lesbarkeit abgekürzt mit @) neu bestimmt werden (siehe auch Beispiel).

Innerhalb einer reinen @INPUT-Prozedur sind Sprünge nicht erlaubt.

Beispiel für unbedingte und bedingte Sprünge:

```
 1.00 @PROC 2.....................................................................
 2.00 @DELETE....................................................................
 3.00 @1.00......................................................................
 4.00 @ @CONTINUE *** AB HIER ZEILENNUMMER 1.00 ***.............................
 5.00 @ @CREATE #S1 READ 'BITTE SUCHBEGRIFF EINGEBEN: '........................
 6.00 @ @ON & FIND #S1 MARK 5.....................................................
 7.00  @NOTE *** WENN TREFFER BEI @ON GEFUNDEN GEHEZU ZEILENNUMMER 2.00 ***...
 8.00 @ @IF .TRUE. GOTO 2........................................................
 9.00 @ @CREATE #S2: 'KEIN TREFFER GEFUNDEN'     ..............................
10.00 @ @PRINT #S2.....................................................................
11.00  @NOTE *** WENN KEIN TREFFER GEFUNDEN GEHEZU ZEILENNUMMER 3.00 ***.....
12.00 @ @GOTO 3.....................................................................
13.00 @2.00.....................................................................
14.00 @ @CONTINUE *** AB HIER ZEILENNUMMER 2.00 ***..............................
15.00 @ @DELETE MARK 5.....................................................
16.00 @ @ON & PRINT #S1.....................................................................
17.00 @3.00.....................................................................
18.00 @ @CONTINUE *** AB HIER ZEILENNUMMER 3.00 ***..............................
19.00 @END.....................................................................
20.00 @DO 2.....................................................................
21.00 .....................................................................
```

Nach Eingabe eines Suchbegriffs (Zeile 5.00) wird zuerst geprüft, ob Datensätze mit dem Suchbegriff in der Datei vorhanden sind (Zeile 6.00). Werden keine entsprechenden Datensätze gefunden, wird KEIN TREFFER GEFUNDEN am Bildschirm ausgegeben (Zeile 9.00-10.00) und die Prozedur beendet. Ansonsten werden alle Datensätze, die den Suchbegriff enthalten, am Bildschirm aufgelistet (Zeile 15.00-16.00).

Die Anweisung @1.00 setzt die aktuelle Zeilennummer auf 1 und implizit auch die aktuelle Schrittweite auf 0.01.

5.8 Äußere und innere Schleifen

Mit äußeren Schleifen können Prozeduren mehrmals vollständig durchlaufen werden. Sollen jedoch nur Teile einer Prozedur mehrmals durchlaufen werden, müssen diese Teile in Form von inneren Schleifen formuliert werden.

Äußere Schleifen können durch innere Schleifen ersetzt werden. In einer äußeren Schleife kann nur eine feste Schrittweite, ein beliebiger positiver oder negativer Wert angegeben werden. In einer inneren Schleife kann die Schrittweite variabel, z.B. über eine Zeilennummervariable angegeben werden.

Beispiel für eine äußere Schleife

```
  1.00 @PROC 3.....................................................................
  2.00 @DELETE.....................................................................
  3.00  @NOTE *** FUER ! WIRD DIE JEWEILIGE ZEILENNUMMER EINGESETZT ...........
  4.00  @NOTE     BEHANDELT WERDEN DIE ZEILEN 11,12,13,14 UND 15    ***........
  5.00 @ @COLUMN 10 ON ! INSERT !:27-36:...........................................
  6.00 @END........................................................................
  7.00  @NOTE *** AUFRUF DER PROZEDUR MIT ! ALS SCHLEIFENZAEHLER ..............
  8.00  @NOTE     ANFANGSWERT: 11                                  ..............
  9.00  @NOTE     ENDWERT:     15                                  ..............
 10.00  @NOTE     SCHRITTWEITE: 1                                  ***...........
 11.00 @DO 3, !=11,15.............................................................
 12.00 .....................................................................
```

Die Prozedur ermöglicht ein spaltenweises Kopieren innerhalb einer Datei. Im obigen Beispiel wird nur in den Zeilen 11, 12, 13, 14 und 15 der Inhalt der jeweiligen Zeile von Spalte 27-36 an der Spalte 10 nochmals eingefügt. Das ! wird dabei als Schleifenzähler benutzt (siehe auch @DO).

Um nicht nur die Zeilen 11, 12, 13, 14 und 15 zu kopieren, sondern auch alle Zeilen die dazwischen liegen (also auch z.B die Zeile 12.34) muß die Prozedur mit einer inneren Schleife versehen werden, wenn keine feste Schrittweite gegeben ist (z.B ISAM-Datei, ISAM-Schlüssel als Zeilennummer).

Beispiel für eine innere Schleife

```
 1.00 @PROC 4.....................................................................
 2.00 @DELETE.....................................................................
 3.00 @RESET......................................................................
 4.00  @NOTE *** ANFANGSWERT DER SCHLEIFE AUF 11 SETZEN ***.....................
 5.00 @SET #L10 = 11..............................................................
 6.00 @1.00..................................................................... ...
 7.00  @NOTE *** WENN FEHLER SCHLEIFE ABBRECHEN ***.............................
 8.00 @ @IF ERRORS : @GOTO 2.....................................................
 9.00  @NOTE *** WENN ENDWERT 15 ERREICHT SCHLEIFE ABBRECHEN ***.................
10.00 @ @IF #L10 > 15 GOTO 2.....................................................
11.00  @NOTE *** FUER #L10 WIRD DIE JEWEILIGE ZEILENNUMMER EINGESETZT ***.......
12.00 @ @COLUMN 10 ON #L10 INSERT #L10:27-36:....................................
13.00  @NOTE *** SCHLEIFENZAEHLER AUF NAECHSTE VORHANDENE ZEILE ERHOEHEN ***....
14.00 @ @SET #L10 = #L10 + 1L.....................................................
15.00 @ @GOTO 1.....................................................................
16.00 @2.00.....................................................................
17.00 @ @CONTINUE.....................................................................
18.00 @END.....................................................................
19.00 @DO 4.....................................................................
20.00 .....................................................................
```

Beginnend bei Zeile 11 wird in allen Zeilen bis einschließlich Zeile 15 einer Datei (also z.B. auch in Zeile 13.314) der Inhalt der jeweiligen Zeile von Spalte 27-36 an der Spalte 10 nochmals eingefügt. Schleifenzähler ist hier die Zeilennummervariable #L10.

Ändert man in der Prozedur die Zeile 11.00 in @ @SET #L10 + 1, werden nur die Zeilen 11, 12, 13, 14 und 15 verändert (Effekt siehe auch vorheriges Beispiel).

Zeile 11.00 muß in der zu bearbeitenden Datei existieren und die letzte Zeile der zu bearbeitenden Datei sollte größer sein als 15.00, ansonsten wird die Prozedur mit einer Fehlermeldung abgebrochen.

5.9 Variable EDT-Prozeduren - Parameter

Bei der Erstellung von Prozeduren im EDT können über @PARAMS Parameter definiert werden. Mit Parametern können verschiedene Werte beim Aufruf mit @DO an eine Prozedur übergeben werden.

Die @PARAMS-Anweisung muß als erste Anweisung in einer @DO-Prozedur stehen und darf nur einmal in der Prozedur vorkommen. Genau wie im BS2000-System sind Stellungs- und Schlüsselwortparameter erlaubt. Alle Stellungsparameter müssen vor den Schlüsselwortparametern definiert werden.

Beim Aufruf der Prozedur werden die Parameter in der @DO-Anweisung als Aktualparameter angegeben. Beim Ablauf der Prozedur werden die Formalparameter innerhalb der Prozedur mit den Werten dieser Aktualparameter versorgt.

Ein Parameter beginnt mit dem Zeichen &. Ihm folgt ein Buchstabe, dem bis zu 6 weitere Buchstaben oder Ziffern folgen können.

Beispiel für die Verwendung von Parametern in einer EDT-Prozedur

Im folgenden Beispiel wird eine Datei in die Arbeitsdatei 0 eingelesen. Die Datensätze, die den Suchbegriff enthalten, werden in die Arbeitsdatei 5 kopiert, entsprechend aufbereitet und auf dem Bildschirm ausgegeben.

```
 1.     @PROC 4
 1.     @DELETE
 1.     @ @PARAMS &DATEI,&SUCH ---------------------------------------- (01)
 2.     @ @DELETE
 3.     @ @READ '&DATEI'
 4.     @ @ON & FIND PATTERN '&SUCH' COPY TO (5)
 5.     @ @PROC 5
 6.     @ @CREATE 0.01: '-' * 68
 7.     @ @CREATE 0.02: 'MANUAL-LISTE FUER:','&SUCH'
 8.     @ @CREATE 0.03: '-' * 68
 9.     @ @RENUMBER
10.     @ @PREFIX 4-.$ WITH '      '
11.     @ @SEQUENCE 4-.$:1:0001(1)
12.     @ @CREATE $+1: '-' * 68
13.     @ @PRINT
14.     @ @END
15.     @END
 1.     @DO 4 (MANUALDATEI,EDT) ---------------------------------------- (02)
 1.0000 ------------------------------------------------------------------
 2.0000 MANUAL-LISTE FUER: EDT
 3.0000 ------------------------------------------------------------------
 4.0000 0001  EDT  V16.3A      ANWEISUNGEN                   U1884-J-Z125-5
 5.0000 0002  EDT  V16.3A      ANWEISUNGSFORMATE             U1978-J-Z125-4
 6.0000 0003  EDT  V16.3A      UNTERPROGRAMMSCHNITTSTELLEN   U5133-J-Z125-1
```

```
 7.0000 0004  EDT  V16.4A      ANWEISUNGEN                    U1884-J-Z125-6
 8.0000 0005  EDT  V16.4A      ANWEISUNGSFORMATE              U1978-J-Z125-5
 9.0000 0006  EDT  V16.4A      UNTERPROGRAMMSCHNITTSTELLEN    U5133-J-Z125-2
 9.0000 0007  EDT  V16.4A      EDT-OPERANDEN                  U20207-J-Z125-1
 7.0000 0008  EDT  V16.5A      ANWEISUNGEN                    U1884-J-Z125-7
 8.0000 0009  EDT  V16.5A      ANWEISUNGSFORMATE              U1978-J-Z125-6
 9.0000 0010  EDT  V16.5A      UNTERPROGRAMMSCHNITTSTELLEN    U5133-J-Z125-3
10.0000 0011  EDT  V16.5A      EDT-OPERANDEN                  U20207-J-Z125-2
11.0000 ------------------------------------------------------------------
 1.     @DO 4 (MANUALDATEI,EDT*V16.4*) ---------------------------------- (02)
 1.0000 ------------------------------------------------------------------
 2.0000 MANUAL-LISTE FUER: EDT*V16.5*
 3.0000 ------------------------------------------------------------------
 4.0000 0001  EDT  V16.5A      ANWEISUNGEN                    U1884-J-Z125-7
 5.0000 0002  EDT  V16.5A      ANWEISUNGSFORMATE              U1978-J-Z125-6
 6.0000 0003  EDT  V16.5A      UNTERPROGRAMMSCHNITTSTELLEN    U5133-J-Z125-3
 7.0000 0004  EDT  V16.5A      EDT-OPERANDEN                  U20207-J-Z125-2
 8.0000 ------------------------------------------------------------------
```

(01) Definieren der symbolischen Parameter (zwei Stellungsparameter).

(02) Aufruf der Prozedur mit den jeweilgen Aktualparametern. Die Formalparameter in der @READ-, @ON-Anweisung werden bei jedem @DO-Aufruf durch die aktuellen Werte ersetzt.

Weitere Beispiele siehe auch @PARAMS.

BS2000-Systemprozedur-Parameter

```
/BEGIN-PROCEDURE
/  PARAMETER=YES(PROCEDURE-PARAMETERS=(-
/  &DATEI, -
/  &ANFSPAL, -
/  &ENDSPAL, -                                              ---------------- (01)
/  &ZIELSPAL,-
/  &VONZEILE=%, -
/  &BISZEILE=$),-
/  ESCAPE-CHARACTER='&')
/SHOW-FILE-ATTRIBUT &DATEI
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/MODIFY-JOB-SWITCHES ON=(4,5)
/START-PROGRAM $EDT ------------------------------------------------------ (02)
@READ '&DATEI' ----------------------------------------------------------- (03)
@PROC 4
@DELETE
 @NOTE SPALTENWEISE EINFUEGEN                               ---------------- (04)
@ @COLUMN &ZIELSPAL ON ! INSERT !:&ANFSPAL-&ENDSPAL:
@END
@DO 4, !=&VONZEILE,&BISZEILE --------------------------------------------- (05)
@WRITE '&DATEI.NEU' ------------------------------------------------------ (06)
@HALT -------------------------------------------------------------------- (07)
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/MODIFY-JOB-SWITCHES OFF=(4,5)
/SET-JOB-STEP
/END-PROCEDURE
```

(01) Symbolische Parameter vereinbaren.

(02) EDT aufrufen.

(03) Datei ’&datei’ in die Arbeitsdatei 0 einlesen.

(04) EDT-Prozedur:
- In die Arbeitsdatei 4 wechseln.
- Arbeitsdatei 4 löschen.
- @COLUMN-Anweisung mit den aktuellen Parameterwerten in die Arbeitsdatei 4 schreiben.
- Zurück in die Arbeitsdatei 0 wechseln.

(05) Aufruf der EDT-Prozedur:
In der Datei ’&datei’ werden für den angegebenen Zeilenbereich ’&vonzeile’ bis ’&biszeile’ jeweils die Spalten ’&anfspalte’ bis ’&endspalte’ an der Spalte ’&zielspalte’ eingefügt.

(06) Speichern der geänderten Datei unter dem Namen ’&datei.NEU’.

(07) EDT beenden.

6 Anweisungen des EDT

In diesem Kapitel werden alle Anweisungen beschrieben, die im F-Modus in der EDT-Anweisungszeile und/oder im L-Modus angegeben werden dürfen. Die EDT-Kurzanweisungen in der Markierungsspalte sind im Abschnitt „F-Modus“ auf Seite 63ff. beschrieben .

6.1 Beschreibung der Syntax

Allgemeines Format einer Anweisung:

Operation	Operanden	F-Modus / L-Modus / @PROC
Operation	{ Operanden &str-var }	

Operation — Die Operation entspricht dem Anweisungsnamen, z.B. @OPEN, @COPY, @WRITE. Sie muß am Anfang der Anweisung stehen. Im F-Modus kann das EDT-Anweisungssymbol (standardmäßig @) auch weggelassen werden.

Operanden — Der Operation folgen - durch ein oder mehrere Leerzeichen getrennt - die Operanden. Die Operanden sind in der vorgegebenen Reihenfolge anzugeben. Vor bzw. nach jedem Operanden können beliebig viele Leerzeichen eingegeben werden.

str-var — Zeichenfolgevariable, die die Operanden enthält (indirekte Angabe). Näheres zur indirekten Angabe der Operanden siehe Abschnitt „Eingeben von Anweisungen“ auf Seite 37.

Das Trennzeichen (Leerzeichen) zwischen Operation und Operanden, bzw. zwischen den einzelnen Operanden muß dann angegeben werden, wenn Operation und Operand bzw. zwei aufeinanderfolgende Operanden nicht unterscheidbar sind (Beispiel: @SYMBOLS=’?’ ist falsch; richtig ist @SYMBOL S=’?’).

Für die formale Darstellung der Anweisungen wird folgende Metasyntax verwendet:

Formale Darstellung	**Erläuterung**	**Beispiele**
GROSSBUCHSTABEN und Sonderzeichen	Großbuchstaben und Sonderzeichen beeichnen Konstanten, die der Benutzern in dieser Form eingeben muß.	UPDATE, OVERWRITE
GROSSBUCHSTABEN in Halbfett	Halbfette Großbuchstaben kennzeichnen die Kurzform der Konstanten. Zuläßig sind alle Eingaben zwischen Kurzform und Langform der Konstanten.	@**H**ALT Einzugeben ist: @H, @HA, @HAL oder @HALT
kleinbuchstaben	Kleinbuchstaben bezeichnen Variable, die der Benutzer bei der Eingabe durch aktuelle Werte ersetzen muß.	@CODE ln, SHOW Einzugeben ist: @CODE 3,SHOW
{ }	Geschweifte Klammern schließen Alternativen ein, d.h. eine der Angaben muß ausgewählt werden..	@LOWER { ON / OFF } Einzugeben ist: @LOWER ON oder @LOWER OFF
|	| trennt Alternativen (wie geschweifte Klammern)	
[ ]	Angaben in eckigen Klammern sind optional; sie können wahlweise angegeben werden	
....	3 Punkte bedeuten eine Wiederholung; die davor stehende Einheit kann mehrmals hintereinander wiederholt werden.	ln,... Einzugeben ist: 1,3,7
<u>Unterstreichung</u>	Wert, der vom EDT beim Aufruf eingestellt wird.	

Beispiel

Die Anweisungen werden folgendermaßen dargestellt.

Operation	Operanden	F-Modus / L-Modus / @PROC
@**C**OPY	rng [(procnr)] [**T**O ln1 [(inc)] [:] [ln2]] [,...]	

Es bedeutet:

F-Modus Die Anweisung kann im F-Modus angegeben werden.

L-Modus Die Anweisung kann im L-Modus angegeben werden.

@PROC Die Anweisung kann nur in EDT-Prozeduren angegeben werden bzw. sie wird hauptsächlich dort verwendet.

Im L-Modus muß man der Anweisung das EDT-Anweisungssymbol voranstellen. Der EDT unterscheidet bei der Eingabe so die Anweisungen von den Daten. Das Anweisungssymbol ist nach Aufruf des EDT mit @ voreingestellt. Es kann auch ein anderes Zeichen vereinbart werden (siehe @:).

In der Anweisungszeile des F-Modus kann das Anweisungssymbol weggelassen werden. Es ist zur Unterscheidung zwischen Anweisungen und Daten nicht notwendig, da in der Anweisungszeile nur Anweisungen eingegeben werden dürfen.

6.2 Beschreibung der Operanden

Das Zeichen „|“ in der Definition trennt Alternativen.

Definition	**Bedeutung**
binary::=0|1	Binärzahl.
char	Jedes Zeichen, das an einer Datenstation oder über Kartenleser eingegeben werden kann.
chars::=char|chars char	Zeichenfolge.
cl::=int	Spaltennummer, die zwischen 1 und 256 liegen muß. Einige EDT-Anweisungen verlangen jedoch zwingend einen cl-Wert, der unter 256 liegt.
clrng::=cl[-cl]	Spaltenbereich vom Satzende in Richtung Satzanfang
col::=domain|col,domain| cl|col,cl	Mehrere Spaltenbereiche oder Spalten, die durch Komma voneinander getrennt werden, z.B. 1-6, 8-11 oder 1-6, 3-10, 3-10.
comment::=chars	Beliebiger Kommentar.
dd::=0|1|2|3|4|5|6|7|8|9	Dezimalziffer.
domain::=cl[-cl]	Spaltenbereich der Form cl-cl, wodurch ein vorgegebener Zeilenbereich auf einen bestimmten Spaltenbereich eingegrenzt wird. Dieser beginnt beim 1. cl. Das 2. cl darf nicht kleiner als das 1. sein. Wird es nicht angegeben, so beginnt der Spaltenbereich beim 1. cl und endet am jeweiligen Zeilenende. Ist das 2. cl angegeben und größer als die Zeilenlänge, so erstreckt sich der Spaltenbereich bis an das Ende solcher Zeilen. Ist bereits das 1. cl größer als die Zeilenlänge, so wird die zugehörige Zeile ignoriert.
edtsymb::=spec	Anweisungsymbol, das standardmäßig das Zeichen @ ist; es kann über @: verändert werden.
entry::=name|.str-var	Name einer Einsprungstelle (ENTRY) oder einer CSECT-Anweisung in einem Programm.
elemname::=chars|.str-var	Name des Bibliothekselements.
elemtyp::=S|M|R|C|P|J|D|X| H|L|U|F|*STD| freetyp|.str-var	Typ des Bibliothekselements. Zur Angabe des Elementyps sind freie Typnamen zugelassen. Es erfolgt keine Prüfung auf den Basistyp.

Definition	Bedeutung
file::=str	Dateiname, der über abdruckbare, hexadezimale oder binäre Darstellung vereinbart werden kann. Der Dateiname darf aus max. 54 Zeichen bestehen, wobei die DVS-Restriktionen für Dateinamen zu berücksichtigen sind. Der EDT überprüft nicht, ob der Dateiname gültig ist. Somit führt die Verwendung eines nicht zulässigen Dateinamens zu DVS-Fehlermeldungen. Statt des Dateinamens kann für file auch der komplette Pfadname angegeben werden. Die Verwendung des speziellen Dateinamens '/' (Dateikettung) ist bei @OPEN nicht möglich.
formal::=&id	Formalparameter der Form &id, der in der @PARAMS-Anweisung einer Prozedurdatei anzugeben ist. id ist der Name, der aus bis zu 7 Buchstaben oder Ziffern bestehen kann. Das 1. Zeichen muß ein Buchstabe sein. Dieser Operand findet in Schlüsselwort- und Formalparametern Verwendung.
fraction::=.dd|fraction dd	Der hinter dem Dezimalpunkt stehende Teil einer Zeilennummer (also .0001 bis .9999).
freetyp::=chars	Freier Typname eines Bibiliothekselements. Zeichenfolge der Länge 2 bis 8 Zeichen, die nicht mit $ oder SYS beginnen darf.
fwkfnr::=dd	Nummer der Arbeitsdatei. Mindestwert ist 0, Höchstwert ist 9.
fwkfv::=$dd	Nummer der Arbeitsdateivariablen $0 bis $9.
hd::=dd|A|B|C|D|E|F	Hexadezimalziffer.
hex::=hd|hex hd	Hexadezimaler Ausdruck.
hpos::=op n|vpos-op| vpos-op (m,...)	Relative vertikale Position.
hpos-op::=>|<|hpos-op n	Vertikale Positionieranweisung.
inc::=n|fraction|n fraction	Zeilennummer bzw. Schrittweite für Zeilennummern, die zwischen 0.0001 und 9999.9999 liegen muß.
int::=n|op n|int-var	Ganzzahl, die entweder explizit oder über Ganzzahlvariablen angegeben wird, z.B.: -5,0,23456,... oder #I0,#I1,... #I20.

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>int-var::=#ln</td><td>Eine der Ganzzahlvariablen #I0, #I1, ..., #I20. Diese müssen kleiner als 2**31 (=2147483648) sein. Die zugelassenen Grenzen hängen jedoch von der Anwendung ab. Alle Ganzzahlvariablen sind vorbesetzt mit dem Wert 0. Die Ganzzahlvariablen #I0 und #I1 werden von einer @ON- Anweisung verändert.</td></tr>
<tr><td>linkname::=chars</td><td>Spezifiziert eine Datei oder Jobvariable über ihren Kettungsnamen. Bei linkname sind keine Jokerzeichen erlaubt.</td></tr>
<tr><td>line::=ln|str-ln</td><td>Zeilennummer, die entweder wie der Operand ln oder wie str-ln angegeben werden kann.</td></tr>
<tr><td>ln::=

ln-sym[op int-var]|</td><td>Mit dem Operanden ln können Zeilennummern direkt oder wie im folgenden dargestellt symbolisch angegeben werden:
Darstellung 1:
<table>
<tr><th>ln-sym</th><th>op</th><th>int-var</th></tr>
<tr><td>%
*
$
?
ln-var</td><td>+ / –</td><td>#I0...#I20</td></tr>
</table>
Die Symbole %, *, $, ? sind bei ln-sym erklärt.
Beispiel: % + #I15. Gilt etwa %=1.0000 und #I15=6, so wird mit % + #I15 die 6. Zeile hinter Zeilennummer 1.0000 angesprochen. (Dies muß nicht unbedingt Zeile 7.0000 sein.)</td></tr>
</table>

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>ln::=(Fortsetzung)

ln-sym[op nL]|</td><td>Darstellung 2:

<table>
<tr><th>ln-sym</th><th>op</th><th>nL</th></tr>
<tr><td>%
*
$
?
ln-var</td><td>+ / –</td><td>1L...nL</td></tr>
</table>
Mit nL werden - von ln-sym beginnend - n Zeilen übersprungen. Der Maximalwert für n hängt damit von der Anzahl der Zeilen ab.
Beispiel: Ist % = 1.000, so spricht man mit
% + 2L die 2. Zeile hinter 1.0000 an.</td></tr>
<tr><td>ln-sym[op ln-sym]|</td><td>Darstellung 3:

<table>
<tr><th>ln-sym</th><th>op</th><th>ln-sym</th></tr>
<tr><td>%
*
$
?
ln-var</td><td>+ / –</td><td>%
*
$
?
ln-var</td></tr>
</table>
Beispiel: Ist % = 1.0000 und * = 3.0000, so spricht man mit % + * die Zeile 4.0000 an.</td></tr>
<tr><td>[ln-sym op]inc[op ln-sym]|</td><td>Darstellung 4:

<table>
<tr><th>ln-sym</th><th>op</th><th>inc</th><th>op</th><th>ln-sym</th></tr>
<tr><td>%
*
$
?
ln-var</td><td>+ / –</td><td>0000.0001
.
.
.
9999.9999</td><td>+ / –</td><td>%
*
$
?
ln-var</td></tr>
</table>
Beispiel: Ist * = 50.1 und % = 1.0000, so spricht man mit * + 3.5-% die Zeile 52.6000 an.</td></tr>
</table>

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>ln::=(Fortsetzung)

[ln-sym op]inc[op int-var]|</td><td>Darstellung 5:

<table>
<tr><th>ln-sym</th><th>op</th><th>inc</th><th>op</th><th>ln-sym</th></tr>
<tr><td>%
*
$
?
ln-var</td><td>+ / –</td><td>0000.0001
.
.
.
9999.9999</td><td>+ / –</td><td>#I0...#I20</td></tr>
</table>
Beispiel: Ist * = 50.1 und #I5 = 1, so spricht man mit * + 3.5 + #I5 die Zeile an, die der Zeile 53.6000 folgt. (Dies muß nicht unbedingt die Zeile 53.7000 sein.)</td></tr>
<tr><td>[ln-sym op]inc[op nL]</td><td>Darstellung 6:

<table>
<tr><th>ln-sym</th><th>op</th><th>inc</th><th>op</th><th>nL</th></tr>
<tr><td>%
*
$
?
ln-var</td><td>+ / –</td><td>0000.0001
.
.
.
9999.9999</td><td>+ / –</td><td>1L...nL</td></tr>
</table>
Mit nL kann man n Zeilen überspringen. Damit hängt das Maximum von n von der Anzahl der Zeilen ab.
Beispiel: Ist * = 50.1000, so spricht man mit * + 3.5 + 6L die 6. Zeile hinter 53.6000 an.

Diese Darstellungen kann man in 2 Klassen einteilen:

1. Absolute Zeilennummern
Hiermit sind alle Zeilennummern angesprochen, die weder über int-var noch über nL angegeben werden. Diese absoluten Zeilennummern ergeben sich somit durch Summieren der einzelnen Zeilennummern. Gilt etwa % = 1.0000 und * = 3.0000, so erhält die Zeilennummer % + 2.1 + * den Wert 6.1000.</td></tr>
</table>

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>ln::=(Fortsetzung)</td><td>2. Relative Zeilennummern
Relative Zeilennummern sind solche, die man durch Überspringen von gewissen Zeilen erhält, unabhängig davon, welche Schrittweite zwischen den einzelnen übersprungenen Zeilen liegt. Gilt etwa % = 1.0000, so wird mit % + 2.1 + 5L die 5. Zeile hinter der Zeile 3.1000 angesprochen.

Im Ausdruck nL darf n nicht den Wert 0 annehmen. Jedoch ist es möglich, in einer Ganzzahlvariablen diesen Wert abzulegen. Die relative Zeilennummer kann man nur dann zuweisen, wenn es eine solche Zeile auch gibt. Andernfalls wird eine Fehlermeldung ausgegeben.</td></tr>
<tr><td>ln-sym::=ln-var|%|*|$|?</td><td>Zeilennummervariable oder eine der folgenden symbolischen Zeilennummern:

<table><tr><th>Symbol</th><th>Erläuterung</th></tr><tr><td>*</td><td>Aktuelle Zeilennummer, also die Zeilennummer, die der EDT zuletzt als Quittung auf die Datenstation geschrieben hat.</td></tr><tr><td>%</td><td>Niedrigste Zeilennummer der Datei.</td></tr><tr><td>$</td><td>Höchste Zeilennummer der Datei. Ist die Datei leer oder enthält sie nur eine einzige Zeile, so ist % = $.</td></tr><tr><td>?</td><td>Zeilennummer der 1. Trefferzeile einer vorangegangenen @ON-Anweisung. Ausgangswert ist 0.0001. Dieser wird lediglich durch eine erfolgreiche @ON-Anweisung verändert. ? hat also nach @ON den gleichen Wert wie #L00.</td></tr></table>
*,%,$,? beziehen sich auf die aktuelle Arbeitsdatei.</td></tr>
<tr><td>ln-var::=#Ln</td><td>Eine der Zeilennummervariablen #L0, #L1, ..., #L20. Der kleinste Inhalt einer Zeilennummernvariablen ist 0.0001, der größte 9999.9999. Der Ausgangswert aller Zeilennummervariablen ist jedoch 0. Dies ist ein ungültiger Wert. Demnach müssen den Zeilennummervariablen vor ihrer Benutzung zugelassene Werte zugewiesen werden. Die Zeilennummervariable #L0 wird von einer @ON-Anweisung verändert.</td></tr>
</table>

Definition	Bedeutung
m::=dd|int-var	Satzmarkierung 1 bis 9.
message::=chars	Beliebiger Text, der bei Aufruf des EDT als Unterprogramm mit @RETURN oder @HALT an das rufende Programm übergeben wird.
modlib::=path|.str-var	Modulbibliothek oder Programm-Bibliothek, in der sich ein Modul oder eine Ladeeinheit befindet.
n::=dd|n dd	Eine der Ziffern 0,1,2,3,4,5,6,7,8,9 oder eine Kombination dieser Ziffern. Die Anzahl der Ziffern hängt von der jeweiligen Anweisung ab. Daher muß nicht unbedingt 00005 gleichwertig mit 5 sein.
name::=chars	Zeichenfolge mit maximaler Länge von 8 Zeichen.
op::=+|–	Eine der mathematischen Funktionen + oder –. Diese finden Verwendung im Zusammenhang mit den Operanden int, str-ln und ln.
param::= '[jede Zeichenfolge]'| jede Zeichenfolge	Über @DO können Parameter an eine auszuführende Prozedurdatei übergeben werden. Diese bestehen aus irgendeiner Zeichenfolge, die man dann in Hochkomma einzuschließen hat, wenn man ein Komma oder eine rechte runde Klammer als Bestandteil der Parameterfolge übergeben will. In diesem Fall ist jedes zu übergebende Hochkomma im Parameterausdruck durch 2 zu schreibende Hochkommas zu kennzeichnen. Die eingrenzenden Hochkommas können durch @QUOTE nicht redefiniert werden.
path::=chars|.str-var	Pfadname einer Datei bzw. einer Jobvariablen. path darf aus max. 54 Zeichen bestehen, wobei die DVS-Restriktionen für Dateinamen zu berücksichtigen sind. Eine Datei bzw. eine Jobvariable kann auch teilqualifiziert angegeben werden.
procnr::=int	Nummer der Arbeitsdatei. Mindestwert ist 1, Höchstwert ist 22. Ausnahme: Bei @COMPARE, @SETF, @COPY, @MOVE, @STATUS und @ON ist 0 zugelassen.
r::=spec	Bereichssymbol, das standardmäßig das Zeichen & ist; es kann über @RANGE verändert werden.
range::=rng|range,rng	Ein oder mehrere Zeilenbereiche, die durch Komma voneinander zu trennen sind.
range*::=rng*|range*,rng*	Analog dem Operanden range; jedoch dürfen keine Zeichenfolgevariablen angegeben werden.

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>rel::=GT|LT|GE|LE|EQ|NE|
>|<|>=|<=|=|<></td><td>Zeichen für eine Relation, die über die @IF-Anweisung abgefragt werden kann.
Erlaubte Zeichen sind:

<table><tr><th colspan="3">Symbol</th><th>Erläuterung</th></tr><tr><td>GT</td><td>bzw.</td><td>></td><td>größer als</td></tr><tr><td>LT</td><td>bzw.</td><td><</td><td>kleiner als</td></tr><tr><td>GE</td><td>bzw.</td><td>>=</td><td>größer oder gleich</td></tr><tr><td>LE</td><td>bzw.</td><td><=</td><td>kleiner oder gleich</td></tr><tr><td>EQ</td><td>bzw.</td><td>=</td><td>gleich</td></tr><tr><td>NE</td><td>bzw.</td><td><></td><td>ungleich</td></tr></table></td></tr>
<tr><td>rng::=line[-line]|r</td><td>Zeilenbereich der Form line-line. Die Angabe line1-line2 (z.B. 1-10) bewirkt das gleiche wie line2-line1 (10-1).
Wird nur ein line Operand angegeben, so besteht der Zeilenbereich nur aus dieser einen Zeile.
Werden beide line-Operanden angegeben und ist der 1. davon eine Zeichenfolgevariable, so muß auch der 2. eine Zeichenfolgevariable sein.
Der oben aufgeführte Operand r stellt einen Zeilenbereich dar, der über die @RANGE-Anweisung vereinbart werden kann und mit 0.0001-9999.9999 vorbelegt ist.
Über die @RANGE-Anweisung können sowohl das Bereichssymbol als auch der Zeilenbereich verändert werden.
Der gewünschte Zeilenbereich kann sich auch über Zeichenfolgevariablen erstrecken. Vorsicht ist geboten bei Verwendung des Zeichens - und der Operanden ln-sym bzw. str-var. Da das Minuszeichen auch zum Ausdrücken eines Zeilenbereiches benutzt wird, kann eine Mehrgültigkeit ent stehen.
Um dieses Problem zu vermeiden, wurden nachfolgende Konventionen eingeführt.

1. Beginnt ein Bereich mit ln-sym und endet er mit ln (egal welche Form), so schreibt man:
ln-sym**.**-ln</td></tr>
</table>

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>rng::=(Fortsetzung)</td><td>
2. Beginnt ein Bereich mit ln (außer ln-sym) und endet er bei einer Zeilennummer, die über die Operanden ln-sym und op ausgedrückt wird, so empfiehlt sich die Schreibweise:
ln - 0 ln-sym [op int-var]
ln - 0 ln-sym [op nL]
ln - 0 ln-sym [op ln-sym]
ln - [0 ln sym op] inc [op ln-sym]
ln - [0 ln-sym op] inc [op int-var]
ln - [0 ln sym op] inc [op nL]
Mit Angabe von 0 wird zum Ausdruck gebracht, daß es sich um einen Bereich, nicht etwa um eine Differenz handelt. 0 kann auch durch . (Punkt) ersetzt werden.
3. Beginnt ein Bereich mit einer Zeichenfolgevariablen und endet er mit str-ln (egal welche Form), so schreibt man:
str-var[-0 str-var][op int-var]
str-var[-0 str-var][op nL]
Beispiele:
<table>
<tr><th>rng</th><th>Bedeutung</th></tr>
<tr><td>1-10</td><td>Zeilen 1 bis 10</td></tr>
<tr><td>%.-5</td><td>1. Zeile der Datei bis Zeile 5</td></tr>
<tr><td>%+5L-$-10L</td><td>6. Zeile der Datei bis zur 10. Zeile vor Ende der Datei</td></tr>
<tr><td>%.-$</td><td>Gesamte Datei</td></tr>
<tr><td>*+2.1-?-%+5L</td><td>Von *+2.1-? bis zur 6. Zeile der Datei</td></tr>
<tr><td>#L1.-#L2</td><td>Von #L1 bis #L2</td></tr>
<tr><td>12.011</td><td>Lediglich Zeile 12.011</td></tr>
<tr><td>#L9</td><td>Lediglich die Zeile, derenWert in #L9 abgelegt ist</td></tr>
<tr><td>#S3</td><td>Lediglich die Zeichenfolge-variable #S3</td></tr>
<tr><td>#S4.-#S7</td><td>#S4, #S5, #S6, #S7</td></tr>
<tr><td>#S2+1L-#S6-#I3</td><td>#S3,..., #S6-#I3</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>rng*::=ln[-ln]|r</td><td>Spezieller Zeilenbereich, der über r oder ln - ln ausgedrückt wird. Im Unterschied zu rng dürfen hier keine Zeichenfolgevariablen angegeben werden. Deshalb findet anstelle von line der oben aufgeführte Operand ln Verwendung.</td></tr>
<tr><td>search::=strng|line[:col:]</td><td>Suchbegriff.</td></tr>
<tr><td>spec</td><td>Sonderzeichen.</td></tr>
<tr><td>str::=
'jede Zeichenfolge'[*int]|
B'binary'[*int]|
X'hex'[*int]</td><td>3 Fälle sind möglich:
1. 'Irgendeine Zeichenfolge' [*int]
2. B'Binärzahl' [*int]
3. X'Hexadezimalzahl' [*int]
Wird B oder X verwendet, so muß dieser Buchstabe unmittelbar vor der Binär- bzw. Hexadezimalzahl stehen, die in Hochkomma eingeschlossen ist.
Soll im 1. Fall die Zeichenfolge ein solches Hochkomma enthalten, so müssen hierfür 2 Hochkommas geschrieben werden. Das gültige Zeichen für Hochkomma kann über @QUOTE verändert werden.
Die wahlfreie Angabe von *int ist dafür vorgesehen, eine Zeichenfolge zu wiederholen. Z.B. ist 'ab'*3 gleichwertig mit 'ababab'. Da die max. Länge einer Zeichenfolge 256 ist, darf int diesen Wert nicht überschreiten. Hat int den Wert 0 oder die zu wiederholende Zeichenfolge die Länge 0, so hat die daraus resultierende Zeichenfolge die Länge 0.
Beispiele:
<table><tr><th>str</th><th>erzeugt Zeichenfolge</th></tr><tr><td>'A''BC''D'</td><td>A'BC'D</td></tr><tr><td>'ABC' *5</td><td>ABCABCABCABCABC</td></tr><tr><td>X'C1F2' *4</td><td>A2A2A2A2</td></tr><tr><td>B'11110000' *3</td><td>000</td></tr></table>
Bemerkung: Wird bei hexadezimaler Angabe eine ungerade Anzahl von Zeichen verwendet, so wird linksbündig mit Nullen aufgefüllt. Z.B. ist X'F' gleichwertig mit X'0F' oder X'A' *4 gleichwertig mit X'0A' *4.
Analoges gilt bei Binärdarstellung, wenn die Anzahl der Binärzeichen nicht ein Vielfaches von 8 ist. Auch hier wird linksbündig mit Nullen aufgefüllt, bis die Anzahl der Binärzeichen ein Vielfaches von 8 ist. Z.B. ist B'1' gleichwertig mit B'00000001' oder B'1111'*2 gleichwertig mit B'00001111'*2.</td></tr>
</table>

Definition	Bedeutung
string::=str|line[:col:]	Entweder der Operand str oder line [:col:]. Wird hierfür eine Zeichenfolgevariable oder eine Zeilennummer angegeben, so verwendet der EDT als string den Inhalt der Zeichenfolgevariablen bzw. Inhalt der entsprechenden Zeile. Falls eine solche Zeile nicht existiert, wird eine Fehlermeldung ausgegeben und die Anweisung abgelehnt. Ist als string lediglich ein Teil einer Zeile gewünscht, so kann dies über entsprechende Spaltenangaben zum Ausdruck gebracht werden. Werden Spaltenwerte angegeben, die die Zeilenlänge überschreiten, so werden dafür Leerzeichen eingesetzt. Enthält z.B. die Zeile 6 die Zeichenfolge AB3CD6EF9 und ist string angegeben als 6:1-3,9,8,9,8-9,5-7,30,1,30,1:, so ist die hierüber ausgedrückte Zeichenfolge AB39F9F9D6E A A Wird von dieser Möglichkeit der Spaltenangabe in einer der Anweisungen @INPUT, @GET, @SAVE, @READ oder @WRITE Gebrauch gemacht, so gilt das Obige für die Zeilen, die von der Platte zu lesen bzw. dorthin zu schreiben sind. Wird als string eine Zeilennummer angegeben, die über eine EDT-Anweisung mit diesem Operanden string verändert werden soll, so ist diesem Operanden der ursprüngliche Inhalt der Zeile zugeordnet. Es habe etwa Zeile 1 den Inhalt ABC und die EDT-Anweisung lautet: @CREATE1:1,'D'*3,1 Dann hat die Zeile 1 nach Durchführung dieser Anweisung den Inhalt ABCDDDABC

<table>
<tr><th>Definition</th><th>Bedeutung</th></tr>
<tr><td>str-ln::=
str-var[op int-var]|
str-var[op nL]</td><td>Besondere Form, eine Zeichenfolgevariable auszudrücken. Zwei Möglichkeiten gibt es:

Darstellung 1:

<table>
<tr><th>str-var</th><th>op</th><th>intvar</th></tr>
<tr><td>#Sn1</td><td>+ / –</td><td>#In2</td></tr>
</table>

Beispiel: Ist #I10=5, so ist #S0+#I10=#S5 oder #S13+#I10=#S18.

Darstellung 2:

<table>
<tr><th>str-var</th><th>op</th><th>nL</th></tr>
<tr><td>#Sn</td><td>+ / –</td><td>1L...nL</td></tr>
</table>

Beispiel: #S15-5L=#S10 oder #S3+8L=#S11.</td></tr>
</table>

Definition	Bedeutung

strng::=
'jede zeichenfolge'[*int]|
B'binary'[*int|
X'hex'[*int]
	Analog dem Operanden str, jedoch kann anstelle eines einfachen Hochkommas (') auch ein Anführungszeichen (") verwendet werden. Die Verwendung dieses Operanden ist lediglich in der @ON-Anweisung im 1. Operanden (search) erlaubt. Soll ein Hochkomma oder ein Anführungszeichen Bestandteil einer Suchzeichenfolge sein, so ist dieses Zeichen 2mal anzugeben. Auch darf - in Analogie zum Operanden str - ein Faktor für Vervielfachung (*int) angegeben werden. Die Suchzeichenfolge darf aber nicht länger als 256 Zeichen werden.
	Eine Suchzeichenfolge ist sowohl links als auch rechts begrenzt von einem Hochkomma oder einem Anführungszeichen. Beide können über @QUOTE redefiniert werden. Wird eine Suchzeichenfolge links oder rechts von einem Anführungszeichen begrenzt, so bringt man dadurch zum Ausdruck, daß links bzw. rechts von dieser Suchzeichenfolge ein Textbegrenzer stehen muß.
	Der Satz der Textbegrenzerzeichen ist durch den EDT vorbelegt mit den Zeichen +.!*();-/,?:'=" und dem Leerzeichen (X'40'). Dieser Zeichensatz kann über die @DELIMIT-Anweisung verändert werden. Per Definition steht vor dem 1. und hinter dem letzten Zeichen immer ein Textbegrenzer. Für das folgende Beispiel sei angenommen, daß über @DELIMIT als Textbegrenzerzeichen das Komma und das Leerzeichen vereinbart wurden. Über die @ON-Anweisung soll in einer Zeile mit dem nachfolgenden Inhalt gesucht werden:
	`FLIEGEN+*FLIEGEN  ,FLIEGEN  +FLIEGEN*`
	`1        10        20        30`
	Die Zahlen unter der Zeile zeigen die Spaltennummer an. Die Suchzeichenfolge sei jeweils das Wort 'FLIEGEN'; jedoch ist die Einschließung in Hochkomma bzw. Anführungszeichen zu beachten:
	`'FLIEGEN' Treffer auf Spalte 1, 10, 20 und 30`
	`'FLIEGEN" Treffer auf Spalte 10 und 20`
	`"FLIEGEN' Treffer auf Spalte 1 und 20`
	`"FLIEGEN" Treffer auf Spalte 20`

Definition	**Bedeutung**
str-var::=#Sn	Eine der 21 Zeichenfolgevariablen #SO, #S1 ..., #S20. Zeichenfolgevariablen sind als besondere zusätzliche Zeilen einer Datei anzusehen. Hier können irgendwelche Zeichenfolgen oder Inhalte anderer Zeilen zwischengespeichert werden. Bis auf wenige Ausnahmen können Zeichenfolgevariablen wie herkömmliche Zeilen behandelt werden. Die Ausnahmefälle kann man der Syntax der Anweisungen entnehmen. Der Inhalt einer Zeichenfolgevariablen hat mindestens die Länge 1 und darf die Länge 256 nicht überschreiten. Löscht man eine Zeichenfolgevariable, so hat diese danach als Inhalt ein Leerzeichen. Dies ist auch die Vorbelegung aller Zeichenfolgevariablen.
tab::=char	Tabulatorzeichen, das ungleich dem Anweisungssymbol edtsymb sein muß.
text::=chars	Ein über die Datenstation einzugebender Text, der aus nicht mehr als 256 Zeichen bestehen darf. Beginnt die eingegebene Zeichenfolge mit dem EDT-Anweisungssymbol (@) und folgt ein von diesem Anweisungssymbol verschiedenes Zeichen (außer Leerzeichen) nach, so wird die Eingabe als EDT-Anweisung interpretiert. Will man dies vermeiden und ab Spalte 1 das Anweisungssymbolzeichen als Text stehen haben, so sind 2 Anweisungssymbole (@@) zu schreiben.
usersymb::=char	Benutzerfluchtsymbol für externe Anweisungsroutinen. Es wird mit @USE definiert.

Definition	Bedeutung
ver::=*|int	Versionsnummer einer katalogisierten Datei. Hierfür kann entweder (*) oder (int) angegeben werden, wobei int für eine nicht negative Zahl steht. In den EDT-Anweisungen @OPEN, @ELIM, @GET, @INPUT, @READ, @SAVE, @UNSAVE und @WRITE kann eine solche Versionsnummer angegeben werden, etwa @GET 'P.BEISPIEL'(2) Wird bei den @GET, @INPUT und @READ für die Versionsnummer (*) angegeben, so wird nach der Durchführung der Anweisung die aktuelle Versionsnummer auf die Datenstation ausgegeben. Wird jedoch eine Zahl - etwa (2) - angegeben, so wird die gültige Versionsnummer nur dann ausgegeben, wenn die angegebene Versionsnummer falsch ist. Es wird jedoch trotzdem die angegebene Datei gelesen. Ausnahme: Die Abarbeitung einer @INPUT-Datei wird wie bei DMS-Fehler abgebrochen, auch wenn es sich um eine Eingabe-Anweisung handelt. Wird bei den Ausgabe-Anweisungen die Versionsnummer (*) angegeben, so erscheint die neue Versionsnummer der Datei auf der Datenstation und danach erfolgt die Ausgabe der Datei. Wird anstelle des Sterns eine Zahl angegeben, so erfolgt die Ausgabe nur dann, wenn die richtige Versionsnummer angegeben wurde. Im anderen Fall wird lediglich die richtige Versionsnummer auf die Datenstation ausgegeben. Jedesmal dann, wenn eine Datei verändert oder erstmalig angelegt wird, wird eine neue Versionsnummer erzeugt. Diese neue Versionsnummer entsteht aus der alten durch Hinzuaddieren von 1 (Ausnahmen sind für die @OPEN-Anweisung möglich).

Definition	**Bedeutung**
ver::=(Fortsetzung)	Eine neue Datei hat die Versionsnummer 1. Die höchste erreichbare Versionsnummer ist 255. Danach wird die Versionsnummer nach 0 geändert. Dies ist auch die Versionsnummer einer Datei vor ihrer Katalogisierung. Die Versionsnummer ändert sich dann nicht, wenn eine Ausgabe-Anweisung keine Zugriffe auf die Datei zur Folge hat. Wird z.B. in @ELIM ein Bereich genannt, der in dieser Datei nicht existiert, so bleibt die Versionsnummer dieser Datei unverändert. In den anderen Fällen ändert sich jedoch die Versionsnummer auch dann, wenn eine Datei vollständig überschrieben wurde und auch dann, wenn aus einer SAM- eine ISAM-Datei oder umgekehrt gemacht wird. Die Versionsnummern bieten einen erhöhten Schutz vor Zerstörung einer Datei. Wird beim Einlesen der Datei eine Versionsnummer angegeben, so erhält man mit dem Einlesen die gültige Versionsnummer ausgegeben und kann daran einen etwaigen veralteten Stand der Datei erkennen.
vers::=chars|*STD	Versionsbezeichnung eines Bibliothekselements.
vpos::=op n|vpos-op| vpos-op (m,...)	Relative horizontale Position.
vpos-op::=+|–|++|– –	Horizontale Positionieranweisung.
xpath::=chars|.str-var	Zeichenfolge mit maximaler Länge von 256 Zeichen. Gibt den Namen einer POSIX-Datei an (eventuell inclusive Verzeichnis). Leerzeichen und nichtabdruckbare Zeichen innerhalb des Namens sind nur bei Angabe in str-var möglich.

6.3 Übersicht der Anweisungen

Verwaltung des EDT

	Es wird ein neues Anweisungssymbol für die Anweisungen definiert.	F-Modus L-Modus
@**:**	edtsymb	
	Automatisches Sichern von nicht gesicherten virtuellen Dateien ein- oder ausschalten.	F-Modus L-Modus
@**AUTO**SAVE	{ [**ID**=name] [[,] **T**IME=n] [**ON**] **O**FF }	
	Der Block-Modus für Blockeingabe wird ein- bzw. ausgeschaltet.	F-Modus L-Modus
@**B**LOCK @**BK**	{ [**ON** [,] [**A**UTOFORM]] **OF**F }	
	Durch Anweisungen veränderte Zeilen werden am Bildschirm ausgegeben. Eingegebene Zeilen werden auf ihre Länge geprüft.	L-Modus
@**CH**ECK	{ [**ON**] **OF**F } [,] [cl]	
	Die Codiertabelle einer Zeile wird eingeschaltet und am Bildschirm angezeigt (Format 1).	F-Modus L-Modus
@**CODE**	ln, **S**HOW	
	Die Codiertabelle einer Zeile wird eingeschaltet (Format 2).	F-Modus L-Modus
@**CODE**	ln	
	Die aktuelle Codiertabelle wird eingeschaltet, angezeigt oder ausgeschaltet (Format 3).	F-Modus L-Modus
@**CODE**	[**ON**] | **S**HOW | **O**FF	
	Der angegebene Coded Character Set wird eingestellt.	F-Modus L-Modus
@**CODEN**AME	[name]	

@**D**ELIMIT	Die Menge der Textbegrenzerzeichen (siehe @ON) kann verändert werden.	F-Modus L-Modus
	= { **R** | str1 | **+|–** str2 }	
@**I**NPUT	Der Eingabemodus des L-Modus wird geändert (Format 3).	L-Modus
	{ [**C**HAR] | **H**EX|**X** [**I**SO] | **B**INARY }	
@**LIM**ITS	Die niedrigste und die höchste vergebene Zeilennummer und die Anzahl der Zeilen der aktuellen Arbeitsdatei werden ausgegeben.	F-Modus L-Modus
@**LOW**ER	Es wird angegeben, ob zwischen Groß- und Kleinbuchstaben unterschieden werden soll.	F-Modus L-Modus
	[**O**N] | **OF**F	
@**P-K**EYS	Der Standardwertesatz wird in die programmierbaren Tasten geladen oder am Bildschirm angezeigt.	F-Modus L-Modus
	[**S**HOW]	

	Es werden Werte für die Ein- und Ausgabe und für die Dateibearbeitung voreingestellt.	F-Modus L-Modus
@**PAR**	[fwkfv | **GL**OBAL] [,] [**E**DIT[[-] **LONG**] [=**ON**] | =**O**FF] [,] [**HEX** [=**ON**] | =**O**FF] [,] [**LOW**ER [=**ON**] | =**O**FF] [,] [**E**DIT[-]**FULL** [=**ON**] | =**O**FF] [,] [**PROT**ECTION [=**ON**] | =**O**FF] [,] [**SC**ALE [=**ON**] | =**O**FF] [,] [**INF**ORMATION [=**ON**] | =**O**FF] [,] [**I**NDEX [=**ON**] | =**O**FF] [,] [**OPT**IMIZE [=**ON**] | =**O**FF] [,] [**REN**UMBER [=**ON**] | =**O**FF] [,] [**SPLIT** = n fwkfv | **O**FF] [,] [**SEP**ARATOR = 'char' | **O**FF] [,] [**C**ODE= **E**BCDIC | **I**SO] [,] [[**ELEMENT**] [-] **TYP**E = elemtyp] [,] [**INC**REMENT = inc] [,] [**L**IBRARY = path] [,] [**LIM**IT = cl] [,] [**STRUC**TURE = 'char'] [,] [**SDF-PROG**RAM = name]	
	Die Hochkommas und Anführungszeichen, die als Begrenzerzeichen für Dateinamen und Zeichenfolgen in Anweisungen dienen, werden durch die angegebenen Zeichen ersetzt.	F-Modus L-Modus
@**Q**UOTE @**QE**	{ spec, char spec ,char }	
	Das Zeilenbereichssymbol (standardmäßig &) und der Spaltenbereich (standardmäßig 0.0001 bis 9999.9999) können verändert werden.	F-Modus L-Modus
@**RA**NGE	[=r =rng [:domain]]	
	Die Auftrags- und Benutzerschalter können gesetzt und rückgesetzt werden.	F-Modus L-Modus
@**SETS**W	{ [**ON**=] **OF**F= } [**U**] int1 [-int2] [,...]	

@**SYM**BOLS	Definieren des aktuellen Füllzeichens und der Jokersymbole Asterisk und Slash.	F-Modus L-Modus
	[,] [**A**STERISK [= '*' | ='spec1'] [,] [**S**LASH [= '/' | ='spec2'] [,] [**FILLER**] [= X'00' | =X'hex' | ='char']	
@**SYN**TAX	Einstellen der Syntaxkontrolle und des Ausführungsmodus.	F-Modus L-Modus
	[**SEC**URITY { [=**HIGH**] | =**LOW** }] [[,] **TEST**MODE { [**=ON**] | **=OF**F }]	
@**T**ABS	Ein Tabulatorzeichen und die dazugehörigen Spalten können vereinbart und am Bildschirm ausgegeben werden.	F-Modus L-Modus
	:: [tab [[:]cl1 [,cl2,...]] [{ **C**HECK | **F**ORWARD } [cl]]] [cl1[,cl2,...]] [**ON**] | **O**FF [::] **V**ALUES	
@**V**DT	Die Anzahl der Bildschirmzeilen wird vereinbart. (Bei DSS 9763 auch Anzahl der Bildschirmspalten)	F-Modus L-Modus
	[int] [,] [**F1** | **F2**]	
@**VT**CSET	Bei der Ausgabe am Bildschirm werden LINE-Mode-Steuerzeichen ausgewertet.	F-Modus L-Modus
	[**ON**] | **OF**F	
EDIT **L**ONG	Die Sätze werden vollständig am Bildschirm dargestellt.	F-Modus
	[**ON**] | **O**FF	
HEX	Der Hexadezimalmodus wird ein- oder ausgeschaltet.	F-Modus
	[**ON**] | **O**FF	
INDEX	Die Zeilennummernanzeige wird ein- oder ausgeschaltet.	F-Modus
	[**ON**] | **O**FF	
SCALE	Der Spaltenzähler wird ein- oder ausgeschaltet.	F-Modus
	[**ON**] | **O**FF	
SPLIT	Der Bildschirm wird in 2 Arbeitsfenster geteilt.	F-Modus
	n(fwkfnr) | 0 | **O**FF	

Bearbeitung von Dateien

@**CL**OSE	Es wird die zuvor durch @OPEN eröffnete Datei geschlossen.	F-Modus L-Modus
	[**N**OWRITE]	
@**EL**IM	Eine ISAM-Datei wird gelöscht. Der Katalogeintrag bleibt erhalten.	F-Modus L-Modus
	['file'] [(ver)] range* [**B**OTH]	
@**F**ILE	Ein Dateiname für @READ, @WRITE, @GET, @SAVE und @OPEN wird vereinbart.	F-Modus L-Modus
	[string [(ver)]] [**LOC**AL]	
@**G**ET	Eine ISAM-Datei wird in die aktuelle Arbeitsdatei eingelesen.	F-Modus L-Modus
	['file'] [(ver)] [range*] [:col:] [**N**ORESEQ]	
@**OP**EN	Eine ISAM-Datei wird in der Arbeitsdatei 0 real eröffnet.	F-Modus L-Modus
	['file1'] [(ver)] [[**K**EY] [**A**S 'file2' [**O**VERWRITE]]]	
@**REA**D	Eine SAM-Datei wird in die aktuelle Arbeitsdatei eingelesen.	F-Modus L-Modus
	['file'] [(ver)] [range*] [:col:] [{**R**ECORD**S** / **K**EY}] [**S**TRIP]	
@**SA**VE	Die aktuelle Arbeitsdatei wird als ISAM-Datei gespeichert.	F-Modus L-Modus
	['file'] [(ver)] [range*] [:col:] [{**U**PDATE / [**R**ENUMBER [ln [(inc)]]] [**O**VERWRITE]}]	
@**U**NSAVE	Die Datei und ihr Katalogeintrag werden gelöscht.	F-Modus L-Modus
	'file' [(ver)]	
@**W**RITE	Die aktuelle Arbeitsdatei wird als SAM-Datei gespeichert (Format 1).	F-Modus L-Modus
	['file'] [(ver)] [range*] [:col:] [**K**EY] [{**U**PDATE / **O**VERWRITE}]	

Bearbeitung von POSIX-Dateien

	Es wird eine zuvor durch @XOPEN eröffnete POSIX-Datei geschlossen.	F-Modus L-Modus
@**CL**OSE	[**N**OWRITE]	
	Einlesen einer POSIX-Datei.	F-Modus L-Modus
@**XC**OPY	**F**ILE=xpath [,**C**ODE=**E**BCDIC | **I**SO]	
	Öffnen und Einlesen einer POSIX-Datei.	F-Modus L-Modus
@**XO**PEN	**F**ILE=xpath [,**C**ODE=**E**BCDIC | **I**SO] [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	
	Inhalt der aktuellen Arbeitsdatei in eine POSIX-Datei schreiben.	F-Modus L-Modus
@**XW**RITE	**F**ILE=xpath [,**C**ODE=**E**BCDIC | **I**SO] [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	

Bearbeitung von Programm-Bibliotheken und Dateien

	Es wird das zuvor durch @OPEN Format 2 geöffnete Element einer Programm-Bibliothek zurückgeschrieben und geschlossen.	F-Modus L-Modus
@**CL**OSE	[**N**OWRITE]	
	Das Element der Programm-Bibliothek oder eine Datei wird in die aktuelle Arbeitsdatei kopiert (Format 2).	F-Modus L-Modus
@**C**OPY	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2 } [{ **B**EFORE **A**FTER } ln]	

@**D**ELETE	Das angegebene Element einer Programm-Bibliothek oder die angegebene Datei wird gelöscht (Format 2).	F-Modus L-Modus
	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **F**ILE=path2 }	
@**OP**EN	Das Programm-Bibliothekselement oder eine Datei (SAM/ISAM) wird in der aktuellen Arbeitsdatei eröffnet (Format 2).	F-Modus L-Modus
	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2 [,**T**YPE=**I**SAM | **S**AM | **C**ATALOG] } [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	
@**SHOW**	Das Inhaltsverzeichnis einer Programm-Bibliothek oder eines Benutzerkataloges wird am Bildschirm oder in eine Arbeitsdatei ausgegeben (Format 1).	F-Modus L-Modus
	{ **L**IBRARY=path1 [,[**TYP**E=]elemtyp]] **TYP**E=elemtyp **F**ILES[=path2] } [**T**O] [ln [(inc)]] [**S**HORT | **LONG**]	
@**W**RITE	Die aktuelle Arbeitsdatei wird in das angegebene Element einer Programm-Bibliothek oder in eine Datei (SAM/ISAM) geschrieben (Format 2).	F-Modus L-Modus
	[{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2 [,**T**YPE=**I**SAM | **S**AM] }] [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	

Wechseln oder Positionieren der Arbeitsdatei

	Es wird in der Arbeitsdatei rückwärts oder vorwärts positioniert.	F-Modus
+ **–**	[n]	
++ **– –**		

+ **++** **–** **– –**	Es wird in der angegebenen Richtung auf die nächste Satzmarkierung positioniert.	F-Modus
	[([m,...])]	
> **<**	Es wird nach links oder nach rechts positioniert.	F-Modus
	[n]	
<<		
@**EN**D	Es wird in die zuletzt aktuelle Arbeitsdatei gewechselt.	L-Modus @PROC
	[comment]	
@**O**N	Sätze, die den Suchbegriff enthalten, werden markiert. Im F-Modus wird das Datenfenster auf den ersten Treffersatz positioniert (Format 4).	F-Modus L-Modus
	range* [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**NOT**] [**P**ATTERN] search [,int] **M**ARK [m]	
@**PRO**C	Es wird in eine andere Arbeitsdatei gewechselt.	L-Modus @PROC
	procnr [comment]	
@**SETF** #	Es wird in einer Arbeitsdatei zeilen- und spaltengerecht positioniert und es kann die Arbeitsdatei gewechselt werden. # ist nur im F-Modus zulässig.	F-Modus L-Modus
	[{ fwkfv / **GL**OBAL / (fwkfnr) }] [{ ln / vpos }] [{ :cl: / hpos }]	
fwkfnr fwkfv	Es wird in eine andere Arbeitsdatei gewechselt.	F-Modus

Behandlung der Zeilennummern

	Die aktuelle Zeilennummer und Schrittweite werden verändert und die Kellerungseinträge verschoben.	L-Modus
@	[ln [(inc)] [:text]]	
	Die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht.	L-Modus
@+	[:text]	
	Die aktuelle Zeilennummer wird um die aktuelle Schrittweite vermindert.	L-Modus
@–	[:text]	
	Die Zeilennummern der aktuellen Arbeitsdatei werden neu numeriert.	F-Modus L-Modus
@**R**ENUMBER	[ln [(inc)]]	
	Eine Folgenummer wird in jeder Zeile des angegebenen Bereichs ab einer bestimmten Spalte abgelegt (Format 1).	F-Modus L-Modus
@**SEQ**UENCE	[rng] [:[cl] [:[[n1] (n2)]]]	
	Die Zeilennummer jeder Zeile wird in jeder Zeile des angegebenen Bereichs ab einer bestimmten Spalte abgelegt (Format 2).	F-Modus L-Modus
@**SEQ**UENCE	[rng*] : [cl] : **L**INE	
	Der angegebene Bereich wird auf aufsteigende Folgenummern geprüft (Format 3).	F-Modus L-Modus
@**SEQ**UENCE	[rng] : [cl] : **C**HECK [int]	
	Eine neue aktuelle Zeilennummer und die Schrittweite werden festgelegt. Eine gleichzeitige Texteingabe ist möglich (Format 6).	F-Modus L-Modus
@SET	ln [(inc)] [:text]	

Erzeugen, Einfügen und Ändern von Texten

	Ab der angegebenen Spalte wird Text eingefügt oder überschrieben. Leerzeichen am Satzende werden gelöscht.	F-Modus L-Modus
@**COL**UMN	cl **ON** range { [**C**HANGE] / **I**NSERT } [:] string	

	Die angegebene Zeile wird erzeugt (Format 1).	F-Modus L-Modus
@**CR**EATE	line [:] [string[,...]]	
	Wird im angegebenen Bereich der Suchbegriff gefunden, so wird sie durch die angegebene Zeichenfolge ersetzt (Format 7).	F-Modus L-Modus
@**O**N	range [:domain] **C**HANGE [{ **A**LL / **F** }] [**R**] [**P**ATTERN] search [,int] [**T**O] string	
	Wird im angegebenen Bereich der Suchbegriff gefunden, ersetzt die angegebene Zeichenfolge den Text vor bzw. nach dem Treffer bzw. sie wird vor bzw. nach dem Treffer eingefügt (Format 8).	F-Modus L-Modus
@**O**N	range [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**P**ATTERN] search [,int] { **C**HANGE / **I**NSERT } { **P**REFIX / **S**UFFIX } string	
	Jeder Zeile des angegebenen Bereichs wird string vorangestellt.	F-Modus L-Modus
@**PRE**FIX	range **W**ITH string	
	Syntaxpüfung von Datenzeilen durch SDF.	F-Modus L-Modus
@**SDF**TEST	[range*] [**PROG**RAM [=name]]	
	Zusammenhängende Zeilenbereiche sortieren.	F-Modus L-Modus
@**SORT**	[rng*] { [:domain] / :**R** (clrng) } { [**A**] / **D** }	
	An jede Zeile des angegebenen Bereichs wird string angehängt.	F-Modus L-Modus
@**SU**FFIX	range **W**ITH string	

	Bestehende Datensätze werden ganz oder teilweise geändert oder gelöscht (Format 1).	L-Modus
@**U**PDATE	ln [:domain] ; text	
	Ein Dateiabschnitt wird in aufbereiteter Form ausgegeben (Format 2).	L-Modus
@**U**PDATE	[ln] [:domain]	
	Für Korrekturen mit @UPDATE wird ein Standard-Spaltenbereich vereinbart (Format 3).	L-Modus
@**U**PDATE	**C**OLUMN [domain]	

Kopieren und Übertragen von Zeilen

	Der angegebene Bereich wird aus einer beliebigen Arbeitsdatei in die aktuelle Arbeitsdatei kopiert (Format 1).	F-Modus L-Modus
@**C**OPY	rng [(procnr)] [**T**O ln1 [(inc)] [:] [ln2]] [,...]	
	Der angegebene Bereich aus einer beliebigen Arbeitsdatei wird in die aktuelle Arbeitsdatei übertragen. Der ursprüngliche Bereich wird also gelöscht.	F-Modus L-Modus
@**M**OVE	rng [(procnr)] [**T**O ln1 [(inc)] [:] [ln2]] [,...]	
	Die mit der Satzmarkierung m markierten Sätze werden in die Arbeitsdatei procnr kopiert. Bei KEEP werden die Zeilennummern beibehalten (Format 5).	F-Modus L-Modus
@**O**N	range* [:domain] **F**IND [{ **A**LL / **F** }] [**NOT**] **M**ARK m [**C**OPY [**T**O]] (procnr) [**K**EEP] [**O**LD]	
	Sätze, die den Suchbegriff enthalten, werden in die Arbeitsdatei procnr kopiert. Bei KEEP werden die Zeilennummern beibehalten (Format 6).	F-Modus L-Modus
@**O**N	range* [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**C**OPY [**T**O]] (procnr) [**K**EEP] [**O**LD]	

Löschen von Arbeitsdateien, Zeilen, Texten und Satzmarkierungen

@**D**ELETE	Die angegebenen Bereiche werden gelöscht (Format 1).	F-Modus L-Modus
	[rng [:domain]] [,...]	
@**D**ELETE	Die angegebenen Satzmarkierungen werden gelöscht (Format 3).	F-Modus L-Modus
	MARK [m, [...]]	
@**DR**OP	Es werden alle oder die angegebenen Arbeitsdateien gelöscht und freigegeben.	F-Modus L-Modus
	{ procnr [,...] / **A**LL }	
@**O**N	Wird im angegebenen Bereich der Suchbegriff gefunden, wird er aus den betreffenden Zeilen gelöscht (Format 9).	F-Modus L-Modus
	range [:domain] **D**ELETE [{ **A**LL / **F** }] [**R**] [**P**ATTERN] search [,int]	
@**O**N	Wird im angegebenen Bereich der Suchbegriff gefunden, wird in den Trefferzeilen der Text vor bzw. nach dem Treffer gelöscht (Format 10).	F-Modus L-Modus
	range [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**P**ATTERN] search [,int] **D**ELETE { **P**REFIX / **S**UFFIX }	
@**O**N	Wird im angegebenen Bereich der Suchbegriff gefunden, wird die Trefferzeile gelöscht (Format 11).	F-Modus L-Modus
	range [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**NOT**] [**P**ATTERN] search [,int] **D**ELETE	

Vergleich von Arbeitsdateien

	Die angegebenen Bereiche zweier Arbeitsdateien werden miteinander verglichen. Der Vergleich liefert als Ergebnis Zeilennummern (Format 1).	F-Modus L-Modus
@**COM**PARE	[procnr1] :rng*1 **W**ITH [procnr2] :rng*2 [,[int1] [(int2)] [**L**IST [ln [(inc)]]]]	
	Zwei Arbeitsdateien werden vollständig miteinander verglichen. Der Vergleich liefert als Ergebnis Zeilennummern und den Inhalt der Zeilen (Format 2).	F-Modus L-Modus
@**COMP**ARE	[[procnr1] [**W**ITH]] procnr2 [**L**IST [procnr3]] [,procnr4]	

Wechseln des Arbeitsmodus

	Es wird in den F-Modus Bildschirmdialog umgeschaltet. Rückkehr zur unterbrochenen Abarbeitung mit @HALT oder @RETURN (z.B. in Systemprozeduren).	F-Modus L-Modus
@**DIALOG**		
	Es wird vom L-Modus aus in den F-Modus-Bildschirmdialog umgeschaltet und umgekehrt. Weiter können Einstellungen für das Editieren im L-Modus vorgenommen werden.	F-Modus L-Modus
@**E**DIT	{ **F**ULL SCREEN [**O**NLY] [**P**RINT] [**S**EQUENTIAL] [cl] }	

Ausgabe von Zeilen und Informationen

	Die letzte n-te Anweisung wird in der Anweisungszeile ausgegeben.	F-Modus
[n]#		
	Die gefragten Dateien (path) werden am Bildschirm oder in eine Arbeitsdatei ausgegeben.	F-Modus L-Modus
@**FS**TAT	{ ['file'] str-var } [[**T**O] ln [(inc)]] [{ **S**HORT **LONG** }]	

@**L**IST	Der gewünschte Bereich wird auf SYSLST ausgegeben.	F-Modus L-Modus
	[rng [:domain] [**X**] [**N**] [**C** [int]|**P** int] [**I**] [**S**]] [,...]	
@**LOG**	Die Protokollierung wird gesteuert.	F-Modus L-Modus
	[{**A**LL / **C**OMMANDS / **N**ONE}] [{**SYSLST** / **SYSLST** nn / LIST-**VAR**=chars}]	
@**O**N	Jede Zeile des angegebenen Bereichs, in der der Suchbegriff vorkommt, wird am Bildschirm ausgegeben (Format 1).	F-Modus L-Modus
	range [:domain] **P**RINT [{**A**LL / **F**}] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**S**] [**N**]	
@**O**N	Die Spaltennummer, ab der der Suchbegriff gefunden wurde, wird am Bildschirm ausgegeben (Format 2).	F-Modus L-Modus
	range [:domain] **CO**LUMN [{**A**LL / **F**}] [**R**] [**P**ATTERN] [search [,int]]	
@**O**N	Wird der Suchbegriff gefunden, so wird die Nummer der Trefferzeile in #L00, die Beginn- und Endespalte in #I00 und #I01 abgelegt (Format 3).	L-Modus @PROC
	range [:domain] **F**IND [{**A**LL / **F**}] [**R**] [**NOT**] [**P**ATTERN] search [,int]	
@**PA**GE	Bewirkt einen Seitenvorschub auf SYSLST.	F-Modus L-Modus
@**P**RINT	Im F-Modus kann der Inhalt von Zeichenfolgevariablen am Bildschirm ausgegeben werden. Im L-Modus auch der Inhalt von Zeilenbereichen.	F-Modus L-Modus
	[rng [:domain] [**X**] [**N**] [**S**] [**V** | **E**]] [,...]	

	Die aktuelle Arbeitsdatei wird ausgegeben.	L-Modus
@**PRO**C		
	Eine Liste der im System möglichen Coded Character Set Namen wird am Bildschirm oder in eine Arbeitsdatei ausgegeben (Format 1).	F-Modus L-Modus
@**SHOW**	**C**CS [[**T**O] ln [(inc)]]	
	Es können aktuelle Einstellungen und der Inhalt von Zeilennummern und Ganzzahlvariablen am Bildschirm ausgegeben werden.	F-Modus L-Modus
@**ST**ATUS	[=**A**LL] | [= **T**IME | **B**UFFER | **S**IZE | **SY**MBOLS | **D**ELIM | **V**DT | **M**ODES | **F**ILE | **P**AR[(procnr)] | **L**INEV | **I**NTV | ln-var | int-var | **SDF** | **C**CS | **LOG**] [,...] [**T**O ln [(inc)]]	
	Es werden Prozeßinformationen am Bildschirm ausgegeben.	F-Modus L-Modus
@**TMODE**		

Unterbrechen oder Beenden des EDT

	Der EDT wird beendet.	F-Modus L-Modus
@**EN**D	[comment]	
	Der EDT wird beendet. Das angegebene Programm wird geladen und gestartet.	F-Modus L-Modus
@**EX**EC	string	
	Der EDT wird beendet. Bei Aufruf des EDT als Unterprogramm kann eine Nachricht angegeben werden.	F-Modus L-Modus
@**H**ALT	[**ABNORMAL**] [message]	
	Der EDT wird beendet. Das angegebene Programm wird geladen.	F-Modus L-Modus
@**LO**AD	string	
	Der EDT wird beendet. Bei Aufruf des EDT als Unterprogramm kann eine Nachricht angegeben werden.	F-Modus L-Modus
@**RET**URN	[message]	
	Es wird in das Betriebssystem verzweigt bzw. ein BS2000-Kommando zur Abarbeitung an das BS2000 durchgereicht.	F-Modus L-Modus
@**SY**STEM	[string [**T**O ln [(inc)]]]	

Springen in EDT-Prozeduren

	Eine Zeile wird erzeugt, die über @GOTO angesprungen werden kann bzw. die als Kommentar dient.	L-Modus @PROC
@**CON**TINUE	[comment]	
	Es wird zur angegebenen Zeile verzweigt.	@PROC
@**GO**TO	ln	

@IF	Zuvor in Anweisungen aufgetretene EDT- oder DVS- Fehler setzen den EDT- bzw. DVS-Fehlerschalter. Mit @IF kann dieser abgefragt werden. Ist die Bedingung erfüllt, wird text ausgeführt (Format 1).	L-Modus @PROC
	{ **E**RRORS **N**O ERRORS **D**MS ERRORS **N**O **D**MS ERRORS } : text	
@IF	Es werden Inhalte von Variablen oder Zeilen miteinander verglichen. Ist die Bedingung erfüllt, wird zur angegebenen Zeile verzweigt bzw. die Prozedur abgebrochen (Format 2).	@PROC
	{ [**S**] string1 rel string2 ln1 rel ln2 [**I**] int1 rel int2 } { **G**OTO ln **R**ETURN }	
@IF	Es wird abgefragt, ob beim vorhergegangenen @ON ein Treffer festgestellt wurde bzw. ob die aktuelle Arbeitsdatei leer ist . Ist die Bedingung erfüllt, wird zur angegebenen Zeile verzweigt bzw. die Prozedur abgebrochen (Format 3).	@PROC
	{ **.T**RUE**.** [rel cl] **.F**ALSE**.** **.E**MPTY**.** } { **G**OTO ln **R**ETURN }	
@IF	Es wird geprüft, welche Auftrags- bzw. Benutzerschalter ein- bzw. ausgeschaltet sind. Ist die Bedingung erfüllt, wird zur angegebenen Zeile verzweigt bzw. die Prozedur abgebrochen (Format 4).	@PROC
	{ **ON** **OF**F } = [**U**] int { **G**OTO ln **R**ETURN }	
@RESET	Der EDT- und DVS-Fehlerschalter werden zurückgesetzt.	F-Modus L-Modus @PROC

Verwaltung und Ausführung von EDT-Prozeduren

	Die vom Bildschirm eingelesene Zeichenfolge wird in die angegebene Zeile geschrieben (Format 2).	L-Modus @PROC
@**CR**EATE	line **R**EAD [string [,...]]	
	Die angegebene Prozedur wird ausgeführt. (Format 1).	F-Modus L-Modus @PROC
@**DO**	procnr [,] [(param [,...])] [spec] [=ln1,ln2 [, [-] ln3]] [**P**RINT]	
	Die Protokollierung der abgearbeiteten Zeilen wird ein- bzw. ausgeschaltet (Format 2).	@PROC
@**DO**	**N** | **P**	
	Die Bearbeitung der aktuellen Datei wird beendet. Es wird wieder in die zuletzt aktuelle Arbeitsdatei zurückgekehrt.	L-Modus @PROC
@**EN**D	[comment]	
	Die angegebene Datei wird ganz oder teilweise als Prozedur abgearbeitet (Format 1).	F-Modus L-Modus
@**I**NPUT	'file' [(ver)] [range*] [:col:] [{**R**ECORD**S** | **KEY**}] [**P**RINT]	
	Eine @INPUT-Prozedur aus einem Bibliothekselement oder aus einer Datei starten (Format 2).	F-Modus L-Modus
@**I**NPUT	{**L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2} [**P**RINT]	
	Es können Prozeduren kommentiert werden.	L-Modus @PROC
@**N**OTE	[comment]	
	Im Prozedurkopf werden Parameter vereinbart.	@PROC
@**PAR**AMS	formal [,...]	
	Es wird in eine andere Arbeitsdatei gewechselt (Format 1).	L-Modus @PROC
@**PRO**C	procnr [comment]	

@**PRO**C	Die Nummern der Arbeitsdateien 1 bis 22, die frei bzw. belegt sind, werden am Bildschirm ausgegeben (Format 2).	L-Modus @PROC
	FREE | **U**SED	
@SET	Einer Ganzzahlvariablen wird ein Wert zugewiesen (Format 1).	F-Modus L-Modus @PROC
	int-var = { [+|–] int [+|– int] [...] **S**UBSTR string ln-var **L**ENGTH line **ST**RING string }	
@SET	Einer Zeichenfolgevariablen wird ein Wert zugewiesen (Format 2).	F-Modus L-Modus @PROC
	str-ln { = string = **I**NTERNAL { int-var | ln | str-var } [,cl] = **C**HAR { int-var | ln-var | str-var } }	
@SET	Einer Zeilennummervariablen wird ein Wert zugewiesen (Format 3).	F-Modus L-Modus @PROC
	ln-var = { ln int-var **S**UBSTR string **ST**RING string }	
@SET	Ein Wert wird ab der Spalte cl in der Zeile abgelegt, deren Nummer in der Zeilennummervariablen steht (Format 4).	F-Modus L-Modus @PROC
	ln-var [,cl] = **C**HAR { int-var | str-var | ln-var1 }	

@SET	Datum und Urzeit werden einer Zeichenfolgevariablen zugewiesen oder in einer Zeile abgelegt (Format 5).	F-Modus L-Modus @PROC
	{ str-ln / ln-var } [,cl] = { **D**ATE [**I**SO[**4**]] / **T**IME }	

Aufruf eines Anwenderprogramms

@**RUN**	Ein anderes Programm wird geladen und gestartet. Das Programm muß als Modul vorliegen. Der EDT bleibt geladen.	F-Modus L-Modus
	(entry [,modlib]) [:string] [[,]**U**NLOAD]	
@**UNL**OAD	Ein geladenes Programm wird entladen. Das Programm muß als Modul vorliegen.	F-Modus L-Modus
	(name)	
@**USE**	Eine externe Anweisungsroutine und das zugehörige Anweisungssymbol werden vereinbart.	F-Modus L-Modus
	COMMAND = 'usersymb' [({ entry / * } [,modlib])]	

Löschen, Lesen, Katalogisieren und Ausgeben von Jobvariablen

@**ERAJV**	Löschen von Einträgen von Jobvariablen aus dem Katalog.	F-Modus L-Modus
	str [**A**LL]	
@**GETJV**	Wert einer Jobvariablen am Bildschirm ausgeben, in eine Arbeitsdatei schreiben oder einer Zeichenfolgevariablen zuordnen.	F-Modus L-Modus
	[string] [=line]	
@**SETJV**	Eintragen einer Jobvariablen in den Katalog oder einer Jobvariablen einen Wert zuweisen.	F-Modus L-Modus
	{ [string1] = string2 [,...] / string1 }	

	Abfrage, welche Jobvariablen vorhanden sind und welche Eigenschaften diese haben.	F-Modus L-Modus
@**STAJV**	[string] [**T**O ln [(inc)]] [**S**HORT] | **LONG**	

Deklarieren und Lesen von S-Variablen und Listenvariablen

	Einlesen von Elementen einer Listenvariablen.	F-Modus L-Modus
@**GETL**IST	string [range*] [:col:]	
	Inhalte von S-Variablen am Bildschirm ausgeben oder einer Zeichenfolgevariablen zuordnen.	F-Modus L-Modus
@**GETV**AR	{ string [=line | =int-var] **SYSEDT** }	
	Erweitern und neu Beschreiben einer Listenvariablen.	F-Modus L-Modus
@**SETL**IST	string { [range*] [**MA**RK [m]] str-var } [:col:][,] [**M**ODE=**A**PPEND | **P**REFIX | **O**VERWRITE]	
	Deklarieren von S-Variablen und diesen Werte zuweisen.	F-Modus L-Modus
@**SETV**AR	{ string [=string1 | =int-var] **SYSEDT** } [,**M**ODE=**A**NY | **N**EW | **U**PDATE]	

6.4 Beschreibung der Anweisungen

@ Verändern der aktuellen Schrittweite und Zeilennummer

@ definiert die aktuelle Zeilennummer und die aktuelle Schrittweite und verschiebt dabei die Kellerungseinträge.

Der 3-stufige Keller des EDT

Der EDT arbeitet mit einem 3-stufigen Keller. Jeder Kellerungseintrag besteht aus einem Wertepaar für Zeilennummer und Schrittweite. Zu Beginn des EDT-Laufs ist der Keller leer. Kellerungseinträge können mit @ eingetragen und verschoben werden.

Operation	Operanden	L-Modus
@	[ln [(inc)] [:text]]	

ln — Neue aktuelle Zeilennummer, z.B. 5.
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. Fehlt inc, wird mit ln implizit auch die neue aktuelle Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

inc — Neue aktuelle Schrittweite.
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.

text — Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt.
   Für die Behandlung gilt:
   - text steht am Anfang der durch ln angegebenen Zeile;
   - vorhandene Tabulatorzeichen werden berücksichtigt.
   - die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht.

2. das EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   - kein EDT-Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   - das EDT-Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt.
3. das Benutzerfluchtsymbol, wird die externe Anweisungsroutine ausgeführt (siehe @USE).

Wirkung von @ bzw. @ln

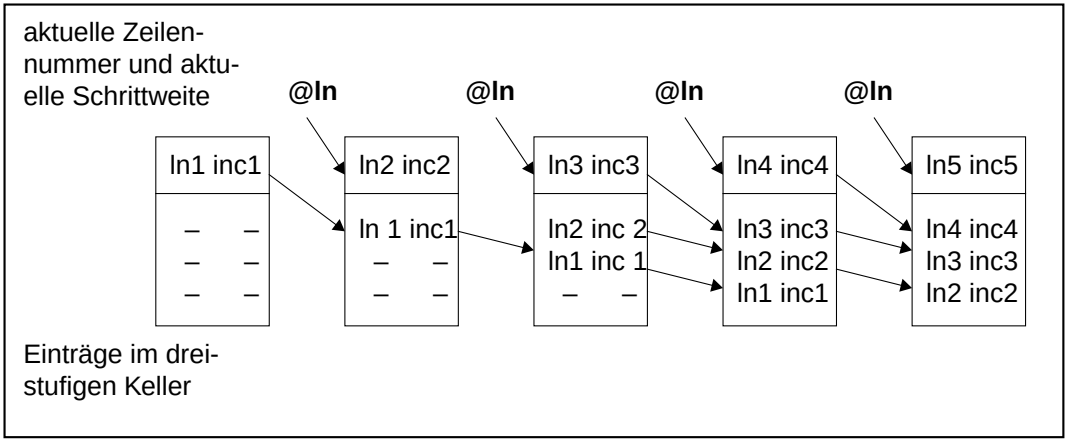


Bild 8: Wirkung von @ln

Mit @ln [(inc)] wird eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite festgelegt. Die bisherige aktuelle Zeilennummer und die bisherige aktuelle Schrittweite werden im 3-stufigen Keller des EDT gespeichert.

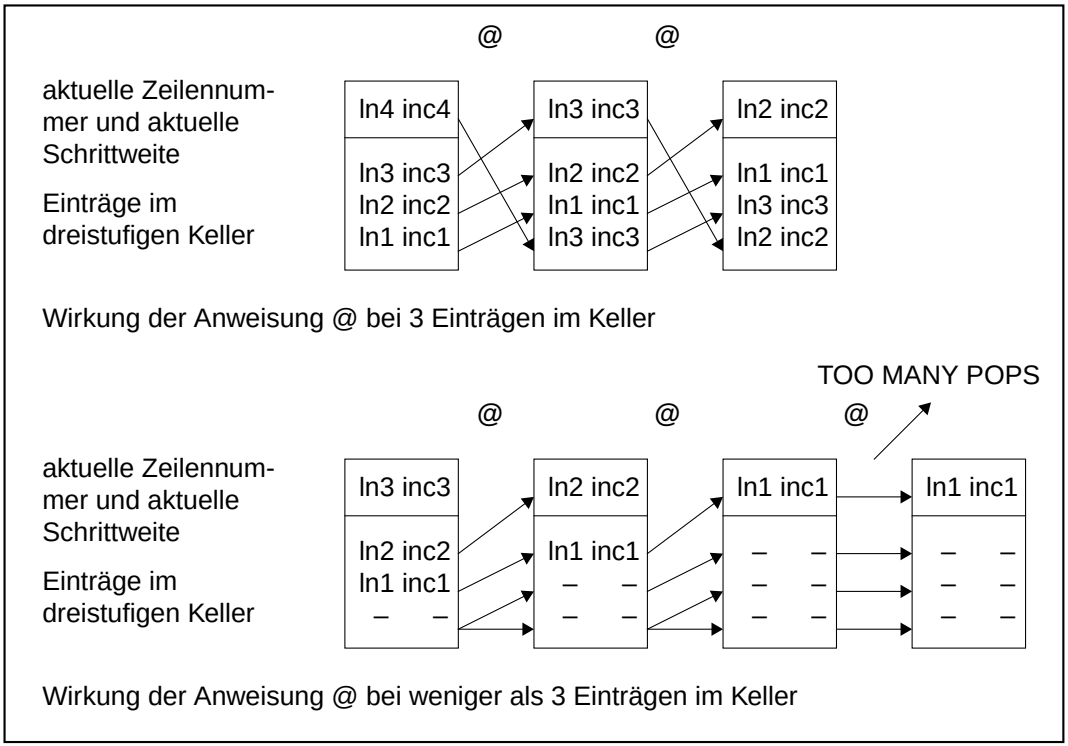


Bild 9: Kellerungseinträge verschieben

Mit @ ohne Operanden werden die im obersten Kellerungseintrag stehenden Werte zur aktuellen Zeilennummer und zur aktuellen Schrittweite. Ist kein Kellerungseintrag vorhanden, wird eine Meldung ausgegeben.

Die durch @ln erzeugten Kellerungseinträge werden gemäß Bild 9 verschoben.

@+ Erhöhen der aktuellen Zeilennummer

Mit @+ wird die aktuelle Zeilennummer um die aktuelle Schrittweite erhöht. Ist der SEQUENTIAL-Modus (siehe @EDIT) ausgeschaltet, wird die um die Schrittweite erhöhte Zeilennummer zur neuen aktuellen Zeilennummer. Ist der SEQUENTIAL-Modus eingeschaltet, wird die um die Schrittweite erhöhte Zeilennummer nur dann zur neuen aktuellen Zeilennummer, wenn keine Zeile zwischen ihr und der bisherigen aktuellen Zeilennummer liegt. Existiert eine Zeilennummer zwischen der bisherigen und der errechneten, wird diese zur neuen aktuellen Zeilennummer. Dadurch können bereits existierende Zeilen im SEQUENTIAL-Modus nicht übergangen werden.

Operation	Operanden	L-Modus
@+	[:text]	

text Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt.
   Für die Behandlung gilt:
   - text steht am Anfang der durch ln angegebenen Zeile;
   - vorhandene Tabulatorzeichen werden berücksichtigt.
   - die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht.
2. das EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   - kein EDT-Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   - das EDT-Anweisungssymbol, wird text als Textzeile wie bei 1. behandelt.
3. das Benutzerfluchtsymbol, wird die externe Anweisungsroutine ausgeführt (siehe @USE).

text kann auch @+ sein. So ist es möglich, diese Anweisung mehrmals mit sich selbst zu verketten.

@– Herabsetzen der aktuellen Zeilennummer

Mit @– wird die aktuelle Zeilennummer um die aktuelle Schrittweite herabgesetzt. Ist der SEQUENTIAL-Modus (siehe @EDIT) ausgeschaltet, wird die aktuelle Zeilennummer um die aktuelle Schrittweite verringert. Ist der SEQUENTIAL-Modus eingeschaltet, wird die aktuelle Zeilennummer nur dann um die aktuelle Schrittweite verringert, wenn keine Zeile existiert, deren Nummer zwischen der errechneten und der bisherigen Zeilennummer liegt. Existiert eine Zeilennummer zwischen der errechneten und der bisherigen, wird diese zur neuen aktuellen Zeilennummer. Existieren mehrere Zeilennummern zwischen der errechneten und der bisherigen, wird die unmittelbar vor der aktuellen Zeilennummer stehende zur neuen aktuellen Zeilennummer.

Operation	Operanden	L-Modus
@–	[:text]	

text Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt.
   Für die Behandlung gilt:
   – text steht am Anfang der durch ln angegebenen Zeile;
   – vorhandene Tabulatorzeichen werden berücksichtigt.
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht.
2. das EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein EDT-Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   – das EDT-Anweisungssymbol, wird text als Textzeile wie bei 1. behandelt.
3. das Benutzerfluchtsymbol, wird die externe Anweisungsroutine ausgeführt (siehe @USE).

text kann auch @– sein. So ist es möglich, diese Anweisung mehrmals mit sich selbst zu verketten.

@: Vereinbaren eines Anweisungssymbols

Diese Anweisung ermöglicht dem Benutzer die Vereinbarung eines neuen Anweisungssymbols.

Operation	Operanden	F-Modus / L-Modus
@:	edtsymb	

Bei dieser Anweisung muß auch im F-Modus das Anweisungssymbol vorangestellt werden (:edtsymb ist nicht erlaubt).

edtsymb Sonderzeichen für das neue Anweisungssymbol.
Es darf nicht gleich dem Doppelpunkt oder dem aktuellen Bereichssymbol sein (siehe @RANGE).

Ist edtsymb kein Sonderzeichen, wird @: mit der Fehlermeldung abgewiesen:
% EDT3952 INVALID SYMBOL

Bei @: muß auch im F-Modus das Anweisungssymbol eingegeben werden. Indirekte Operandenangabe ist nicht zulässig.

Die Eindeutigkeit der Anweisungen ist nur dann gewährleistet, wenn edtsymb verschieden ist von:

– +, – und dem Benutzerfluchtsymbol (siehe @USE).
– <, > # und dem Semikolon; im F-Modus.
– $, %, #, *, ?, falls bei @SET der Anweisungsname nicht angegeben wird oder eine Anweisung im Operand text gegeben wird.

Beispiel

```
3.     @print ------------------------------------------------------------ (01)
1.0000 Diese Anweisung ermoeglicht dem Benutzer die Vereinbarung eines
2.0000 neuen Anweisungssymbols.
3.     @:! ---------------------------------------------------------------- (02)
3.     @print ------------------------------------------------------------ (03)
4.     !print ------------------------------------------------------------ (04)
1.0000 Diese Anweisung ermoeglicht dem Benutzer die Vereinbarung eines
2.0000 neuen Anweisungssymbols.
3.0000 @print
4.
```

(01) Mit @PRINT wird der Inhalt der Arbeitsdatei ausgegeben.

(02) Als neues Anweisungssymbol wird **!** vereinbart.

(03) @PRINT wird jetzt nicht als Anweisung sondern als Text interpretiert.

(04) Mit !PRINT wird der Inhalt der Arbeitsdatei ausgegeben.

@AUTOSAVE Automatisches Sichern

Mit @AUTOSAVE wird ein das automatische Sichern (Retten) von nicht gesicherten Arbeitsdateien ein- oder ausgeschaltet.

Operation	Operanden	F-Modus / L-Modus
@**AUTO**SAVE	{ [**ID**=name] [[,] **T**IME=n] [**ON**] / **O**FF }	

name — Frei wählbarer Identifikator für die Sicherungsdateien, maximal 8 Zeichen lang
Standardwert: EDT

n — Zeitdifferenz in Minuten zwischen einer manuellen oder automatischen Sicherung und der nächsten automatischen Sicherung. Der Minimalwert ist 0, der Maximalwert 255.
Standardwert: 5
Bei TIME=0 wird nach jedem Dialogschritt gesichert.

ON — Die automatische Sicherung wird eingeschaltet. Bei einer Sicherung wird der Inhalt von allen geänderten und auf eine andere Art nicht gesicherten virtuellen Dateien in separate ISAM-Dateien geschrieben. Die Zeilennummern werden als ISAM-Schlüssel gesichert.

OFF — Die AUTOSAVE-Funktion wird ausgeschaltet. Die Sicherungsdateien werden gelöscht.

Die Anweisung @AUTOSAVE kann in allen Arbeitsmodi außer im Batch abgesetzt werden, auch in einer EDTSTART-Prozedur. Die Anweisung wird im Batch ohne Meldung ignoriert. Die effektive Datensicherung wird nur im Dialog ausgeführt.

Gesichert werden alle Arbeitsdateien, die seit der letzten Sicherung verändert worden sind. Gesichert wird immer nach einem Dialogschritt, also vor der nächsten Eingabeaufforderung, wenn die folgenden Bedingungen erfüllt sind:

– die AUTOSAVE-Funktion ist eingeschaltet
– die definierte Zeit seit der letzten Sicherung ist verstrichen
– die Arbeitsdatei wurde vom Benutzer seit der letzten Sicherung nicht explizit gesichert

Bei Beginn des EDT-Laufs ist die AUTOSAVE-Funktion ausgeschaltet.

Bei jedem Einschalten von AUTOSAVE werden alle nicht leeren Arbeitsdateien gesichert.

Der Name der Sicherungsdateien wird wie folgt gebildet:

S.name.jjjj-mm-tt.hhmmss.SAVEnn

Die Angabe jjjj-mm-tt.hhmmss ist der Zeitpunkt, an dem die AUTOSAVE-Funktion eingeschaltet wurde.

Die Angabe nn ist die Nummer der aktuellen Arbeitsdatei.

Eine zugehörige Sicherungsdatei wird gelöscht, wenn:

– die Arbeitsdatei leer wird, (z.B. @DELETE)
– der Inhalt der Arbeitsdatei als Datei oder Bibliothekselement gesichert wird. Die möglichen Anweisungen sind: @WRITE, @SAVE, @XWRITE und @CLOSE.

Alle Sicherungsdateien werden gelöscht

– bei @AUTOSAVE OFF
– bei Beendigung des EDT-Laufes durch @HALT, @END, @EXEC oder @LOAD
– bei Rückkehr zum Hauptprogramm

Die Sicherungsdateien bleiben erhalten

– bei abnormaler Beendigung
– bei Verlassen des EDT durch K2 oder @SYSTEM ohne Rückkehr.

Real eröffnete ISAM-Dateien werden nicht gesichert.

i

Die Wiederherstellung der einzelnen Arbeitsdateien kann erreicht werden mit:

@GET 'S.name.jjjj-mm-tt.hhmmss.SAVEnn' NORESEQ

@BLOCK Blockmodus einstellen

Diese Anweisung schaltet den geblockten Ein-Ausgabemodus (BLOCK-Modus) des EDT ein bzw. aus. Im BLOCK-Modus kann man mit einer einzigen Eingabe

– mehrere Zeilen erstellen,
– mehrere Befehle eingeben, die sequentiell abgearbeitet werden.

Die einzelnen Zeilen bzw. Befehle sind durch das (bildschirmspezifische) Zeilenendekennzeichen zu trennen. Maximal können so viele Zeichen in einem Block eingegeben werden, wie auf einer Bildschirmseite darstellbar sind. Die maximale Zeilengröße in einem Block darf 256 Zeichen nicht überschreiten. Bei Datenschreibstationen beträgt die maximale Blockgröße 1020 Zeichen.

Enthält der eingegebene Anweisungsblock eine fehlerhafte Anweisung, wird diese zusammen mit der aktuellen Zeilennummer und der entsprechenden Fehlermeldung zum Zeitpunkt ihrer Ausführung ausgegeben. Danach wird der Rest des Anweisungsblocks ausgeführt.

Operation	Operanden	F-Modus / L-Modus
@**B**LOCK @**BK**	{ [**ON** [,] [**A**UTOFORM]] **OF**F }	

ON Schaltet den BLOCK-Modus ein.

AUTOFORM Bewirkt, daß Leerzeilen beim Einlesen von der Tastatur im Line-Modus oder beim Einlesen von einer POSIX-Datei mittels @XOPEN oder @XCOPY als Inhalt das Zeilenendekennzeichen X'0D' erhalten und angelegt werden. Beim Schreiben mit @XWRITE und @CLOSE nach @XOPEN werden Datenzeilen mit dem Inhalt X'0D' als Zeilen der Länge 0 in die Datei geschrieben.
Wird der BLOCK-Modus ohne AUTOFORM eingeschaltet, werden Zeilen der Länge 0 unterdrückt.

OFF Setzt den BLOCK-Modus zurück und nimmt die AUTOFORM-Funktion zurück.

Enthält bei geblockter Eingabe ein Anweisungsblock ein @BLOCK OFF, werden die nachfolgenden Anweisungen oder Texteingaben dieses Blocks ignoriert.

Bei Beginn des EDT-Laufs ist der BLOCK-Modus standardmäßig eingeschaltet. Innerhalb von EDT-Prozeduren wird die Anweisung ignoriert.

Blockmodus an Datensichtstationen 816x

Das Arbeiten im Blockmodus ist an den Datensichtstationen 816x nur möglich, wenn mit dem Kommando MODIFY-TERMINAL-OPTIONS LINE-END-CHARACTER = C'a' ein beliebiges Zeilenendekennzeichen a vereinbart wurde.

Standardmäßig ist an den Geräten 816x für das Zeilenendekennzeichen C'\' vereinbart. LZE an den Datensichtstationen 8160 kann als Zeilenendekennzeichen verwendet werden, wenn diese Datensichtstation auch als 8160 generiert ist.

Eine Nichtvereinbarung mit MODIFY-TERMINAL-OPTIONS LINE-END-CHARACTER = NONE kann zu Fehlern führen.

@CHECK Zeilen prüfen

Diese Anweisung ermöglicht die Protokollierung jeder Zeile, die in der virtuellen oder in einer mit @OPEN eröffneten Datei durch eine Anweisung aufgebaut oder verändert wird. Die betreffende Zeile wird am Bildschirm ausgegeben. Ferner kann mit @CHECK die Überprüfung der Zeilenlänge gesteuert werden.

Operation	Operanden	L-Modus
@**CH**ECK	{ [**ON**] / **OF**F } [,] [cl]	

ON — Schaltet den CHECK-Modus ein. Bei eingeschaltetem CHECK-Modus wird jede Zeile auf dem Bildschirm ausgegeben, die in der virtuellen oder in einer mit @OPEN eröffneten Datei durch eine der folgenden Anweisungen aufgebaut oder verändert wird: @COLUMN, @COPY, @CREATE, @MOVE, @ON, @PREFIX, @SUFFIX

OFF — Schaltet den CHECK-Modus aus. Die Überprüfung der Zeilenlänge bleibt davon unberührt.

cl — Gibt den Wert der Zeilenlänge für die Zeilenlängenprüfung an. EDT überprüft die Länge jeder Zeile, die neu eingegeben oder durch eine der folgenden Anweisungen aufgebaut wird: @+, @–, @IF, @ln, @SET, @UPDATE

Ist eine Zeile länger als cl, wird diese Zeile zwar angelegt, aber der EDT macht mit der Meldung CHECK LINE LENGTH darauf aufmerksam, daß die vorgegebene Zeilenlänge überschritten wurde. Der standardmäßig vorgegebene Wert von cl entspricht dem Maximalwert von 256, der kleinstmögliche Wert von cl ist 1.

Der Wert für cl kann auch durch @TABS verändert werden.

@CHECK ist nur im L-Modus wirksam. Ein vorübergehender Wechsel in den F-Modus schaltet den CHECK-Modus aus.

Wird nur ON oder OFF angegeben, wird der gerade belegte Wert von cl nicht verändert. Wird lediglich cl angegeben, wird der momentane CHECK-Modus beibehalten.

Es ist möglich, mit @CHECK OFF,cl den gerade gültigen Wert von cl zu verändern. Wird cl nicht angegeben, bleibt der zuletzt gültige Wert von cl bestehen; cl wird also durch @CHECK OFF nicht auf den Standardwert 256 zurückgesetzt. Wird dies gewünscht, muß @CHECK OFF, 256 eingegeben werden.

@CLOSE Schließen und Schreiben einer Datei oder eines Bibliothekselementes

Mit @CLOSE wird

– die zuvor real eröffnete ISAM-Datei geschlossen
– die aktuelle Arbeitsdatei auf Platte oder Band geschrieben und das Bibliothekselement geschlossen
– eine zuvor mit @XOPEN eröffnete POSIX-Datei geschlossen

Die Arbeitsdatei wird gelöscht.

Vor der Eingabe von @CLOSE muß eine Datei oder ein Bibliothekselement mit @OPEN, bzw. @XOPEN eröffnet worden sein.

Operation	Operanden	F-Modus / L-Modus
@**CL**OSE	[**N**OWRITE]	

NOWRITE Die Arbeitsdatei wird gelöscht und nicht zurückgeschrieben. Die eröffnete Datei bzw. das Bibliothekselement wird unverändert geschlossen. Bei einer real eröffneten Datei in der Arbeitsdatei 0 (siehe @OPEN, Format 1) ist NOWRITE wirkungslos.

Ohne Operanden bewirkt @CLOSE bei

– realer Bearbeitung
das Schließen der mit @OPEN real eröffneten Datei, das Löschen der Arbeitsdatei 0 und das Löschen des lokalen Eintrags für den Dateinamen.

– Dateien und Bibliothekselementen, die mit @OPEN Format 2 eröffnet wurden,
das Rückschreiben und Löschen der aktuellen Arbeitsdatei sowie das Schließen der eröffneten Datei bzw. des Bibliothekselementes.

– POSIX-Dateien, die mit @XOPEN eröffnet wurden,
daß die Arbeitsdatei mit dem gleichen Code, mit dem sie eingelesen wurde, wieder in das POSIX-Dateisystem zurückgeschrieben wird.

Etwaige Sekundärschlüssel einer mit @OPEN Format 2 eröffneten NKISAM-Datei werden nach dem Schließen der Datei wieder definiert. Sind die Felder eines Sekundärschlüssels im Datenbereich inkonsistent geändert worden, wird dieser Schlüssel nicht gesetzt und zusätzlich wird eine Fehlermeldung ausgegeben.

Eine mit @OPEN real eröffnete Datei wird auch dann ordnungsgemäß geschlossen, wenn anstatt @CLOSE eine der Anweisungen @HALT, @LOAD, @EXEC oder erneut @OPEN eingegeben wird. Es wird ein implizites @CLOSE ausgeführt.

Nach @CLOSE gibt der EDT alle nicht mehr belegten Speicherplätze frei.

Ausgabe einer neuen Datei-Versionsnummer nach @CLOSE

Nach Angabe der Datei-Versionsnummer (ver) bei @OPEN, Format 1 ohne AS 'file' hat die neue Versionsnummer den Wert der alten Versionsnummer plus 1.

Nach Ausführung des @CLOSE haben sowohl die aktuelle Zeilennummer als auch die Schrittweite den Wert 1. Eventuelle Einträge im 3-stufigen Keller des EDT (siehe @) werden gelöscht.

@CODE Umcodieren von Zeichen

Mit @CODE kann der Benutzer für Zeichen, die an einem bestimmten Bildschirm nicht abbildbar sind, Ersatzdarstellungen festlegen. Er kann in einer Codiertabelle festlegen, welche Codierung ein eingegebenes Zeichen für die Ausgabe am Bildschirm erhalten soll.

Die Codierung gilt nur für die Ein-Ausgabe im Datenfenster und in der Anweisungszeile. Die Markierungsspalte wird nicht umcodiert.

Die Codiertabelle liegt in einer Standardform im Modul CODTAB in der Nachladebibliothek des EDT vor (siehe auch Kapitel „Installationshinweise“ auf Seite 598ff.). Diese Standard-Codiertabelle kann mit @CODE am Bildschirm ausgegeben bzw. verändert werden. Die Codiertabelle wird erst wirksam, nachdem die Codierfunktion mit einem der drei Formate von @CODE eingeschaltet wurde.

Format	Anweisung	Bedeutung
1	@CODE ln,SHOW	Codiertabelle – ausgeben in eine Bildschirmzeile – aus einer Bildschirmzeile in das Modul CODTAB übertragen – ausgeben und ändern Codierfunktion einschalten
2	@CODE ln	Codierfunktion einschalten Dabei enthält der angegebene Satz (ln) die Codiertabelle
3	@CODE SHOW | ON | OFF	Codiertabelle ausgeben Codierfunktion – einschalten – ausschalten

Die Standard-Codiertabelle:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0					␣	&	-									0
1							/		a	j			A	J		1
2									b	k	s		B	K	S	2
3									c	l	t		C	L	T	3
4									d	m	u		D	M	U	4
5									e	n	v		E	N	V	5
6									f	o	w		F	O	W	6
7								~	g	p	x		G	P	X	7
8									h	q	y		H	Q	Y	8
9									i	r	z		I	R	Z	9
A					‘	!	^	:								
B					.	$	,	#	[		{					
C					<	*	%	@	\		|					
D					(	)	_	’	]		}					
E					+	;	>	=								
F							?	„								

Die Codeplätze für nicht darstellbare Zeichen werden mit X’07’ belegt und am Bildschirm als Schmierzeichen dargestellt (DSS 3270: X’41’). Die Codiertabelle sollte eindeutig sein, d.h. eine Codierung außer X’07’ sollte nur einmal vorkommen. Für die Eindeutigkeit der Codiertabelle ist der Benutzer selbst verantwortlich. Sie wird vom EDT nicht überprüft.

Die Standard-Codiertabelle im Modul CODTAB bewirkt folgende Umcodierung:

Tastatur	Zeichen	Datei
X'FB'	(ä) {	X'AB'
X'4F'	(ö) |	X'AC'
X'FD'	(ü) }	X'AD'
X'BB'	(Ä) [	X'8B'
X'BC'	(Ö) \	X'8C'
X'BD'	(Ü)]	X'8D'
X'FF'	(ß) -	X'67'

Der Benutzer kann über Format 2 eine Codiertabelle erstellen, ohne die rechnerinterne Codierung eines Zeichens zu kennen.

Zusammenhang mit LOWER OFF

Ausgabe (Datei → Bildschirm):

Der Satz wird entsprechend der Codiertabelle umgesetzt. Kleinbuchstaben werden für die Ausgabe am Bildschirm in Schmierzeichen umgesetzt. Der Satz wird am Bildschirm ausgegeben.

Eingabe (Bildschirm → Datei):

Die eingegebenen Kleinbuchstaben werden in Großbuchstaben umgesetzt. Es wird entsprechend der gültigen Codiertabelle umcodiert.

i Das Tabulatorsymbol darf nicht umcodiert werden.

@CODE und erweiterte Zeichensätze

Eine gleichzeitige Verwendung von XHCS und @CODE ist nicht sinnvoll. Die @CODE-Anweisung wird jedoch weiterhin unterstützt und koexistiert mit XHCS.

Die @CODE-Anweisung verändert das im EDT eingestellte Coded Character Set (CCS) nicht. Daher wird allen Dateien, auch Bibliothekselementen, der eingestellte Coded Character Set Name (CCSN) als Codemerkmal mitgeliefert. Ebenso erfolgen die Ein-/Ausgaben WRTRD, WROUT und RDATA mit dem eingestellten CCSN.

@CODE (Format 1) Codiertabelle ausgeben und Codierfunktion einschalten

Ein Satz mit einer Codiertabelle wird in übersichtlicher Form dargestellt und kann geändert werden. Die Codierfunktion wird eingeschaltet.

Operation	Operanden	F-Modus / L-Modus
@CODE	ln, **S**HOW	

ln Zeilennummer eines Satzes, der 256 Zeichen lang sein muß. Die Zeilennummer kann auch als Zeilennummervariable oder als symbolische Zeilennummer angegeben werden.
Existiert der Satz mit der angegebenen ln, wird der Satz als Codiertabelle in die Standard-Codiertabelle im Modul CODTAB übertragen.
Existiert kein Satz mit der angegebenen ln, wird ein Satz mit der Zeilennummer ln und der Länge 256 Byte erzeugt. Die Standard-Codiertabelle wird aus dem Modul CODTAB in diesen Satz übertragen.

SHOW Die Codiertabelle wird in übersichtlicher Form dargestellt und kann geändert werden.
Die Codierfunktion wird eingeschaltet.

Dabei ist zu beachten, daß die Codiertabelle eindeutig bleibt, d.h. jedes Zeichen außer NIL darf nur einem Codeplatz zugeordnet werden. Das Zeichen NIL kann mehreren Codeplätzen zugewiesen werden und bedeutet, daß dort die Codierung X'07' (Schmierzeichen am Bildschirm) eingetragen wird.

Die geänderte Codiertabelle wird dann als Satz mit der Zeilennummer ln in die Arbeitsdatei übernommen.

@CODE (Format 2) Codierfunktion einschalten

Die Codierfunktion wird eingeschaltet, wobei der Satz mit der Zeilennummer ln als Codiertabelle verwendet wird.

Operation	Operanden	F-Modus / L-Modus
@CODE	ln	

ln Zeilennummer eines Satzes, der eine Codiertabelle enthält. Der Satz muß vorhanden und 256 Byte lang sein. Die Zeilennummer kann auch als Zeilennummervariable oder als symbolische Zeilennummer angegeben werden.

@CODE (Format 3) Codierfunktion ein- oder ausschalten

Mit diesem Format kann die Codierfunktion ein- oder ausgeschaltet und die aktuelle Codiertabelle ausgegeben und geändert werden bzw. die Codiertabelle gezeigt werden.

Operation	Operanden	F-Modus / L-Modus
@CODE	[**ON**] | **S**HOW | **O**FF	

ON — Schaltet die Codierfunktion mit der aktuell gültigen Codiertabelle ein. Aktuell gültig ist vorerst die Codiertabelle aus dem Modul CODTAB, solange bis sie mit @CODE SHOW abgeändert wird. Dann ist die geänderte Codiertabelle nach @CODE ON gültig.

SHOW — Zeigt die aktuell gültige Codiertabelle, die wie bei Format 1 abgeändert werden kann. Erst durch ein nachfolgendes @CODE ON wird die Codierfunktion eingeschaltet.
Die geänderte Codiertabelle wird beim Beenden des EDT und bei @CODE OFF gelöscht.

OFF — Schaltet die Codierfunktion aus und löscht die aktuelle Codiertabelle.

Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

Beispiel

```
  1.00 MIT DIESER  BUNG M CHTEN WIR IHNEN ERKL REN, WIE SIE.....................
       DCE4CCCECD48CEDC4D8CCECD4ECD4CCDCD4CDDD8DCD64ECC4ECC
       49304952590D245704C3835506990985550592 3B955B06950295
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
  2.00 DIE CODE-ANWEISUNG BEN TZEN K NNEN. .....................................
       CCC4CDCC6CDECCEEDC4CCD8EECD4D8DDCD44
       49503645015659245702 55D395502C5555B0
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
  3.00 .........................................................................
       00000000000000000000000000000000000000000000000000000000000000000000000000
       00000000000000000000000000000000000000000000000000000000000000000000000000
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
code show ; code on.....................................................0001.00:001(0)
```

Der Text in den Zeilen 1.00 und 2.00 enthält Umlaute, die im Datenfenster durch Schmierzeichen dargestellt werden. Der Hexadezimal-Modus ist eingeschaltet.

Mit @CODE SHOW wird die aktuell gültige Codiertabelle am Bildschirm ausgegeben. Diese soll verändert und die Codierfunktion eingeschaltet werden.

```
***                          C O D E - M O D E
*******************************************************************************

   0 1 2 3 4 5 6 7 8 9 A B C D E F
 0  .       ␣ & -                 0
 1              /   a j     A J   1
 2                  b k s   B K S 2
 3                  c l t   C L T 3
 4                  d m u   D M U 4
 5                  e n v   E N V 5
 6                  f o w   F O W 6
 7              ~   g p x   G P X 7
 8                  h q y   H Q Y 8
 9                  i r z   I R Z 9
 A            ` ! ^ :
 B            . $ , # [   {
 C            < * % @ \
 D            ( )   ' ]   }
 E            + ; > =
 F                ? "

*******************************************************************************

 PRESS K1 OR DUE FOR RETURN
```

Die Position 8C der Codiertabelle wird nun mit dem Zeichen \ belegt; somit werden die Umlaute folgendermaßen dargestellt:

```
Ä ⟶ X'8B' ⟶ [
Ö ⟶ X'8C' ⟶ \
Ü ⟶ X'8D' ⟶ ]
```

```
   1.00 MIT DIESER ]BUNG M\CHTEN WIR IHNEN ERKL[REN, WIE SIE.....................
        DCE4CCCECD48CEDC4D8CCECD4ECD4CCDCD4CDDD8DCD64ECC4ECC
        49304952590D245704C38355069909855505923B955B06950295

        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   2.00 DIE CODE-ANWEISUNG BEN]TZEN K\NNEN.....................................
        CCC4CDCC6CDECCEEDC4CCD8EECD4D8DDCD
        4950364501565924570255D395502C5555
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   3.00 .......................................................................
        00000000000000000000000000000000000000000000000000000000000000000000
        00000000000000000000000000000000000000000000000000000000000000000000
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7-
```

Da die Codierfunktion mit @CODE ON eingeschaltet wurde, werden die Umlaute nun durch die Zeichen [, / und] dargestellt.

Ausdruck der beiden umcodierten Zeilen:

```
1.00    MIT DIESER ÜBUNG MÖCHTEN WIR IHNEN ERKLÄREN, WIE SIE
2.00    DIE CODE-ANWEISUNG BENÜTZEN KÖNNEN.
```

@CODENAME Explizites Umschalten des CCSN

Mit @CODENAME wird das gewünschte Coded Character Set (CCS) im EDT eingestellt.

In Systemen, in denen das Subsystem XHCS nicht installiert ist, wird @CODENAME mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@**CODEN**AME	[name]	

name Name des Coded Character Sets (CCSN).

Wird name nicht angegeben, wird auf das Coded Character Set EDF03IRV umgeschaltet.

Die Umschaltung auf das gewünschte Coded Character Set erfolgt, wenn folgende Bedingungen erfüllt sind:

– Der Coded Character Set Name (CCSN) ist in der Liste der gültigen CCSN für die Datensichtstation enthalten.

– In den Arbeitsdateien des EDT sind keine Dateien mit einem anderen CCSN (d.h. alle Arbeitsdateien sind leer).

– Das Coded Character Set (CCS) ist kein ISO-Code und kein anderer 7-Bit-Code als EDF03IRV.

Sind diese Bedingungen nicht erfüllt, wird @CODENAME abgewiesen und der bisher eingestellte CCSN bleibt weiterhin gültig.

Eine Umschaltung auf den aktuell eingestellten CCSN wird ignoriert.

@CODENAME darf nicht in @INPUT- und @DO-Prozeduren angegeben werden.

@COLUMN Text einfügen oder Leerzeichen am Zeilenende löschen

@COLUMN fügt in bereits existierende Zeilen ab der angegebenen Spalte eine Zeichenfolge ein. Dabei kann gewählt werden, ob bereits vorhandener Text überschrieben werden soll oder nicht.

Diese Anweisung löscht auch in einem von rechts nach links verlaufenden Suchvorgang alle Leerzeichen - beginnend bei Spalte 256 - und beendet den Vorgang sofort, wenn eine Spalte ungleich Leerzeichen gefunden wird. Sollte eine Zeile ausschließlich aus Leerzeichen bestehen, wird sie mit dieser Anweisung gelöscht (siehe auch Hinweis, weiter unten).

Operation	Operanden	F-Modus / L-Modus
@**COL**UMN	cl **ON** range { [**C**HANGE] / **I**NSERT } [:] string	

cl — Spalte, ab der Text ersetzt oder eingefügt werden soll.

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennt) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennt) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden. Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

CHANGE — Ab Spalte cl vorhandener Text wird mit der neuen Zeichenfolge überschrieben.

INSERT — Die neue Zeichenfolge wird ab Spalte cl eingefügt.

: — Ist anzugeben, wenn weder CHANGE noch INSERT angegeben ist, um range in eindeutiger Weise von string zu trennen.

string — Zeichenfolge, die vorhandenen Text ersetzen soll bzw. die einzufügen ist. string kann angegeben werden:
- explizite Angabe in Hochkomma,
- implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Wenn die Spalte, ab der Text eingefügt werden soll, hinter dem bisherigen Zeilenende liegt, werden die dazwischenliegenden Spalten mit Leerzeichen aufgefüllt.

- Man kann diese Anweisung zum Löschen von Leerzeilen bzw. zum Löschen von Leerzeichen am Zeilenende verwenden. Wenn zum Beispiel bekannt ist, daß die Zeilen nie über 80 Zeichen lang sind, können mit @COLUMN 81 ON & '␣' alle Leerzeichen gelöscht werden, die an den Zeilenenden stehen. Zunächst wird auf Spalte 81 in jeder Zeile ein Leerzeichen eingefügt; der von rechts nach links verlaufende Löschvorgang für Leerzeichen entfernt es jedoch wieder.
- Das Verschieben bzw. Einfügen ganzer Spaltenblöcke wird von @COLUMN nicht unterstützt. Die Lösung des Problems kann jedoch leicht von einer kleinen EDT-Prozedur übernommen werden (siehe dazu Beispiele im Kapitel „EDT-Prozeduren“ auf Seite 142ff.).

Beispiel

```
   1.00 126790.....................................................................
   2.00 348........................................................................
   3.00 ...........................................................................
```

```
column 3 on 1:2 ..............................................0001.00:001(0)
```

In Zeile 1.00 soll ab Spalte 3 der Inhalt der Zeile 2.00 stehen. Der alte Inhalt von Zeile 1.00 wird somit überschrieben.

```
   1.00 123480.....................................................................
   2.00 348........................................................................
   3.00 ...........................................................................
```

```
column 5 on 1 insert '567'....................................0001.00:001(0)
```

In Zeile 1.00 soll ab Spalte 5 die Zeichenfolge 567 eingefügt werden. Somit wird kein Zeichen von Zeile 1.00 überschrieben.

```
1.00 123456780.....................................................................
2.00 348...........................................................................
3.00 ..............................................................................
```

@COMPARE Vergleichen von Arbeitsdateien

@COMPARE bietet in zwei Formaten die Möglichkeit Arbeitsdateien ganz oder teilweise miteinander zu vergleichen.

@COMPARE (Format 1) Vergleichen zweier Arbeitsdateien

@COMPARE bewirkt, daß der EDT zwei Arbeitsdateien ganz oder teilweise miteinander vergleicht.

Das Ergebnis des Arbeitsdateienvergleichs gibt er wahlfrei aus

– auf den Bildschirm,
– in eine Arbeitsdatei,
– auf den Drucker (SYSLST).

Operation	Operanden	F-Modus / L-Modus
@**COM**PARE	[procnr1] :rng*1 **W**ITH [procnr2] :rng*2 [,[int1] [(int2)] [**L**IST [ln [(inc)]]]]	

procnr1, procnr2
: Nummern der miteinander zu vergleichenden Arbeitsdateien (0 bis 22). Ist eine der Vergleichsdateien die Arbeitsdatei 0 und ist in ihr eine Datei durch @OPEN, Format 1 real eröffnet, so wird @COMPARE mit einer Fehlermeldung abgewiesen. Wird procnr1 oder procnr2 nicht angegeben, wird die aktuelle Arbeitsdatei verwendet.

rng*1
: Zeilenbereich in der ersten Arbeitsdatei (procnr1), bestehend aus:
– einer einzelnen Zeile (z.B. 6)
– mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

rng*2
: Zeilenbereich in der zweiten Arbeitsdatei (procnr2), bestehend aus:
– einer einzelnen Zeile (z.B. 6)
– mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

int1 Gibt an, wieviele Zeilen der EDT in jeder Datei betrachtet, um einen Zeilenbereich zu finden, der in beiden Dateien gleich ist. Findet der EDT innerhalb von int1 Zeilen nicht mindestens int2 aufeinanderfolgende Zeilen, die in beiden Dateien gleich sind, bricht er den Vergleich ab.

int1 muß größer oder gleich int2 sein. Der Standardwert ist 10.

int2 Gibt an, wieviele aufeinanderfolgende Zeilen in beiden Arbeitsdateien gleich sein müssen, damit der EDT Gleichheit zwischen den aus diesen Zeilen bestehenden Bereichen feststellt. int2 muß kleiner oder gleich int1 sein. Der Standardwert ist 1.

LIST Ohne Angabe von ln bewirkt, daß der EDT das Ergebnis des Vergleichs auf den Schnelldrucker (SYSLST) ausgibt. Dabei gibt der EDT von jeder Zeile, für die er keine Übereinstimmung feststellt, außer der Zeilennummer auch die ersten 51 Zeichen des Zeileninhalts aus.

Ist ln angegeben, schreibt der EDT das Ergebnis in die aktuelle Arbeitsdatei, falls sie nicht eine der Vergleichsdateien ist.

Fehlt LIST, gibt der EDT das Ergebnis auf dem Bildschirm aus.

Sind LIST und ln nicht angegeben, gibt der EDT lediglich die Nummern der Zeilen aus, für die er keine Übereinstimmung feststellt. Die Ausgabe der ersten 51 Zeichen des Zeileninhalts unterbleibt.

ln Der EDT schreibt das Vergleichsergebnis in die aktuelle Arbeitsdatei (siehe @LIST). ln gibt die Nummer der Zeile an, in der die erste Zeile des Vergleichsergebnisses stehen soll. Das Format, in dem der EDT das Ergebnis in die Datei schreibt, ist gleich dem, das er bei Ausgabe auf dem Bildschirm benutzt.

inc Schrittweite, aus der sich die auf ln folgenden Zeilennummern ergeben. Fehlt inc, nimmt der EDT die implizit durch ln gegebene Schrittweite.

Der EDT beginnt den Vergleich am Anfang der angegebenen Dateibereiche. Wenn der EDT ein ungleiches Zeilenpaar findet, überspringt er in einer oder in beiden Dateien eine oder mehrere Zeilen. Die Zahl der übersprungenen Zeilen kann in beiden Dateien verschieden sein. Findet der EDT im Anschluß an die übersprungenen Zeilen int2 aufeinanderfolgende Zeilen, die in beiden Dateien gleich sind, richtet er für den weiteren Vergleich die beiden Dateien an diesen Zeilenpaaren aus.

1. Ein Zeilenpaar ist gleich, wenn sowohl Zeileninhalt als auch Zeilenlänge der beiden Zeilen gleich sind. Die Zeilennummern werden beim Dateivergleich nicht berücksichtigt.
2. Wird für int2 der gleiche Wert wie für int1 gewählt, kann der EDT in den beiden Dateien keine übereinstimmende Zeilenbereiche finden, wenn mindestens ein Zeilenpaar nicht identisch ist.
3. Die Ergebnisse des Vergleichs teilt der EDT durch folgende Meldungen mit:
   - Meldungen für Zwischenergebnisse; weitere Meldungen folgen

     ```
     EXTRA LINES IN 1ST FILE
       ln
         .
         .
         .
       ln
     ```

     Der EDT hat die Zeilen, deren Nummern er aufgelistet hat, in der ersten Datei übersprungen. In der zweiten Datei hat er keine Zeile übersprungen. Auf die in der ersten Datei übersprungenen Zeilen folgen int2 Zeilen, die mit int2 aufeinanderfolgenden Zeilen in der zweiten Datei gleich sind.

     1 ≤ Anzahl der aufgelisteten Zeilennummern ≤ int1 - int2

     ```
     EXTRA LINES IN 2ND FILE
       ln
         .
         .
         .
       ln
     ```

     Der EDT hat die Zeilen, deren Nummern er aufgelistet hat, in der zweiten Datei übersprungen. In der ersten Datei hat er keine Zeile übersprungen. Auf die in der zweiten Datei übersprungenen Zeilen folgen int2 Zeilen, die mit int2 aufeinander- folgenden Zeilen in der ersten Datei gleich sind.

     1 ≤ Anzahl der aufgelisteten Zeilennummern ≤ int1 - int2

     ```
     NON-MATCHING LINES
       ln
         .
         .
         .
       ln
     ```

     Der EDT hat die Zeilen, deren Nummern in der ersten Spalte aufgelistet sind, in der ersten Datei übersprungen. Die in der zweiten Spalte aufgelisteten Nummern beziehen sich auf die in der zweiten Datei übersprungenen Zeilen. Auf die in den beiden Dateien übersprungenen Zeilen folgen int2 gleiche Zeilenpaare. 1 ≤ Anzahl der in jeder Spalte aufgelisteten Zeilennummern ≤ int1 - int2.

- Meldungen bei Beendigung des Vergleichs

```
EXTRA LINES IN 1ST FILE
  ln
   .
   .
   .
  ln
REACHED LIMIT ON BOTH FILES
```

Der EDT hat in der ersten Datei int2 aufeinanderfolgende Zeilen gefunden, die mit den letzten int2 Zeilen der zweiten Datei gleich sind. Die Nummern der Zeilen, die in der ersten Datei am Ende des Vergleichs übrig geblieben sind, hat der EDT aufgelistet.

1 ≤ Anzahl der aufgelisteten Zeilennummern ≤ int1

```
EXTRA LINES IN 1ST FILE
  ln
   .
   .
   .
  ln
REACHED 2ND FILE LIMIT
```

Der EDT hat in der ersten Datei int2 aufeinanderfolgende Zeilen gefunden, die mit den letzten int2 Zeilen in der ersten Datei gleich sind. Am Ende des Vergleichs sind in der ersten Datei mehr als int1 Zeilen übriggeblieben. Die Nummern der ersten int1 übriggebliebenen Zeilen hat der EDT aufgelistet.

```
EXTRA LINES IN 2ND FILE
  ln
   .
   .
   .
  ln
REACHED LIMIT ON BOTH FILES
```

Der EDT hat in der zweiten Datei int2 aufeinanderfolgende Zeilen gefunden, die mit den letzten int2 Zeilen in der ersten Datei gleich sind. Die Nummern der Zeilen, die in der zweiten Datei am Ende des Vergleichs übriggeblieben sind, hat der EDT aufgelistet.

1 ≤ Anzahl der aufgelisteten Zeilennummern ≤ int1

```
EXTRA LINES IN 2ND FILE
  ln
   .
   .
   .
  ln
REACHED 1ST FILE LIMIT
```

Der EDT hat in der zweiten Datei int2 aufeinanderfolgende Zeilen gefunden, die mit den letzten int2 Zeilen der ersten Datei gleich sind. Am Ende des Vergleichs sind in der zweiten Datei mehr als int1 Zeilen übriggeblieben. Die Nummern der ersten int1 übriggebliebenen Zeilen hat der EDT aufgelistet.

```
NON-MATCHING LINES
  ln       ln
   .        .
   .        .
   .        .
  ln       ln
NOTHING SEEMS TO MATCH
```

Die Nummern der int1 Zeilen, die der EDT in jeder der beiden Dateien zuletzt betrachtet hat, werden aufgelistet. Unter ihnen gibt es keine int2 aufeinanderfolgende Zeilen, die in beiden Dateien gleich sind. Der EDT bricht den Dateivergleich ab. Er hat bei keiner der beiden Dateien das Endes des zu vergleichenden Bereichs erreicht.

```
NON-MATCHING LINES
  ln       ln
   .        .
   .        .
   .        .
  ln       ln
REACHED LIMIT ON BOTH FILES
```

Der EDT hat in beiden Dateien das Ende der miteinander zu vergleichenden Zeilenbereiche erreicht. In der ersten Spalte stehen die Nummern der letzten Zeilen aus der ersten Datei, in der zweiten Spalte der letzten Zeilen aus der zweiten Datei. Unter ihnen gibt es keine int2 aufeinanderfolgende Zeilen, die in beiden Dateien gleich sind.

1 ≤ Anzahl der in jeder Spalte aufgelisteten Zeilennummern ≤ int1

```
NON-MATCHING LINES
  ln       ln
   .        .
   .        .
   .        .
  ln       ln
REACHED 1ST FILE LIMIT
```

Der EDT hat das Ende des betrachteten Zeilenbereichs in der ersten Datei erreicht. In der ersten Spalte hat er die Nummern der letzten Zeilen aus der ersten Datei aufgelistet. Unter diesen gibt es keine int2 aufeinanderfolgende Zeilen, die mit int2 aufeinanderfolgenden Zeilen der zweiten Datei gleich sind (bezogen auf die int1 Zeilen der zweiten Datei, deren Nummern in der zweiten Spalte aufgelistet sind).

```
NON-MATCHING LINES
  ln       ln
   .        .
   .        .
   .        .
  ln       ln
REACHED 2ND FILE LIMIT
```

Der EDT hat das Ende des betrachteten Zeilenbereichs in der zweiten Datei erreicht. In der zweiten Spalte hat er die Nummern der letzten Zeilen aus der zweiten Datei aufgelistet. Unter diesen gibt es keine int2 aufeinanderfolgende Zeilen, die mit int2 aufeinanderfolgenden Zeilen der ersten Datei gleich sind (bezogen auf die int1 Zeilen der ersten Datei, deren Nummern in der ersten Spalte aufgelistet sind).

$1 \leq$ Anzahl der in der zweiten Spalte aufgelisteten Zeilennummern $\leq$ int1

```
REACHED LIMIT ON BOTH FILES AT SAME TIME
```

Der EDT hat in beiden Dateien das Ende der miteinander zu vergleichenden Zeilenbereiche erreicht. Die letzten int2 Zeilen sind in beiden Dateien gleich.

Beispiel

```
   1.     @PROC 1
   1.     @READ 'PROC-DATEI.1' ---------------------------------------- (01)
   7.     @PRINT
   1.0000 AAAAAA
   2.0000 BBBBBB
   3.0000 CCCCCC
   4.0000 UUUUUU
   5.0000 VVVVVV
   6.0000 WWWWWW
   7.     @END
   1.     @PROC 2
   1.     @READ 'PROC-DATEI.2' ---------------------------------------- (02)
   8.     @PRINT
   1.0000 AAAAAA
   2.0000 BBBBBB
   3.0000 ZZZZZZ
   4.0000 AAAAAA
   5.0000 BBBBBB
   6.0000 CCCCCC
   7.0000 UUUUUU
   8.     @END
   1.     @COMPARE 1:1-6 WITH 2:1-7, 5(2) ------------------------------ (03)
EXTRA LINES IN 2ND FILE
               3.0000
               4.0000
               5.0000
EXTRA LINES IN 1ST FILE
   5.0000
   6.0000
REACHED LIMIT ON BOTH FILES
   1.     @COMPARE 1:1-6 WITH 2:1-7, 5(3) ------------------------------ (04)
NON-MATCHING LINES
   1.0000      1.0000
   2.0000      2.0000
   3.0000      3.0000
   4.0000      4.0000
   5.0000      5.0000
NOTHING SEEMS TO MATCH
   1.     @COMPARE 1:1-6 WITH 2:1-7, 6(3) ------------------------------ (05)
EXTRA LINES IN 2ND FILE
               1.0000
               2.0000
               3.0000
EXTRA LINES IN 1ST FILE
   5.0000
```

```
   6.0000
REACHED LIMIT ON BOTH FILES
   1.
```

(01) Die SAM-Datei PROC-DATEI.1 wird in die Arbeitsdatei 1 eingelesen.

(02) Die SAM-Datei PROC-DATEI.2 wird in die Arbeitsdatei 2 eingelesen.

(03) Lassen sich bei der Betrachtung von jeweils 5 Zeilen in den beiden Dateien nicht mindestens 2 aufeinanderfolgende gleiche Zeilenpaare finden , soll der Vergleich abgebrochen werden. Der Vergleich wird bis zum Ende beider Dateien durchgeführt.

(04) Das unter (03) gegebene @COMPARE wird leicht modifiziert noch einmal gegeben. Es müssen jetzt mindestens 3 aufeinanderfolgende gleiche Zeilenpaare gefunden werden. Diesmal bricht der EDT den Vergleich ab.

(05) Das unter (04) gegebene @COMPARE wird leicht modifiziert noch einmal gegeben. Der Vergleich soll jetzt erst nach Betrachtung von 6 Zeilen abgebrochen werden. Er wird bis zum Ende durchgeführt.

@COMPARE (Format 2) Vergleichen zweier Arbeitsdateien zeilenweise

Mit @COMPARE werden die Inhalte zweier Arbeitsdateien zeilenweise verglichen. Das Vergleichsergebnis legt der EDT in einer Arbeitsdatei ab. Diese wird vor dem Ablegen des Ergebnisses gelöscht.

Als Vergleichsergebnis gibt der EDT aus:

- eine Überschriftenzeile mit dem Dateinamen. In der Überschriftenzeile wird mitausgegeben:
  - der Name, eines mit @OPEN Format 2 eröffneten Bibliothekselementes, bzw .einer Datei
  - der Name einer mit @XOPEN eröffneten POSIX-Datei
  - ein lokaler @FILE-Eintrag, wenn vorhanden
- die Zeilennummer und den Inhalt aller Sätze, die nur in einer der beiden Vergleichsdateien vorkommmen. Sätze, die länger als 239 Zeichen sind, werden abgeschnitten. Dabei gibt die Stellung der Zeilennummer in Spalte 1 oder in Spalte 2 unter der Überschrift LINE#(adatnr) an, in welcher der beiden Vergleichsdateien der Satz steht.
- die Zeilennummern von Sätzen, deren Inhalt gleich ist (z.B. 0001.00=0006.00). Sind mehrere aufeinanderfolgende Sätze gleich (Bereich gleicher Sätze), wird nur das erste und letzte Zeilennummernpaar des Bereiches angegeben (Näheres siehe Beispiel).

Operation	Operanden	F-Modus / L-Modus
@**COMP**ARE	[[procnr1] [**W**ITH]] procnr2 [**L**IST [procnr3]] [,procnr4]	

procnr1 Nummer der ersten Arbeitsdatei, die verglichen werden soll.

Ist procnr1 nicht angegeben, wird die aktuelle Arbeitsdatei verglichen.

procnr2 Nummer der zweiten Arbeitsdatei, mit der verglichen wird. procnr2 muß angegeben werden.

LIST Bei Angabe von LIST wird das Ergebnis in die Arbeitsdatei procnr3 oder bei fehlender Angabe von procnr3 nach SYSLST ausgegeben.
Ist LIST nicht angegeben, wird das Ergebnis im F-Modus in die Arbeitsdatei 9, im L-Modus am Bildschirm und bei Prozeduren auf SYSOUT ausgegeben.

procnr3 Arbeitsdatei, in der das Vergleichsergebnis abgelegt wird. Sie wird vor der Verwendung gelöscht.

procnr4 Arbeitsdatei, welche der EDT als Hilfsdatei verwenden soll. Sie wird vor der Verwendung gelöscht. Fehlt procnr4, wird die Arbeitsdatei 10 verwendet.

Die Arbeitsdateien procnr1, procnr2, procnr3 und procnr4 müssen verschieden sein.

Sind alle zu vergleichenden Zeilen gleich bzw. ungleich, werden folgende Meldungen ausgegeben:

% EDT0291 ALL LINES ARE EQUAL bei Gleichheit aller Zeilen
% EDT0290 ALL LINES ARE DIFFERENT bei Ungleichheit aller Zeilen

Wir das Ergebnis in procnr3 ausgegeben, wird folgende Meldung ausgegeben:

% EDT0297 COMPARE RESULT IN WORK FILE (procnr3)

Ist eine der beiden Vergleichsdateien die Arbeitsdatei, in der das Ergebnis stehen soll, wird folgende Meldung ausgegeben:

% EDT5350 COMPARE RESULT CANNOT BE SHOWN

Das Vergleichsergebis kann nicht ausgegeben werden, weil die angegebene Ergebnisdatei belegt ist.

Ist eine der Vergleichsdateien die Arbeitsdatei 0, darf keine ISAM-Datei durch @OPEN, Format 1 real eröffnet sein.

Durch @COMPARE werden alle Satzmarkierungen gelöscht.

Abfragen von Vergleichsergebnissen

Um in Prozeduren auf das Vergleichsergebnis abfragen zu können, wird zusätzlich zu den Meldungen % EDT0290 und % EDT0297 der EDT-Fehlerschalter gesetzt (Abfragen von Fehlerschalter siehe @IF Formate 1 und 3).

Unterscheidung:

	EDT-Fehlerschalter	**Arbeitsdatei procnr3**
% EDT0291	nicht gesetzt	leer
% EDT0290	gesetzt	leer
% EDT0297	gesetzt	nicht leer

Vor dem Vergleichen mit @COMPARE muß der EDT-Fehlerschalter mit @RESET zurückgesetzt werden und die Arbeitsdatei procnr3 gelöscht werden.

Ist bei einer der Dateien die vierte Nachkommastelle einer Zeilennummer ungleich 0 (z.B.: 0.0009), dann wird @COMPARE mit der Meldung % EDT5352 @COMPARE ABORTED - PLEASE RENUMBER abgebrochen, da diese Stelle der Zeilennummer intern verwendet wird.

Beispiel

```
   1.00 X.....................................................................
   2.00 Y.....................................................................
   3.00 Z.....................................................................
   4.00 G.....................................................................
   5.00 H.....................................................................
   6.00 A.....................................................................
   7.00 B.....................................................................
   8.00 C.....................................................................
   9.00 J.....................................................................
  10.00 K.....................................................................
  11.00 D.....................................................................
  12.00 E.....................................................................
  13.00 .....................................................................
  14.00 .....................................................................
  15.00 .....................................................................
  16.00 .....................................................................
  17.00 .....................................................................
  18.00 .....................................................................
  19.00 .....................................................................
  20.00 .....................................................................
  21.00 .....................................................................
  22.00 .....................................................................
  23.00 .....................................................................
1 ...........................................................0001.00:001(2)
```

Aus der Arbeitsdatei 2 wird in die Arbeitsdatei 1 gewechselt.

```
   1.00 A.....................................................................
   2.00 B.....................................................................
   3.00 C.....................................................................
   4.00 D.....................................................................
   5.00 E.....................................................................
   6.00 F.....................................................................
   7.00 G.....................................................................
   8.00 H.....................................................................
   9.00 I.....................................................................
  10.00 J.....................................................................
  11.00 K.....................................................................
  12.00 .....................................................................
  13.00 .....................................................................
  14.00 .....................................................................
  15.00 .....................................................................
  16.00 .....................................................................
  17.00 .....................................................................
  18.00 .....................................................................
  19.00 .....................................................................
  20.00 .....................................................................
  21.00 .....................................................................
  22.00 .....................................................................
  23.00 .....................................................................
compare 2 with 1 liste 3;  3...............................0001.00:001(1)
```

Arbeitsdatei 2 wird mit Arbeitsdatei 1 verglichen und das Ergebnis in Arbeitsdatei 3 abgelegt. Anschließend wird in die Arbeitsdatei 3 gewechselt.

```
   0.10 LINE#( 1)          FILENAME:.....................................
   0.20          LINE#( 2) FILENAME:.....................................
   0.30         0001.00  X.....................................................
   0.40         0002.00  Y.....................................................
   0.50         0003.00  Z.....................................................
   0.60         0004.00  G.....................................................
   0.70         0005.00  H.....................................................
   0.80 0001.00=0006.00.....................................................
   0.90 0003.00=0008.00.....................................................
   1.00 0004.00          D.....................................................
   1.10 0005.00          E.....................................................
   1.20 0006.00          F.....................................................
   1.30 0007.00          G.....................................................
   1.40 0008.00          H.....................................................
   1.50 0009.00          I.....................................................
   1.60 0010.00=0009.00.....................................................
   1.70 0011.00=0010.00.....................................................
   1.80         0011.00  D.....................................................
   1.90         0012.00  E.....................................................
   2.90 .....................................................................
   3.90 .....................................................................
   4.90 .....................................................................
%  EDT0297 RESULT OF COMPARE IN PROCFILE 3
.....................................................................0000.10:001(3)
```

Das Vergleichsergebnis der Arbeitsdateien 1 und 2 ist in der Arbeitsdatei 3 abgelegt.

@CONTINUE Sprungmarke definieren

@CONTINUE wird verwendet, um in EDT-Prozeduren eine Zeile zu erzeugen, die über @GOTO angesprungen werden kann. Die Anweisung verursacht bei ihrer Ausführung keine Aktion. Sie kann deshalb auch zur Kommentierung von EDT-Prozeduren verwendet werden (wie @NOTE).

Operation	Operanden	L-Modus / @PROC
@**CON**TINUE	[comment]	

comment Beliebiger Kommentar.

Die Hauptanwendung dieser Anweisung ist die Definition einer letzten Zeile innerhalb einer EDT- oder INPUT-Prozedur. Sie wird nur dann zwingend benötigt, wenn eine EDT-Prozedur als äußere Schleife über ein Schleifensymbol aufgerufen wird (z.B. @DO 5,!=%,$). Dann nämlich muß immer an das Ende der Prozedur verzweigt werden, um den nächsten Durchlauf zu starten.

Beispiel

```
6.     @PRINT
1.0000 MIT DEM EDT
2.0000 KANN MAN
3.0000 AUF EINFACHE WEISE
4.0000 PROZEDUR UM PROZEDUR
5.0000 SCHREIBEN
6.     @PROC 1
1.     @1.00
1.00   @ @CON AUFGABE: WENN EINE ZEILE EIN 'M' ---------------------- (01)
1.01   @ @CON          ENTHAELT, SO IST SIE AUSZUGEBEN
1.02   @ @ON * FIND 'M'
1.03   @ @IF .FALSE. GOTO 2
1.04   @ @PRINT *
1.05   @2.00
2.00   @ @CONTINUE ------------------------------------------------- (02)
2.01   @END
6.     @DO 1,*=1,$ ------------------------------------------------- (03)
1.0000 MIT DEM EDT
2.0000 KANN MAN
4.0000 PROZEDUR UM PROZEDUR
6.
```

(01) Hier wird @CONTINUE zur Kommentierung verwendet.

(02) Hier wird @CONTINUE benötigt, da es eine letzte Zeile in der Prozedur geben muß, die angesprungen werden kann.

(03) Über @DO mit Schleifensymbol wird die Prozedur in Arbeitsdatei 1 angestoßen.

@COPY Kopieren

@COPY bietet in zwei Formaten folgende Kopiermöglichkeiten:

– Kopieren einer Zeile oder eines zusammenhängenden Zeilenbereiches aus einer beliebigen Arbeitsdatei in die aktuelle Arbeitsdatei (Format 1)
– Kopieren eines Programm-Bibliothekselementes (Format 2).

Bei @COPY bleibt - im Gegensatz zu @MOVE - der zu kopierende Zeilenbereich (Sendebereich) erhalten.

@COPY (Format 1) Kopieren einer Zeile oder eines Zeilenbereichs

Es wird eine Zeile oder ein zusammenhängender Zeilenbereich aus einer beliebigen Arbeitsdatei in die aktuelle Arbeitsdatei kopiert.

Es kann nicht aus einer Arbeitsdatei kopiert werden, die gerade als EDT-Prozedur (siehe @DO) abgearbeitet wird (aktive Arbeitsdatei).

Operation	Operanden	F-Modus / L-Modus
@**C**OPY	rng [(procnr)] [**T**O ln1 [(inc)] [:] [ln2]] [,...]	

Wird aus der aktuellen Arbeitsdatei kopiert, müssen die Operanden TO und ln1 immer angegeben werden. Wird aus einer anderen Arbeitsdatei kopiert, werden bei Weglassen dieses Operanden die Zeilennummern des Sendebereichs beibehalten.

rng Zeilenbereich, bestehend aus:
– einer einzelnen Zeile (z.B. 6)
– mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

Die symbolischen Zeilennummern beziehen sich auf die aktuelle Arbeitsdatei, d.h. die Werte der symbolischen Zeilennummern entsprechen den Zeilennummern der aktuellen Arbeitsdatei und nicht der Arbeitsdatei, aus der kopiert wird.

procnr Nummer der Arbeitsdatei (0-22), aus der kopiert wird.

ln1 Nummer der ersten Zeile des Empfangsbereichs.
Die Nummern der folgenden Zeilen des Empfangsbereichs errechnet der EDT, indem er die jeweilige Zeilennummer um die für den Empfangsbereich geltende Schrittweite erhöht. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
Fehlt inc, wird mit ln1 implizit die Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.
ln1 darf auch durch Zeilennummervariablen oder symbolisch angegeben werden.

inc aktuelle Schrittweite des Empfangsbereichs.
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.

: Dieses Trennzeichen kann entfallen, wenn inc angegeben wurde und dadurch ln1 und ln2 eindeutig voneinander getrennt sind.

ln2 Nummer der letzten Zeile des Empfangsbereichs.
@COPY kopiert zeilenweise. Beim Erreichen dieser Obergrenze wird der Kopiervorgang beendet, unabhängig davon, ob noch zu kopierende Zeilen vorhanden sind oder nicht. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. ln2 darf auch durch Zeilennummervariablen oder symbolisch angegeben werden.
Wird ln2 nicht angegeben, können unbeabsichtigt Zeilen überschrieben werden.

Vervielfachen von Zeilenbereichen

Mit @COPY kann man Zeilenbereiche vervielfachen, wenn sich Sende- und Empfangsbereich überlappen (siehe Beispiel 2).

Wird inc zu groß gewählt oder wird ln2 nicht angegeben, können im Empfangsbereich Zeilen überschrieben werden.

Übertragen mit Beibehalten der Zeilennumern

Beim Übertragen aus anderen Arbeitsdateien in die aktuelle Arbeitsdatei bleiben die Zeilennummern erhalten, wenn TO ln1 ... nicht angebenen wird.

Beim Übertragen aus der aktuellen Arbeitsdatei ist TO ln1 immer anzugeben.

Aktuelle Schrittweite und Zeilennummer

Die aktuelle Schrittweite wird von @COPY nicht verändert. inc bestimmt nur die Schrittweite zwischen den kopierten Sätzen. Es bezieht sich nicht auf die aktuelle Schrittweite.

Die aktuelle Zeilennummer wird nur dann verändert, wenn eine Zeile angelegt wird, deren Nummer größer als die bisher höchste Zeilennummer ist.

Beispiel 1

```
   1.00 JETZT.....................................................................
   2.00       WIRD.....................................................................
   3.00            KOPIERT.....................................................
   4.00 .....................................................................
```

```
copy 1 to 7 ; copy 2 to 5 ; copy 1-3 to 30.1 (5).................0001.00:001(0)
```

Mit den drei @COPY soll folgendermaßen kopiert werden:

Zeile 1 → Zeile 7
Zeile 2 → Zeile 5 und
Zeilenbereich von Zeilennummer 1 bis 3
→ ab Zeile 30.1 mit der expliziten Schrittweite 5

```
   1.00 JETZT.....................................................................
   2.00       WIRD.....................................................................
   3.00            KOPIERT.....................................................
   5.00       WIRD.....................................................................
   7.00 JETZT.....................................................................
  30.10 JETZT.....................................................................
  35.10       WIRD.....................................................................
  40.10            KOPIERT.....................................................
  41.10 .....................................................................
```

Beispiel 2

Mit @COPY können Zeilenbereiche vervielfacht werden, wenn sich Sende- und Empfangsbereich überlappen. Nachfolgend soll die erste Zeile vervielfacht werden.

```
   1.00 111.....................................................................
   2.00    222..................................................................
   3.00       333...............................................................
   4.00          444............................................................
   5.00             555.........................................................
   6.00 .....................................................................




copy 1-2 to 1.5.....................................................0001.00:001(0)
```

Mit dieser Anweisung wird der Zeilenbereich von Zeilennummer 1 bis 2 in den Bereich ab Zeilennummer 1.5 mit der impliziten Schrittweite 0.1 kopiert.
Dabei kopiert der EDT zunächst die Zeile 1 in die Zeile 1.5. Diese Zeile liegt im angegebenen Sendebereich. Daher wird die Zeile 1.5 auf die Zeile 1.6 kopiert (implizite Schrittweite 0.1). Entsprechend wird Zeile 1.6 auf 1.7, ... , 1.9 auf 2.0 (dabei wird der Inhalt der Zeile 2 überschrieben) und abschließend Zeile 2.0 auf Zeile 2.1 kopiert.

```
   1.00 111.....................................................................
   1.50 111.....................................................................
   1.60 111.....................................................................
   1.70 111.....................................................................
   1.80 111.....................................................................
   1.90 111.....................................................................
   2.00 111.....................................................................
   2.10 111.....................................................................
   3.00       333...............................................................
   4.00          444............................................................
   5.00             555.........................................................
   6.00 .....................................................................
   7.00 .....................................................................
   8.00 .....................................................................
   9.00 .....................................................................
  10.00 .....................................................................
  11.00 .....................................................................
  12.00 .....................................................................
  13.00 .....................................................................
  14.00 .....................................................................
  15.00 .....................................................................
  16.00 .....................................................................
  17.00 .....................................................................
copy 3-5 to 4.1 : 5.....................................................0001.00:001(0)
```

Der Zeilenbereich 3 bis 5 soll in den Bereich von 4.1 bis 5 mit der impliziten Schrittweite 0.1 kopiert werden.

Dabei kopiert der EDT zunächst die Zeile 3 in die Zeile 4.1 und die Zeile 4 in die Zeile 4.2. Beide neuen Zeilen liegen im angegebenen Sendenbereich. Daher wird anschließend die Zeile 4.1 in die Zeile 4.3, die Zeile 4.2 auf 4.4, ... , 4.8 auf 5.0 (dabei wird der Inhalt der Zeile 5 überschrieben). Die Zeilen 4.9 und 5.0 werden nicht mehr kopiert, da die angegebene Obergrenze des Empfangsbereichs bereits erreicht wurde.

```
   1.00 111.....................................................................
   1.50 111.....................................................................
   1.60 111.....................................................................
   1.70 111.....................................................................
   1.80 111.....................................................................
   1.90 111.....................................................................
   2.00 111.....................................................................
   2.10 111.....................................................................
   3.00       333...............................................................
   4.00          444............................................................
   4.10       333...............................................................
   4.20          444............................................................
   4.30       333...............................................................
   4.40          444............................................................
   4.50       333...............................................................
   4.60          444............................................................
   4.70       333...............................................................
   4.80          444............................................................
   4.90       333...............................................................
   5.00          444............................................................
   6.00 .......................................................................
   7.00 .......................................................................
   8.00 .......................................................................
.....................................................................0001.00:001(0)
```

@COPY (Format 2) Kopieren eines Bibliothekselementes oder einer Datei

Bibliothekselemente und Dateien werden komplett in die aktuelle Arbeitsdatei kopiert. Nach dem Kopieren wird das Bibliothekselement bzw. die Datei wieder geschlossen.

Operation	Operanden	F-Modus / L-Modus
@**C**OPY	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2 } [{ **B**EFORE **A**FTER } ln]	

LIBRARY = path1 ([E[LEMENT]=]elemname [(vers)][,elemtyp])
: Name des Elements mit Angabe des Bibliotheknamens.

ELEMENT = elemname [(vers)][,elemtyp]
: Namen des Elements ohne Angabe des Bibliotheknamens. Voraussetzung ist die Voreinstellung des Bibliotheknamens mit @PAR.

path1
: Name der Bibliothek. path1 kann auch über Zeichenfolgevariable angegeben werden.
Wird path1 nicht angegeben, wird die mit @PAR LIBRARY voreingestellte Bibliothek verwendet.

elemname
: Name des Elements. elemname kann auch über Zeichenfolgevariable angegeben werden.

vers
: Versionsbezeichnung des gewünschten Elements (siehe Handbuch „LMS“ [13]). Wird vers nicht angegeben, oder *STD, wird die höchste vorhandene Version des Elementes gewählt.

elemtyp
: Typ des Elements. elemtyp kann auch über Zeichenfolgevariable angegeben werden.
Zulässige Typangaben: S, M, P, J, D, X, *STD und freie Typnamen mit entsprechendem Basistyp. Falls nicht angegeben, wird der in @PAR ELEMENT-TYPE voreingestellte Wert verwendet.

Wird ein freier Typnamen verwendet, so liegt es in der Verantwortung des Benutzers, daß der zugehörige Basistyp einem zulässigen Typ S, M, P, J, D oder X entspricht.

Typ	Elementinhalt
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats

*STD Voreinstellung
Typ S ist die Voreinstellung nach Aufruf des EDT. Mit @PAR kann eine andere zulässige Typangabe als Voreinstellung festgelegt werden.

FILE = path2 Kopieren einer BS2000-Datei.

path2 Name der Datei, die kopiert werden soll. path2 kann auch über Zeichenfolgevariable angegeben werden.

BEFORE Das Bibliothekselement bzw. die Datei wird vor der angegebenen Zielposition eingefügt.
Bei @PAR RENUMBER=ON werden die bereits existierenden Zeilennummern soweit nötig umnumeriert.
Bei @PAR RENUMBER=OFF ist ein Kopieren vor die Zeilennummer 0.01 nicht möglich.

AFTER Das Bibliothekselement bzw. die Datei wird nach der angegebenen Zielposition eingefügt.
Hinter die Zeilennummer 9999.99 können keine Zeilen kopiert werden.

ln Nummer der ersten Zeile des Empfangsbereichs.

Die Nummern der folgenden Zeilen des Empfangsbereichs errechnet der EDT, indem er die jeweilige Zeilennummer um die für den Empfangsbereich geltende Schrittweite erhöht. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
Mit ln wird implizit die Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.
ln kann auch durch Zeilennummervariablen oder symbolische Zeilennummern angegeben werden. Wird ln nicht angegeben, werden die kopierten Sätze an das Ende der aktuellen Arbeitsdatei kopiert.

Berechnung der Zeilennummern

Die Datensätze werden beim Einfügen nach drei Methoden numeriert:

1. Standardnumerierung mit Standardschrittweite 1.0000
   (z.B. 21.0000, 22.0000, 23.0000 ... 99.0000) oder
2. Numerierung mit festgelegten Schrittweiten
   gemäß @PAR INCREMENT oder
3. Automatische Numerierung und Umnumerierung,
   wenn die Schrittweite zu groß gewählt wurde, um die einzufügenden Datensätze aufnehmen zu können. Der EDT wählt dann eine Schrittweite, die um Faktor 10 kleiner ist als die Standardschrittweite (1.) bzw. festgelegte Schrittweite (2.). Mit der kleineren Schrittweite wird versucht, die einzufügenden Sätze zu numerieren.
   Dieser Vorgang wird solange wiederholt, bis die kopierten Sätze erfolgreich eingelesen werden können oder der EDT die minimale Schrittweite von 0.01 gewählt hat.

   Umnumerierung bei @PAR RENUMBER=ON:

   Wenn mit der minimalen Schrittweite 0.01 die kopierten Datensätze nicht eingefügt werden können, numeriert der EDT automatisch die Zeilennummern der hinter dem Zielort bereits bestehenden Sätze mit der Schrittweite 0.01 um.

   Kann nicht genügend Platz gefunden werden, wird kein Satz eingefügt und eine Fehlermeldung ausgegeben. Die Zeilennummer des ersten Datensatzes beim Kopieren in eine leere Datei errechnet der EDT aus 0 + Standardschrittweite bzw. 0 + festgelegte Schrittweite (@PAR INCREMENT).

Bei @PAR INCREMENT mit einer Schrittweite < 0.01 ist zu beachten, daß Zeilennummern von eingelesenen, kopierten oder eingefügten Zeilen im F-Modus nicht vollständig ausgegeben werden (6-stellige Zeilennummernanzeige).
Werden diese unvollständig ausgegebenen Zeilennummern in @COPY verwendet, kann dies zu unvorhersehbaren Ergebnissen führen.

Wird eine Zeile angelegt, deren Nummer größer als die bisher höchste Zeilennummer ist, so wird die aktuelle Zeilennummer verändert.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @COPY der Coded Character Set Name (CCSN) der Datei (Bibliothekselement) berücksichtigt.

Die @COPY-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei (Bibliothekselement) gleich dem im EDT aktuell eingestellten ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

Beispiel

COPY L = MACLIB (E=XYZ,M) AFTER 12.3

Das Element XYZ der Macrobibliothek MACLIB wird vollständig in die aktuelle Arbeitsdatei nach der Zeile 0012.3000 kopiert.

COPY E = PERSONAL (@), D

Aus einer Programm-Bibliothek, die mit PAR L=libname bereits zugewiesen ist, wird das Bibliothekselement PERSONAL mit der höchstmöglichen Versionsnummer (siehe Handbuch „LMS“ [13]) an das Ende der aktuellen Arbeitsdatei kopiert. PERSONAL ist ein Element des Typs D.

@CREATE Textzeilen erzeugen

Mit @CREATE kann man eine beliebige Zeichenfolge in eine beliebige Zeile oder Zeichenfolgevariable schreiben.

Die Zeichenfolge kann

– in der Anweisung angegeben werden (Format 1) oder
– vom Bildschirm eingegeben werden (Format 2).

@ CREATE (Format 1) Erzeugen von Zeilen

Mit diesem Format der Anweisung wird eine angegebene Zeichenfolge in eine Zeile oder Zeichenfolgevariable geschrieben. Eine bereits bestehende Zeile bzw. der Inhalt der Zeichenfolgevariable wird überschrieben. Im Gegensatz zu @SET, Format 6 verändert @CREATE nicht die aktuelle Zeilennummer, auch dann nicht, wenn damit eine höhere Zeilennummer erzeugt wird.

Operation	Operanden	F-Modus / L-Modus
@**CR**EATE	line [:] [string[,...]]	

line — Zeilennummer, in die die Zeichenfolge geschrieben werden soll.
Es kann auch eine Zeichenfolge- oder Zeilennummernvariable angegeben werden.

: — Ist nur dann anzugeben, wenn line nicht eindeutig von string zu trennen ist.

string[,...] — Zeichenfolge, die in eine Zeile oder Zeichenfolgevariable geschrieben wird. string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Die genannten Möglichkeiten können beliebig miteinander kombiniert werden.

Wird string nicht angegeben, wird eine Zeile, bestehend aus einem Leerzeichen, erzeugt. Wird string mehrmals angegeben, wird in der angegebenen Reihenfolge verkettet.

Tabulatorzeichen werden von @CREATE nicht verarbeitet.

Beispiel

```
   1.00 DIESES IST DIE ERSTE ZEILE.....................................
   2.00 DIESES IST DIE ZWEITE ZEILE....................................
   3.00 ...............................................................
```

```
create 3 'zeile 3 wird mit @create angelegt'.....................0001.00:001(0)
```

Zeile 3 wird mit @CREATE neu angelegt.

```
   1.00 DIESES IST DIE ERSTE ZEILE.....................................
   2.00 DIESES IST DIE ZWEITE ZEILE....................................
   3.00 ZEILE 3 WIRD MIT @CREATE ANGELEGT...............................
   4.00 ...............................................................
```

```
create 3:3, ' und wird laenger'.................................0001.00:001(0)
```

Zeile 3 wird neu angelegt aus dem Inhalt der alten Zeile 3 verkettet mit dem neuen Text UND WIRD LAENGER.

```
   1.00 DIESES IST DIE ERSTE ZEILE.....................................
   2.00 DIESES IST DIE ZWEITE ZEILE....................................
   3.00 ZEILE 3 WIRD MIT @CREATE ANGELEGT.UND WIRD LAENGER..............
   4.00 ...............................................................
```

```
create 4:1, ' verkettet mit.',2,1; edit long on..................0001.00:001(0)
```

Die Zeile 4 wird neu angelegt, und zwar besteht sie aus der Verkettung der Zeile 1, des Textes VERKETTET MIT und den Zeilen 2 und 1 in dieser Reihenfolge.

Um den Inhalt der Zeile 4 vollständig im Datenfenster darstellen zu können, wird EDIT LONG ON eingegeben.

```
DIESES IST DIE ERSTE ZEILE.....................................................
DIESES IST DIE ZWEITE ZEILE....................................................
ZEILE 3 WIRD MIT @CREATE ANGELEGT UND WIRD LAENGER.............................
DIESES IST DIE ERSTE ZEILE VERKETTET MIT DIESES IST DIE ZWEITE ZEILEDIESES IST D
IE ERSTE ZEILE.................................................................









create 4:1:1-15:,'vierte',2:22-27: ; index on ...................0001.00:001(0)
```

Die Zeile 4 wird neu angelegt. Der neue Inhalt dieser Zeile ist die Verkettung der Spalten 1 bis 15 von Zeile 1, des Wortes VIERTE sowie der Spalten 22 bis 27 von Zeile 2 in dieser Reihenfolge.

Anschließend wird das Arbeitsfenster im Standardformat wieder eingeschaltet.

```
   1.00 DIESES IST DIE ERSTE ZEILE.....................................
   2.00 DIESES IST DIE ZWEITE ZEILE....................................
   3.00 ZEILE 3 WIRD MIT @CREATE ANGELEGT UND WIRD LAENGER..............
   4.00 DIESES IST DIE VIERTE ZEILE....................................
   5.00 ...............................................................
```

@CREATE (Format 2) Einlesen von Zeichenfolgen

Dieses Format von @CREATE übernimmt vom Bildschirm eine Zeichenfolge in eine Zeile oder Zeichenfolgevariable.

Diese Anweisung ist nur in EDT-Prozeduren (@DO- und @INPUT-Prozeduren) sinnvoll.

Operation	Operanden	L-Modus / @PROC
@**CR**EATE	line **R**EAD [string [,...]]	

line — Zeilennummer, die vom Bildschirm aus mit einer Zeichenfolge versehen werden soll. Es kann auch eine Zeichenfolgevariable oder eine Zeilennummervariable angegeben werden.

string[,...] — Zeichenfolge, die am Bildschirm als Eingabeaufforderung ausgegeben werden soll.
string kann angegeben werden:
- explizite Angabe in Hochkomma,
- implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Bei Ausführung von @CREATE wird der Benutzer über den Bildschirm aufgefordert, eine Zeichenfolge einzugeben. Wurde string angegeben, erscheint er als Eingabeaufforderung, sonst erscheint ein Stern. Die vom Benutzer eingegebene Zeichenfolge wird in die angegebene Zeile oder Zeichenfolgevariable geschrieben.

Tabulatorzeichen werden nicht verarbeitet.

- Wird string angegeben, dann wird über @CREATE, Format 2 der WRTRD-Makro des Ablaufteils angesprochen.
- Wird string nicht angegeben oder @CREATE READ aus einem Stapelprozeß heraus gegeben, wird der RDATA-Makro verwendet.

Beispiel 1

```
6.     @PRINT
1.0000 HALLO
2.0000 KEINER VERLAESST
3.0000 DEN RAUM
4.0000 ZEILE
5.0000 SOLL DENN AUSGEGEBEN
6.     @SET #S1 = ' WERDEN *** '
6.     @PROC 1
```

```
   1.     @ @CREATE #S2 READ '*** WELCHE ',4,5,#S1 ---------------------- (01)
   2.     @ @SET #L2 = SUBSTR #S2 ---------------------------------------- (02)
   3.     @ @PRINT #L2
   4.     @END
   6.     @DO 1
*** WELCHE ZEILE SOLL DENN AUSGEGEBEN WERDEN *** 2 ---------------------- (03)
   2.0000 KEINER VERLAESST
   6.
```

(01) Über @CREATE-READ soll die Zeichenfolgevariable #S2 erzeugt werden. Zuvor wird aber der Text auf dem Bildschirm ausgegeben, der sich aus *** WELCHE und den Inhalten der Zeilen 4 und 5 sowie der Zeichenfolgevariablen #S1 ergibt.

(02) Die Zeilennummervariable #L2 wird erzeugt aus dem Inhalt der Zeichenfolgevariable #S2.

(03) Die über den Bildschirm erschienene Anfrage wird beantwortet.

Beispiel 2

```
   1.     @ @CREATE #S0 READ '*** WELCHE PROZEDURDATEI SOLL ------------ (01)
DIE UHRZEIT AUSGEBEN ? *** '
   2.     @ @SET #I0 = SUBSTR #S0
   3.     @ @PROC #I0
   4.     @ @@SET #S1 = TIME
   5.     @ @@CREATE #S2: '*** ES IST JETZT ',#S1,' ***' ---------------- (02)
   6.     @ @@PRINT #S2 N
   7.     @ @END
   8.     @ @DO #I0
   9.     @SAVE 'TEST.CREATE-READ'
   9.     @INPUT 'TEST.CREATE-READ'
*** WELCHE PROZEDURDATEI SOLL DIE UHRZEIT AUSGEBEN ? *** 1 ------------- (03)
*** ES IST JETZT 110939 ***
   9.
```

(01) Der Text @CREATE #S0.... wird eingegeben. Dieser Text wird erst unter (03) als Anweisung interpretiert.

(02) Hier wird die Prozedurzeile erstellt, die später für das Festhalten der Uhrzeit in #S1 verantwortlich ist.

(03) Über @INPUT werden die in TEST.CREATE-READ festgehaltenen Anweisungen ausgeführt. Also wird hiermit @CREATE..READ außerhalb von Arbeitsdateien benutzt.

@DELETE Löschen von Dateien, Bibliothekselementen und Satzmarkierungen

@DELETE bietet in drei Formaten folgende Möglichkeiten zum Löschen:

– Löschen einer Arbeitsdatei oder von Teilen einer Arbeitsdatei (Format 1),
– Löschen eines Programm-Bibliothekselementes oder einer POSIX-Datei (Format 2),
– Löschen von Satzmarkierungen (Format 3).

Wurde eine ISAM-Datei mit @OPEN real eröffnet, kann diese Datei teilweise oder vollständig auf Platte gelöscht werden (Format 1). Der Katalogeintrag bleibt erhalten.

@DELETE (Format 1) Löschen von Arbeitsdateien

Mit diesem Format können in einer Arbeitsdatei gelöscht werden:

– die vollständige Arbeitsdatei
– einzelne Zeilen und/oder Zeilenbereiche einer Arbeitsdatei
– Spaltenbereiche

Operation	Operanden	F-Modus / L-Modus
@**D**ELETE	[rng [:domain]] [,...]	

In der F-Modus-Anweisungszeile wird nur D alleine mit einer Fehlermeldung abgewiesen, um ein unbeabsichtigtes Löschen der gesamten Arbeitsdatei bei der Eingabe von D-Kurzanweisungen zu verhindern.

rng Zeilenbereich, bestehend aus:
– einer einzelnen Zeile (z.B. 6) oder aus
– mehreren aufeinanderfolgenden Zeilen (z.B. 8-20).

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

Wird rng nicht angegeben, wird die gesamte Arbeitsdatei gelöscht.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte der Rest der Zeile gelöscht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile gelöscht.

Ist die Arbeitsdatei nicht leer, wird durch @DELETE auch der lokale Eintrag für den Dateinamen gelöscht (siehe @FILE, @GET, @READ).

Wird eine Arbeitsdatei vollständig gelöscht, werden sowohl die aktuelle Zeilennummer als auch die Schrittweite auf 1 gesetzt und die Einträge im 3stufigen Keller des EDT gelöscht. Eine eventuell angelegte Sicherungsdatei wird ebenefalls gelöscht.

Beispiel

```
   1.00 111.....................................................................
   1.50 111.....................................................................
   1.60 111.....................................................................
   1.70 111.....................................................................
   1.80 111.....................................................................
   1.90 111.....................................................................
   2.00 111.....................................................................
   2.10 111.....................................................................
   3.00       333...............................................................
   4.00          444............................................................
   4.10       333...............................................................
   4.20          444............................................................
   4.30       333...............................................................
   4.40          444............................................................
   4.50       333...............................................................
   4.60          444............................................................
   4.70       333...............................................................
   4.80          444............................................................
   4.90       333...............................................................
   5.00          444............................................................
   6.00 123456789012............................................................
   7.00 .....................................................................
   8.00 .....................................................................
delete 1-2 ..................................................0001.00:001(0)
```

Der Zeilenbereich von Zeilennummer 1 bis 2 wird in der Arbeitsdatei gelöscht.

```
   2.10 111.....................................................................
   3.00       333...............................................................
   4.00          444............................................................
   4.10       333...............................................................
   4.20          444............................................................
   4.30       333...............................................................
   4.40          444............................................................
   4.50       333...............................................................
   4.60          444............................................................
   4.70       333...............................................................
   4.80          444............................................................
   4.90       333...............................................................
   5.00          444............................................................
   6.00 123456789012............................................................
   7.00 ........................................................................
   8.00 ........................................................................
   9.00 ........................................................................
  10.00 ........................................................................
  11.00 ........................................................................
  12.00 ........................................................................
  13.00 ........................................................................
  14.00 ........................................................................
  15.00 ........................................................................
delete & : 7-10.....................................................0002.10:001(0)
```

In der gesamten Arbeitsdatei sollen die Spalten 7 bis 10 einschließlich gelöscht werden.

```
   2.10 111.....................................................................
   3.00       ..................................................................
   4.00       44................................................................
   4.10       ..................................................................
   4.20       44................................................................
   4.30       ..................................................................
   4.40       44................................................................
   4.50       ..................................................................
   4.60       44................................................................
   4.70       ..................................................................
   4.80       44................................................................
   4.90       ..................................................................
   5.00       44................................................................
   6.00 12345612................................................................
   7.00 ........................................................................
.....................................................................0002.10:001(0)
```

@DELETE (Format 2) Löschen von Bibliothekselementen

Löschen eines Elementes aus einer Programm-Bibliothek oder einer Datei.

Operation	Operanden	F-Modus / L-Modus
@**D**ELETE	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) / **F**ILE=path2 }	

path1 Name der Bibliothek. path1 kann auch über eine Zeichenfolgevariable angegeben werden.

elemname Name des Elements. elemname kann auch über eine Zeichenfolgevariable angegeben werden.

vers Versionsbezeichnung des gewünschten Elements (siehe Handbuch „LMS“ [13]). Wird vers nicht angegeben, oder *STD, wird die höchste vorhandene Version des Elementes gewählt.

elemtyp Typ des Elements. elemtyp kann auch über Zeichenfolgevariable angegeben werden.
Zulässige Typangaben: S, M, P, J, D, X, R, C, H, L, U, F, *STD und freie Typnamen mit entsprechendem Basistyp. Falls nicht angegeben, wird der in @PAR ELEMENT-TYPE voreingestellte Wert verwendet.

Wird ein freier Typnamen verwendet, so liegt es in der Verantwortung des Benutzers, daß der zugehörige Basistyp einem zulässigen Typ S, M, P, J, D, X, R, C, H, L, U oder F entspricht.

Typ	**Elementinhalt**
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats
R	Bindemodule
C	Lademodule
H	von H-Assembler erzeugt
L	vom BINDER erzeugt
U	von IFG erzeugt
F	von IFG erzeugt

*STD Voreinstellung
Typ S ist die Voreinstellung nach Aufruf des EDT. Mit @PAR ELEMENT-TYPE kann eine andere zulässige Typangabe als Voreinstellung festgelegt werden.

path2 Name der BS2000-Datei (vollqualifizierter Dateiname), die gelöscht werden soll. path2 kann auch über eine Zeichenfolgevariable angegeben werden.

Beispiel

```
DELETE LIBRARY = PROGLIB (ELEMENT = TESTALT (2))
```

Die zweite Version des Bibliothekselementes TESTALT der Bibliothek PROGLIB mit dem Elementtyp S wird gelöscht.

@DELETE (Format 3) Löschen von Satzmarkierungen

Mit diesem Format werden Satzmarkierungen (siehe Abschnitt „Beschreibung der Satzmarkierungen des F-Modus“ auf Seite 123) gelöscht.

Operation	Operanden	F-Modus / L-Modus
@**D**ELETE	**MA**RK [m, [...]]	

MARK Satzmarkierungen in der aktuellen Arbeitsdatei werden gelöscht.

m Nummer der Satzmarkierungen, die gelöscht werden sollen.

$1 \leq m \leq 9$.

Die Satzmarkierungen können auch über Ganzzahlvariablen angegeben werden.
Wird m nicht angegeben, werden alle Satzmarkierungen 1 bis 9 der aktuellen Arbeitsdatei gelöscht.

Satzmarkierungen mit Sonderfunktion (Markierung 13, 14, 15) werden nicht gelöscht (z.B. Satzmarkierung 15 für Schreibschutz, die bei EDT als Unterprogramm gesetzt werden kann).

@DELIMIT Textbegrenzerzeichen definieren

Diese Anweisung erlaubt dem Benutzer die Definition einer Menge von Zeichen, die beim Suchen einer Zeichenfolge mit @ON als Textbegrenzer fungieren (siehe @ON).

Operation	Operanden	F-Modus / L-Modus
@**D**ELIMIT	= { **R** | str1 | **+**|**–** str2 }	

Das „=“ Zeichen muß in jedem Fall angegeben werden, da sonst D als @DELETE interpretiert wird.

R — Setzt die Textbegrenzermenge auf die vom EDT gesetzte Standardmenge zurück. Diese besteht aus dem Leerzeichen (X’40’) sowie den Zeichen +.!*();–/,?:’=" .

str1 — Künftige Gesamtmenge der Textbegrenzer.

str2 — Menge von Textbegrenzerzeichen, um die die bestehende Textbegrenzermenge erweitert (+) bzw. vermindert (–) wird.

Wird kein Operand angegeben, enthält die Textbegrenzermenge kein einziges Zeichen. Dies bedeutet, daß das nachfolgende @ON mit Textbegrenzersuche nur dann Treffer finden kann, wenn der Satz genau aus der gesuchten Zeichenfolge besteht.

@DIALOG Umschalten in den F-Modus Bildschirmdialog

Bei RDATA-Eingabe (BS2000-Prozeduren) und bei Aufruf des EDT als Unterprogramm (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]; CMD-Funktion) schaltet @DIALOG in den F-Modus-Bildschirmdialog um.

Mit @END, @HALT, @RETURN oder K1 wird der Bildschirmdialog beendet und die mit @DIALOG unterbrochene Abarbeitung fortgesetzt.

Operation	Operanden	F-Modus / L-Modus
@DIALOG		

@DIALOG wird

– im F-Modus und im Stapelbetrieb ignoriert,
– im L-Modus in EDT-Prozeduren (@DO) oder in einer INPUT-Datei (@INPUT) oder bei WRTRD-Eingabe mit einer Fehlermeldung abgewiesen.

Wurde der Bildschirmdialog aus einer BS2000-Prozedur aufgerufen, sind @SYSTEM ohne Operanden und @EDIT ONLY gesperrt.

Ein Wechsel in das Betriebssystem ist nur über K2 möglich.

Nach Beendigung des Bildschirmdialogs sollten alle benötigten Aktualwerte (Arbeitsdatei, Bibliothek) neu eingestellt werden, da sie vom Benutzer im F-Modus-Dialog verändert werden können.

Beispiel

BS2000-Prozedur PROC.DIALOG

```
/BEGIN-PROCEDURE LOGGING=A,PARAMETERS=YES(-
/   PROCEDURE-PARAMETERS=(&FILE1=,&FILE2=),-
/   ESCAPE-CHARACTER='&')
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/MODIFY-JOB-SWITCHES ON=5------------------------------------------------ (01)
/START-PROGRAM $EDT
@PROC 1 ------------------------------------------------------------------ (02)
@READ '&FILE1' ----------------------------------------------------------- (03)
@PAR LOWER=ON,SCALE=ON --------------------------------------------------- (04)
@DIALOG ------------------------------------------------------------------ (05)
@PROC 1 ------------------------------------------------------------------ (06)
@WRITE '&FILE2' ---------------------------------------------------------- (07)
@HALT -------------------------------------------------------------------- (08)
```

```
/MODIFY-JOB-SWITCHES OFF=5
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/END-PROCEDURE
```

(01) Der Auftragsschalter 5 wird vor dem Laden des EDT gesetzt. Damit wird der L-Modus eingestellt. Der EDT liest die Eingaben mit RDATA von SYSDTA.

(02) Es wird in die Arbeitsdatei 1 umgeschaltet.

(03) Eine Datei soll eingelesen werden. Der Dateiname wird beim Ablauf der Prozedur angefordert.

(04) Kleinbuchstabenverarbeitung und Spaltenzähleranzeige werden eingeschaltet.

(05) Der EDT schaltet in den F-Modus Bildschirmdialog um. Das Arbeitsfenster wird am Bildschirm ausgegeben. Im Dialog können alle Anweisungen des F- und L-Modus eingegeben werden. Mit @END, @HALT oder @RETURN bzw. mit K1 wird der F-Modus-Dialog beendet und die mit @DIALOG unterbrochene Abarbeitung fortgesetzt.

(06) Die Arbeitsdatei 1 wird als aktuelle Arbeitsdatei erneut eingestellt. Dies ist notwendig, da im F-Modus Bildschirmdialog der Benutzer eine andere Arbeitsdatei eingestellt haben könnte.

(07) Die Arbeitsdatei 1 wird in eine SAM-Datei zurückgeschrieben. Der Dateiname wird beim Ablauf der Prozedur angefordert.

(08) Der EDT wird beendet.

```
/call-procedure name=proc.dialog
%/PROCEDURE-A,(&FILE1=,&FILE2=),SUBDTA=&
%/ASSIGN-SYSDTA TO-FILE=*SYSCMD
%/MODIFY-JOB-SWITCHES ON=5
%/START-PROGRAM $EDT
%  BLS0500 PROGRAM 'EDT', VERSION '16.5A' OF 'yy-mm-dd' LOADED.
PROGRAM  EDT/16.5A00  STARTED
%PROC 1
%@READ '&FILE1'
%&FILE1=bsp.dialog
```

Die Prozedur PROC.DIALOG wird gestartet. Dabei wird der Dateiname der einzulesenden Datei angefordert. Anschließend schaltet der EDT in den F-Modus Bildschirmdialog um.

```
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 In BS2000-Prozeduren (Systemprozeduren) bei RDATA-Eingabe und..........
   2.00 beim Aufruf des EDT als Unterprogramm (siehe CMD-Funktion).............
   3.00 schaltet @DIALOG in den Bildschirmdialog um. ..........................
   4.00 Das Arbeitsfenster wird am Bildschirm ausgegeben. Im Dialog............
   5.00 koennen alle Anweisungen des F- und L-Modus eingegeben werden. .........
   6.00 ......................................................................




halt.....................................................0001.00:001(1)
```

Entsprechend den unter (04) definierten Voreinstellwerten werden Kleinbuchstaben und die Spaltenzähleranzeige ausgegeben. Mit @HALT wird der F-Modus-Dialog wieder beendet und die mit mit @DIALOG unterbrochene Prozedur fortgesetzt.

Im weiteren Verlauf der Prozedur wird der Dateiname angefordert, in den die Arbeitsdatei geschrieben werden soll. Abhängig von den Aktionen im F-Modus-Dialog werden ggf. noch weitere Meldungen ausgegeben.

```
%@WRITE 'bsp.dialog'
%@HALT
%&FILE1=bsp.dialog1
% EDT8000 EDT NORMAL END
%/MODIFY-JOB-SWITCHES OFF=5
%/ASSIGN-SYSDTA TO-FILE=*PRIMARY
%/END-PROCEDURE
/
```

@DO Starten von EDT-Prozeduren

Mit @DO kann man

– EDT-Prozeduren starten, d.h. den Inhalt einer Arbeitsdatei (1-22) zeilenweise abarbeiten lassen (Format 1). Die Zeilen können Texte oder EDT-Anweisungen enthalten.

– steuern, welche Zeilen einer Prozedur vor ihrer Abarbeitung auf dem Bildschirm ausgegeben werden sollen (Format 2).

@DO (Format 1) Starten von EDT-Prozeduren

Mit @DO, Format 1 wird eine EDT-Prozedur gestartet, d.h., daß die in der angegebenen Arbeitsdatei 1-22 stehenden Textzeilen und EDT-Anweisungen abgearbeitet werden.

Operation	Operanden	F-Modus / L-Modus / @PROC
@DO	procnr [,] [(param [,...])] [spec] [=ln1,ln2 [, [-] ln3]] [**P**RINT]	

procnr Nummer der Arbeitsdatei, deren Inhalt der EDT als Eingabe verwenden soll. Für procnr ist entweder eine Ganzzahl ($1 \leq procnr \leq 22$) oder eine Ganzzahlvariable anzugeben. Bei Angabe einer Ganzzahlvariablen muß in dieser die Nummer der Arbeitsdatei stehen.

param Parameter, die an die auszuführende Prozedur übergeben werden. Die Parameter müssen in der Prozedur mit @PARAMS definiert sein (siehe @PARAMS). Sie werden voneinander durch Komma getrennt. Ist @PAR LOWER=ON eingeschaltet, können auch Kleinbuchstaben übergeben werden (kein Umsetzen in Großbuchstaben).

Die Stellungsparameter müssen vor den Schlüsselwortparametern stehen und genau in der Reihenfolge angegeben werden, in der sie in @PARAMS definiert wurden. Schlüsselwortparameter können in beliebiger Reihenfolge angegeben werden.

Die Anzahl der Parameter ist durch die maximale Länge einer EDT-Anweisung von 256 Zeichen begrenzt.

spec Schleifensymbol. In der Prozedur kann es als Operand in EDT-Anweisungen verwendet werden, wenn eine Zeilennummer angesprochen werden soll. Bei der Ausführung der Prozedur nimmt der EDT den jeweils aktuellen Wert des Schleifensymbols.

Das Schleifensymbol muß ein Sonderzeichen sein, sonst wird @DO mit der Fehlermeldung abgewiesen: % EDT3952 INVALID SYMBOL

Um Fehler und unvorhersehbare Ergebnisse zu vermeiden, dürfen folgende Zeichen nicht als Schleifensymbol gewählt werden:

% $? * (: # + – . < = > ' ;

Wird die Prozedur im F-Modus gestartet, darf ; nicht verwendet werden.

Geeignete Zeichen für das Schleifensymbol sind:

! " { } [ ] | /

Wird das Schleifensymbol nicht angegeben, gilt es als undefiniert. Wird die Operandenfolge ln1,ln2,[–]ln3 nicht angegeben, hat das Schleifensymbol den Wert 1.

=ln1,ln2,[–]ln3

Eine Prozedur wird mehrmals durchlaufen (siehe Beispiel 3).

Vor dem ersten Durchlaufen weist der EDT ln1 dem Schleifensymbol als Anfangswert zu. Nach jedem Durchlauf erhöht oder vermindert (Minuszeichen vor ln3) der EDT den Wert des Schleifensymbols um ln3. Standardwert für ln3 ist 1. Solange das Schleifensymbol den Wert von ln2 noch nicht überschritten bzw. unterschritten hat, wird die Prozedur erneut durchlaufen. Andernfalls wird das Durchlaufen der Prozedur abgebrochen.

Die Prozedur wird mindestens einmal durchlaufen, da die Prüfung jeweils nach dem Durchlauf erfolgt (REPEAT UNTIL).

Für ln1, ln2 oder ln3 können auch Zeilennummernsymbole (z.B. %,$) angegeben werden. Der EDT nimmt den Wert, den dieses Symbol bei der Ausführung von @DO hat. Ändert sich der Wert dieses Symbols während der Ausführung der Prozedur, bleibt die Zahl der Durchläufe davon unberührt.

Der Standardwert für ln1,ln2 und ln3 ist 1.

PRINT

Jede Zeile der Prozedur wird vor ihrer Verarbeitung ausgegeben.
Durch die Angabe von PRINT wird erreicht, daß alle Fehlermeldungen ausgegeben werden und der EDT-Fehlerschalter gesetzt wird. Normalerweise werden Fehlermeldungen, die die Durchführung der Prozedur nicht beeinflussen (z.B. die Meldungen % EDT0901 NO MATCH IN RANGE oder % EDT4932 LINE NUMBER NOT FOUND) nicht ausgegeben und der EDT-Fehlermeldungsschalter nicht gesetzt.

Regeln bei der Angabe von EDT-Parametern

1. Der Wert des Parameters ergibt sich aus allen angegebenen Zeichen zwischen den Kommas, einschließlich der Leerzeichen.
2. Enthält ein Parameterwert Hochkommas oder schließende Klammern, muß er in Hochkommas eingeschlossen werden. Hochkommas im Parameterwert müssen dann doppelt angegeben werden. Wird zuvor mit @QUOTE einem anderen Zeichen die Funktion des Hochkommas zugeordnet, gilt dies nicht für die den Parameterwert einschließenden Hochkommas.
3. Wird für einen Parameter kein Wert angegeben, erhält der Parameter den Wert leere Zeichenfolge (Leerstring).

   Kein Wert angegeben heißt bei

   – Stellungsparametern, daß zwischen den Begrenzungszeichen der Parameterwerte (Klammer und Komma) kein Wert angegeben ist.

   – Schlüsselwortparametern, daß dem Gleichheitszeichen ein Komma oder eine Klammer folgt.

Abbruch von EDT-Prozeduren

EDT-Prozeduren können durch K2 jederzeit unterbrochen werden. Vom Betriebssystem aus kann man

– mit RESUME-PROGRAM die Prozedur fortsetzen oder

– mit SEND-MESSAGE TO=PROGRAM in den EDT zurückkehren und die Prozedur abbrechen.

Bei der Abarbeitung einer @RUN-Anweisung oder einer Benutzeranweisung (siehe @USE) kann die Prozedur nicht mit SEND-MESSAGE TO=PROGRAM abgebrochen werden.

Eine fehlerhafte Anweisung führt nicht zum Abbruch.

Beispiel 1

```
1.     @SET #S0 = 'TEST VON PROZEDURDATEI 1'
1.     @PROC 1 ------------------------------------------------------ (01)
1.     @ @SET #S1 = #S0:1-4: ---------------------------------------- (02)
2.     @ @CREATE #S2: ' '*4,#S0:5-9:
3.     @ @CREATE #S3: ' '*9,#S0:10-24:
4.     @ @CREATE #S4: ' '*24
5.     @ @PRINT #S1.-#S4
6.     @ @PRINT #S0
7.     @END --------------------------------------------------------- (03)
1.     @DO 1 -------------------------------------------------------- (04)
  #S01 TEST
  #S02      VON
  #S03          PROZEDURDATEI 1
  #S04
  #S00 TEST VON PROZEDURDATEI 1
1.
```

(01) Es wird in die Arbeitsdatei 1 umgeschaltet.

(02) In die Arbeitsdatei 1 werden EDT-Anweisungen geschrieben. Diese bewirken beim Aufruf der Prozedur mit @DO, daß die Zeichenfolgevariablen #S1 bis #S4 erzeugt und zusammen mit #S0 ausgegeben werden.

(03) Mit @END wird aus der Arbeitsdatei 1 zurückgekehrt.

(04) Die in Arbeitsdatei 1 stehende Prozedur wird aufgerufen.

Beispiel 2

```
1.     @PROC 2 ------------------------------------------------------ (01)
1.     @ @PARAMS &STRING -------------------------------------------- (02)
2.     @ @SET #S1 = '++++++++++'
3.     @ @SET #S2 = &STRING ----------------------------------------- (03)
4.     @ @PRINT #S2
5.     @END
1.     @DO 2(#S1) PRINT --------------------------------------------- (04)
1.     @SET #S1 = '++++++++++'
1.     @SET #S2 = #S1
1.     @PRINT #S2
  #SO2 ++++++++++
1.     @DO 2('#S1') PRINT ------------------------------------------- (05)
1.     @SET #S1 = '++++++++++'
1.     @SET #S2 = #S1
1.     @PRINT #S2
  #SO2 ++++++++++
1.     @DO 2( '#S1' ) PRINT ----------------------------------------- (06)
1.     @SET #S1 = '++++++++++'
1.     @SET #S2 = '#S1'
1.     @PRINT #S2
  #SO2 #S1
1.
```

(01) Es wird in die Arbeitsdatei 2 umgeschaltet.

(02) Die erste in dieser Arbeitsdatei abgelegte Zeile ist eine @PARAMS-Anweisung. Damit kann innerhalb dieser Arbeitsdatei der Stellungsparameter &STRING öfter Verwendung finden.

(03) #S2 soll ein Wert zugewiesen werden, der jedoch zum Zeitpunkt der Definition der Arbeitsdatei 2 nicht feststeht und erst in einem @DO 2(...)... angegeben wird.

(04) Durch den in Klammern stehenden Wert #S1 wird vor Ausführung der in der Arbeitsdatei 2 stehenden Anweisungen überall für &STRING der Wert #S1 eingesetzt. PRINT bewirkt das Ausgeben der Anweisungen vor ihrer Durchführung.

(05) Nun wird der Wert #S1 für den Stellungsparameter &STRING übergeben. Da das erste und letzte Zeichen dieses Parameterwertes ein Hochkomma ist, werden diese beim Ersetzen des Parameterwertes in Arbeitsdatei 2 unterdrückt, was auch hier der PRINT-Operand deutlich zeigt. Somit führt dies zum selben Effekt wie (04).

(06) Der einzige Unterschied zu (05) ist, daß der Parameterwert um ein vorangehendes bzw. nachfolgendes Leerzeichen erweitert wurde. Dies genügt aber, um das Hochkomma als Inhalt des Parameterwertes zu übergeben.

Beispiel 3

```
 1.     *
 2.     @PROC 3 ------------------------------------------------------ (01)
 1.     @ @CREATE $+1: $,'*' ----------------------------------------- (02)
 2.     @END ---------------------------------------------------------- (03)
 2.     @DO 3,!=1,15 ------------------------------------------------- (04)
 2.     @PRINT
 1.0000 *
 2.0000 **
 3.0000 ***
 4.0000 ****
 5.0000 *****
 6.0000 ******
 7.0000 *******
 8.0000 ********
 9.0000 *********
10.0000 **********
11.0000 ***********
12.0000 ************
13.0000 *************
14.0000 **************
15.0000 ***************
16.0000 **************** ------------------------------------------- (05)
 2.
```

(01) Es wird in die Arbeitsdatei 3 umgeschaltet.

(02) Eine einzige EDT-Anweisung wird in die Arbeitsdatei 3 geschrieben.

(03) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(04) Die Arbeitsdatei 3 wird ausgeführt. Hierbei wird als Schleifensymbol das Zeichen ! verwendet. Die Arbeitsdatei 3 wird 15mal durchlaufen. Man könnte dort Zeilennummern über ! ansprechen, kann dies aber auch unterlassen wie in diesem Beispiel. Mit !=1,15 erreicht man dasselbe wie durch 15maliges Abschicken von @DO 3 ohne diese Operandenfolge.

(05) Beim Ausgeben erkennt man, daß 15 neue Zeilen angelegt wurden.

Beispiel 4

```
   5.     @PRINT
   1.0000 1111111
   2.0000 2222222
   3.0000 3333333
   4.0000 4444444
   5.     @SET #S4 = '-----------------'
   5.     @PROC 4 ------------------------------------------------------ (01)
   1.     @ @PRINT !.-.$ ----------------------------------------------- (02)
   2.     @ @PRINT #S4 N
   3.     @END
   5.     @DO 4,!=$,%,-1 ----------------------------------------------- (03)
   4.0000 4444444
-----------------
   3.0000 3333333
   4.0000 4444444
-----------------
   2.0000 2222222
   3.0000 3333333
   4.0000 4444444
-----------------
   1.0000 1111111
   2.0000 2222222
   3.0000 3333333
   4.0000 4444444
-----------------
   5.
```

(01) Es wird in die Arbeitsdatei 4 umgeschaltet.

(02) Eine Zeilennummer wird über das Schleifensymbol ! angesprochen.

(03) Die Arbeitsdatei 4 wird mehrmals angestoßen. Beim 1. Durchlauf wird für ! der Wert der höchsten vergebenen Zeilennummer angenommen. Bei jedem weiteren Durchlauf erniedrigt sich dieser Wert um 1 (drittes ln= -1), bis das Schleifensymbol den Wert der niedrigsten vergebenen Zeilennummer (%) angenommen hat.

Beispiel 5

```
   1.     @PROC 4 ------------------------------------------------------ (01)
   1.     @READ 'PROC-DATEI.4' ----------------------------------------- (02)
   6.     @PRINT
   1.0000 @PARAMS &A
   2.0000 ABC
   3.0000 EFG
   4.0000 @ON  1 CHANGE 'ABC' TO '&A'
   5.0000 @5: &A
   6.     @END --------------------------------------------------------- (03)
   1.     @DO 4 ('A,'B') ----------------------------------------------- (04)
INVALID VALUE
   1.     @DO 4 ('A,''B')
   6.     @PRINT
   1.0000 A,'B -------------------------------------------------------- (05)
   2.0000 EFG
   5.0000  A,''B
   6.
```

(01) Es wird in die Arbeitsdatei 4 umgeschaltet.

(02) Die SAM-Datei PROC-DATEI.4 wird in die Arbeitsdatei 4 gelesen.

(03) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(04) Die Angabe eines einzelnen Hochkommas in einem Parameterwert weist der EDT zurück.

(05) Beim Ausführen der Arbeitsdatei wurden Zeilen in die Hauptdatei geschrieben. In Zeile 1 hat der EDT eines der 2 aufeinanderfolgenden Hochkommas unterdrückt. In Zeile 3 hat der EDT den Parameterwert unverändert übernommen.

@DO (Format 2) Protokollierung aus- oder einschalten

Mit diesem Format kann PRINT von @DO, Format 1 an einer beliebigen Stelle innerhalb der Prozedur zurückgenommen oder auch gesetzt werden.

Operation	Operanden	@PROC
@DO	**N** | **P**	

N Der EDT protokolliert die folgenden Zeilen der Prozedur nicht mehr vor ihrer Verarbeitung.

P Der EDT protokolliert die folgenden Zeilen der Prozedur vor ihrer Verarbeitung.

Diese Anweisung wird hauptsächlich zur Fehlersuche in EDT-Prozeduren verwendet. Man kann z.B. feststellen, ob eine bestimmte Stelle einer Prozedur durchlaufen wird oder nicht.

Beispiel

```
1.     @PROC 5 ------------------------------------------------------ (01)
1.     @ @SET #S5 = 'A'
2.     @ @DO N ------------------------------------------------------ (02)
3.     @ @CREATE #S6: 'B'*6,#S5
4.     @ @CREATE #S7: #S6,'C',#S6
5.     @ @DO P
6.     @ @PRINT #S5.-#S7 -------------------------------------------- (03)
7.     @ @DELETE #S5.-#S7
8.     @END --------------------------------------------------------- (04)
1.     @DO 5 PRINT -------------------------------------------------- (05)
1.     @SET #S5 = 'A'
1.     @DO N
1.     @PRINT #S5.-#S7
  #S05 A
  #S06 BBBBBBA
  #S07 BBBBBBACBBBBBBA
1.     @DELETE #S5.-#S7
1.
```

(01) Es wird in die Arbeitsdatei 5 umgeschaltet.

(02) Die folgenden Zeilen der Prozedur werden nicht mehr protokolliert.

(03) Der EDT protokolliert die folgenden Zeilen der Prozedur vor der Verarbeitung.

(04) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(05) Die Prozedur in Arbeitsdatei 5 wird gestartet. Die Anweisungen sollen vor der Durchführung protokolliert werden.

@DROP Löschen von Arbeitsdateien

@DROP löscht die Arbeitsdateien 1-22 und gibt die dafür benötigten virtuellen Speicherseiten frei.

@DROP darf nur in der Arbeitsdatei 0 eingegeben werden, also nicht in EDT-Prozeduren.

Operation	Operanden	F-Modus / L-Modus
@**DR**OP	{ procnr [,...] / **A**LL }	

procnr Nummer einer Arbeitsdatei (1-22), die gelöscht werden soll. Es können beliebig viele Arbeitsdateien angegeben werden.

ALL Die Arbeitsdateien 1-22 werden gelöscht und die dafür benötigten virtuellen Speicherseiten freigegeben.

i

– @DROP entfernt die lokalen Dateinamen.

– Eröffnete Dateien Bibliothekselemente sollten vorher geschlossen werden (siehe @CLOSE).

Beispiel 1

```
   1.     @PROC USED ------------------------------------------------- (01)
<03>     1.0000 TO    3.0000
<05>     1.0000 TO    1.0000
<08>     1.0000 TO    1.0000
<10>     1.0000 TO    1.0000
<14>     1.0000 TO    1.0000
   1.     @DROP 10 -------------------------------------------------- (02)
   1.     @PROC USED
<03>     1.0000 TO    3.0000
<05>     1.0000 TO    1.0000   ---------------------------------------- (03)
<08>     1.0000 TO    1.0000
<14>     1.0000 TO    1.0000
   1.     @DROP 8,5 ------------------------------------------------- (04)
   1.     @PROC USED
<03>     1.0000 TO    3.0000   ---------------------------------------- (05)
<14>     1.0000 TO    1.0000
   1.
```

(01) Die belegten Arbeitsdateien 1-22 sollen ausgegeben werden. Es sind in diesem Fall die Arbeitsdateien 3, 5, 8, 10, 14.

(02) Die Arbeitsdatei 10 wird gelöscht und freigegeben.

(03) @PROC USED gibt aus, daß nur noch die Arbeitsdateien 3, 5, 8, 14 belegt sind.

(04) Mit @DROP können auch mehrere Arbeitsdateien gelöscht und freigegeben werden, wie hier z.B. 5 und 8.

(05) Jetzt bleiben nur noch die Arbeitsdateien 3 und 14 übrig.

Beispiel 2

```
   1.     @PROC USED ------------------------------------------------- (01)
<03>     1.0000 TO    3.0000
<14>     1.0000 TO    1.0000
   1.     @DROP ALL ------------------------------------------------- (02)
   1.     @PROC USED
% EDT0907 NO PROCEDURE FILES DECLARED ---------------------------------- (03)
   1.
```

(01) Alle belegten Arbeitsdateien sollen ausgegeben werden.

(02) Die Arbeitsdateien 1-22 werden gelöscht und freigegeben.

(03) Es ist keine der Arbeitsdateien 1 bis 22 mehr belegt.

@EDIT Umschalten des Arbeitsmodus

Mit dieser Anweisung kann man

- vom L-Modus in den F-Modus umschalten und umgekehrt (@EDIT FULL SCREEN, @EDIT ONLY)
- die Ausgabe der aktuellen Zeilennummer unterdrücken
- zwischen Lesen von SYSDTA mit RDATA und WRTRD umschalten (@EDIT ONLY, @EDIT),
- den Inhalt der aktuellen Zeile vor der Änderung anzeigen lassen (@EDIT .. PRINT),
- den SEQUENTIAL-Modus (siehe @+, @–) einschalten (@EDIT .. SEQUENTIAL),
- für Datenschreibstationen die Maximallänge einer Zeile (also den rechten Rand) definieren (@EDIT .. cl).

Operation	Operanden	F-Modus / L-Modus
@**E**DIT	{ **F**ULL SCREEN [**O**NLY] [**P**RINT] [**S**EQUENTIAL] [cl] }	

Werden keine Operanden angegeben wird in den L-Modus umgeschaltet.

FULL SCREEN
: bewirkt, daß der EDT in den F-Modus verzweigt.

  Im Stapelbetrieb und im F-Modus wird @EDIT FULL SCREEN ignoriert.

  Erfolgt diese Anweisung innerhalb einer EDT-Prozedur (@DO) oder einer INPUT-Datei (@INPUT), wird sie mit einer Fehlermeldung abgewiesen.

  Wird @EDIT FULL SCREEN im Dialog innerhalb eines Anweisungsblocks (@BLOCK-Modus) angegeben, werden die nachfolgenden Anweisungen ignoriert.

  Mit @DIALOG kann man ebenfalls in den F-Modus verzweigen (siehe @DIALOG).

ONLY
: Im F-Modus wird in den L-Modus umgeschaltet. Die Ausgabe der aktuellen Zeilennummer wird unterdrückt. Statt dieser wird * ausgegeben. Der EDT verwendet das Makro RDATA anstelle von WRTRD. Während WRTRD ausschließlich vom Bildschirm liest, liest RDATA von der Systemdatei SYSDTA.

  Wird @EDIT ohne ONLY angegeben, wird die aktuelle Zeilennummer wieder ausgegeben. Die Eingabe erfolgt wieder über WRTRD.

PRINT Die Angabe von PRINT bewirkt, daß vor der Ausgabe der Zeilennummer bzw. von * die Zeilennummer und der Zeileninhalt am Bildschirm ausgegeben werden.

SEQUENTIAL

Im Regelfall wird bei Eingabe eines Zeileninhalts oder @+ die aktuelle Zeilennummer um die Schrittweite erhöht (für @– gilt sinngemäß Entsprechendes).
Dadurch kann es vorkommen, daß - vom Benutzer unbemerkt - bereits existierende Zeilen übergangen werden, nämlich die, die zwischen der alten und der neuen aktuellen Zeilennummer liegen.

Wird SEQUENTIAL angegeben, wird die aktuelle Zeilennummer nur dann wie oben beschrieben gebildet, wenn es keine dazwischenliegende Zeile gibt. Im anderen Fall wird die erste dazwischenliegende Zeilennummer zur aktuellen Zeilennummer.

cl Maximallänge einer Zeile für eine Datenschreibstation. Der Wert von cl muß mindestens 50 betragen und darf nicht über 256 liegen.

Den Anfangswert von 72 entnimmt der EDT der Systemeinstellung durch das MODIFY-TERMINAL-OPTIONS-Kommando mit dem Operanden LINE-LENGTH.

i

PRINT kann besonders im Stapelbetrieb nützlich sein. Der Inhalt jeder bearbeiteten Zeile wird vor und nach der Bearbeitung protokolliert, wenn

– @EDIT PRINT und @LOG ALL gegeben werden,

– mittels einer Anweisung vor jeder Texteingabe auf die gewünschte Zeilennummer positioniert wird und

– Zeileninhalte nicht mittels einer Anweisung verändert werden, sondern ausschließlich durch die Eingabe der neuen Zeileninhalte,

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @EDIT ONLY beim Umschalten auf SYSDTA der Coded Character Set Name (CCSN) von SYSDTA geprüft. Das Umschalten auf SYSDTA mit @EDIT ONLY ist nur dann möglich, wenn der CCSN von SYSDTA mit dem aktuellen CCSN übereinstimmt. Falls nicht umgeschaltet werden kann, wird im Dialog nach einer Fehlermeldung auf das Lesen mit WRTRD (@EDIT ohne ONLY) umgestellt, im Stapelbetrieb wird der EDT abgebrochen.

@ELIM ISAM-Datei löschen

Diese Anweisung löscht eine auf Platte befindliche ISAM-Datei teilweise oder ganz. Wird die ganze Datei gelöscht, bleibt - im Gegensatz zu @UNSAVE - der Dateiname im Katalog bestehen. Außerdem ist es möglich, gleichzeitig in der virtuellen Datei und auf der Platte zu löschen.

Die Datei ist nur während der Ausführung von @ELIM geöffnet.

Operation	Operanden	F-Modus / L-Modus
@**EL**IM	['file'] [(ver)] range* [**B**OTH]	

file Dateiname.

Fehlt file, wird, falls vorhanden, der lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE).

ver Versionsnummer der Datei.

Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer. Bei Angabe einer Zahl wird @ELIM nur dann ausgeführt, wenn es sich um die aktuelle Versionsnummer handelt. Andernfalls wird lediglich die aktuelle Versionsnummer am Bildschirm angezeigt.

range* Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden. Zeichenfolgevariablen dürfen nicht angegeben werden.

Der Wert der symbolischen Zeilennummern bezieht sich nicht auf die mit file angegebene Datei, sondern auf die aktuelle Arbeitsdatei.

BOTH Der bezeichnete Zeilenbereich ist nicht nur in der ISAM-Datei, sondern auch in der virtuellen Datei zu löschen.

Die wichtigsten Unterschiede zu @UNSAVE sind:

– bei @ELIM bleibt der Katalogeintrag immer erhalten,
– mit @ELIM können nur ISAM-Dateien behandelt werden.

Beispiel

```
  23.00 ..........................................................................
get 'bsp.elim' noreseq .....................................0001.00:001(0)
```

Die ISAM-Datei BSP.ELIM wird in die Arbeitsdatei 0 eingelesen. Als Zeilennummern sollen die ISAM-Schlüssel übernommen werden.

```
   1.00 EINS.....................................................................
   2.00 ZWEI.....................................................................
   3.00 DREI.....................................................................
   4.00 VIER.....................................................................
   5.00 FUENF....................................................................
   6.00 SECHS....................................................................
   7.00 SIEBEN...................................................................
   8.00 ACHT.....................................................................




elim 'bsp.elim' 5-7 both ...................................0001.00:001(0)
```

Sowohl in der Arbeitsdatei 0 als auch in der ISAM-Datei BSP.ELIM wird der Zeilenbereich 5-7 gelöscht.

Falls die ISAM-Datei mit @GET ohne NORESEQ eingelesen wird, brauchen die zu löschenden Zeilenbereiche in der ISAM-Datei und in der Arbeitsdatei nicht übereinzustimmen.

```
   1.00 EINS.....................................................................
   2.00 ZWEI.....................................................................
   3.00 DREI.....................................................................
   4.00 VIER.....................................................................
   8.00 ACHT.....................................................................
   9.00 .........................................................................




delete ; get 'bsp.elim' noreseq.............................0001.00:001(0)
```

Die Arbeitsdatei 0 wird gelöscht und anschließend erneut die ISAM-Datei BSP.ELIM eingelesen.

```
1.00 EINS.....................................................................
2.00 ZWEI.....................................................................
3.00 DREI.....................................................................
4.00 VIER.....................................................................
8.00 ACHT.....................................................................
9.00 .......................................................................
```

Auch in der ISAM-Datei wurde der Zeilenbereich von 5 bis 7 gelöscht.

@END Bearbeitung der aktuellen Arbeitsdatei beenden

Mit @END wird die Bearbeitung der aktuellen Arbeitsdatei beendet. Es wird wieder in die Arbeitsdatei zurückgeschaltet, von der aus die Bearbeitung mit @PROC eingeleitet wurde.

Operation	Operanden	F-Modus / L-Modus / @PROC
@**EN**D	[comment]	

comment Beliebiger Kommentar.
comment darf nur im L-Modus angegeben werden.

Verhalten im L-Modus

Wird @END in einer Arbeitsdatei (ungleich Arbeitsdatei 0) eingegeben, so wird wieder in die Arbeitsdatei zurückgeschaltet, von der aus die Bearbeitung mit @PROC eingeleitet wurde.

Wird @END im Dialog in der Arbeitsdatei 0 eingegeben, wird nach der Meldung
% EDT4939 @END WITHOUT @PROC die Sicherungsabfrage % EDT0900 und
% EDT0904 ausgegeben, bzw. wenn keine zu sichernde Arbeitsdateien vorhanden sind, nur die Abfrage % EDT0904.

Die Abfrage EDT0904 wird durch Einschalten von Auftragsschalter 4 vor dem EDT-Lauf nicht unterdrückt.

Verhalten im F-Modus bzw. im Bildschirm-Dialog nach @DIALOG

Wird @END in einer Arbeitsdatei eingegeben, so erfolgt eine

– Beendigung des EDT-Laufs

– Rückkehr in eine Systemprozedur

– Fortführung eines Unterprogrammaufrufes.

Existieren noch ungesicherte Arbeitsdateien, so wird die Meldung
% EDT0900 EDITED FILE(S) NOT SAVED! und die Abfrage
% EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO) ausgegeben.

Y, die ungesicherten virtuellen Dateien gehen verloren. Der EDT wird beendet.

Die Sicherungsabfrage unterbleibt, wenn der F-Modus mit @DIALOG aufgerufen wurde.

Beispiel 1

```
   1.     @PROC 1 ------------------------------------------------------ (01)
   1.     @ @SET #S1 = DATE
   2.     @ @SET #S2 = TIME      ------------------------------------- (02)
   3.     @ @PRINT #S1.-#S2 N
   4.     @END ---------------------------------------------------------- (03)
```

(01) Es wird in die Arbeitsdatei 1 umgeschaltet.

(02) In die Arbeitsdatei 1 wird eine EDT-Prozedur eingegeben.

(03) Es wird in die Arbeitsdatei 0 zurückgekehrt. Die in Arbeitsdatei 1 stehende Prozedur kann mit @DO 1 aufgerufen werden.

Beispiel 2

```
   1.     @PROC 7 ------------------------------------------------------ (01)
   1.     @PROC --------------------------------------------------------- (02)
<07>
   1.     @ @SET #S7 = 'HIER SPRICHT PROC 7'
   2.     @ @PRINT #S7
   3.     @PROC 8 ------------------------------------------------------ (03)
    1.    @ @SET #S8 = 'HIER SPRICHT PROC 8'
   2.     @PROC USED
<01>     1.0000 TO    3.0000   ------------------------------------- (04)
<07>     1.0000 TO    2.0000
   2.     @END ---------------------------------------------------------- (05)
   3.     @PROC --------------------------------------------------------- (06)
<07>
   3.     @END ---------------------------------------------------------- (07)
   1.     @PROC --------------------------------------------------------- (08)
<00>
```

(01) Es wird in Arbeitsdatei 7 umgeschaltet.

(02) Abfrage der aktuellen Arbeitsdatei.

(03) Es wird in Arbeitsdatei 8 umgeschaltet.

(04) Auf die Frage, welche Arbeitsdateien belegt sind, wird die Arbeitsdatei 8 noch nicht erwähnt, da sie noch nicht mit @END abgeschlossen wurde.

(05) Der EDT kehrt wieder in die Arbeitsdatei 7 zurück (von hier aus wurde mit @PROC in die Arbeitsdatei 8 verzweigt).

(06) Die Abfrage der aktuellen Arbeitsdatei bestätigt die Rückkehr in die Arbeitsdatei 7.

(07) Es wird wieder in Arbeitsdatei 0 zurückgekehrt.

(08) Erneute Abfrage nach der aktiven Arbeitsdatei zeigt die Rückkehr in die Arbeitsdatei 0, also in die Hauptdatei.

@ERAJV Jobvariablen löschen

@ERAJV löscht Einträge von Jobvariablen aus dem Katalog. Ist das Subsystem „Jobvariablen-Support“ nicht installiert, wird diese Anweisung mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@ERAJV	str [**A**LL]	

str — Auswahl der Jobvariablen, die gelöscht werden sollen. Es sind alle Angaben erlaubt, die im BS2000-Kommando DELETE-JV angegeben werden dürfen, solange die Länge von 54 Zeichen nicht überschritten wird.

Der Jobvariablen-Name kann auch teilqualifiziert angegeben werden oder die Jobvariable kann mit ihrem Kettungsnamen angesprochen werden. Es kann auch ’*str*’ angegeben werden. Der EDT nimmt dann die Auswahl der Namen nach der wildcard-Syntax (analog zum BS2000-Kommando SHOW-FILE-ATTRIBUTES) selbst vor.

Sonst wird der Operand vom EDT nicht überprüft, sondern unverändert an das System weitergegeben.

Falls der Jobvariablen-Name teilqualifiziert bzw. als ’*str*’ angegeben wurde, braucht der EDT einen Puffer von 8 Speicher-Seiten, den er vom System zusätzlich anfordert.

Wenn mehr als ein Jobvariablen-Name die Bedingung erfüllt und der Parameter ALL nicht angegeben ist, gibt der EDT im Dialog eine zusätzliche Frage zur Weiterverarbeitung aus. Im Stapelbetrieb wird die Anweisung in diesem Fall nicht ausgeführt.

ALL — Wird ALL angegeben, werden alle Jobvariablen, auf die der Name zutrifft, ohne Nachfrage aus dem Katalog entfernt.

Wird keine Jobvariable mit entsprechendem Namen gefunden oder ein DELETE-JV vom System abgewiesen, meldet der EDT einen Fehler und setzt den EDT-Schalter für DVS-Fehler. DVS-Fehler können in EDT-Prozeduren mit @IF, Format 1 abgefragt werden.

@EXEC Programm starten

@EXEC bewirkt, daß

– die EDT-Sitzung beendet,
– das angegebene Programm geladen und gestartet wird.

Operation	Operanden	F-Modus / L-Modus
@**EX**EC	string	

string Zeichenfolge, die den Namen des Programms angibt, das geladen und gestartet werden soll.
string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Existieren noch ungesicherte Arbeitsdateien, so werden nach der Meldung:

% EDT0900 EDITED FILE(S) NOT SAVED!

die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben.

Zusätzlich wird ausgegeben, falls vorhanden, entweder

– ein lokaler @FILE-Eintrag
  – explizit definiert durch @FILE LOCAL, oder
  – implizit definiert durch @READ, @GET, @OPEN (Format 1)
– oder der Bibliotheks- und Elementsname eines mit
  – @OPEN (Format 2) eröffneten Bibliothekselements
– oder der Dateiname einer mit
  – @OPEN (Format 2) eröffneten SAM-oder ISAM-Datei
  – @XOPEN eröffneten POSIX-Datei

Danach folgt die Anfrage an den Benutzer:

% EDT0904 TERMINATE EDT? REPLY (Y=YES, N=NO)

N: Im F-Modus erscheint das Arbeitsfenster wieder. Der Benutzer kann ungesicherte Dateien schließen und zurückschreiben.

Y: Die ungesicherten, virtuellen Dateien gehen verloren. Der EDT wird beendet, das angegebene Programm gestartet.

Wurde eine Datei real mit @OPEN eröffnet, entfällt diese Abfrage. Der EDT schließt die Datei durch ein implizites @CLOSE.

Die Sicherungsabfrage kann unterdrückt werden, indem man vor dem Aufruf des EDT den Auftragsschalter 4 einschaltet.

Beispiel

```
   1.00 @EXEC bewirkt, dass.....................................................
   2.00 -  die EDT-Sitzung beendet,..............................................
   3.00 -  das angegebene Programm geladen und gestartet wird. ..................
   4.00 ........................................................................
exec '$lms'.....................................................0001.00:001(0)
```

Der EDT soll beendet und der LMS geladen und gestartet werden.

```
%  EDT0900 EDITED FILE(S) NOT SAVED!
  LOCAL FILE ( 0) :
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)?y
%  BLS0500 PROGRAM 'LMS', VERSION 'V3.0A' OF 'yy-mm-dd' LOADED.
LMS0310 LMS VERSION V03.0A00 LOADED
CTL=(CMD)                                                    PRT=(OUT)
$
```

Da die Arbeitsdatei noch nicht gesichert wurde, fragt der EDT wie bei @HALT nach, ob er tatsächlich beendet werden soll. Erst nach Antwort Y wird der EDT beendet und der LMS geladen und gestartet.

@FILE Dateiname voreinstellen

Mit @FILE kann man für @GET, @READ, @WRITE, @SAVE, @OPEN und @ELIM einen Dateinamen voreinstellen.

Es gibt

– einen lokalen @FILE-Eintrag, der nur arbeitsdateispezifisch wirkt,
– einen globalen @FILE-Eintrag, der auf alle Arbeitsdateien wirkt.

Wird bei @GET, @READ, @WRITE, @SAVE und @ELIM kein Dateiname angegeben, wird, falls vorhanden, der lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag (falls vorhanden) als Dateiname verwendet.

Bei @OPEN (Format 1) wird nur der globale @FILE-Eintrag ausgewertet.

Operation	Operanden	F-Modus / L-Modus
@**F**ILE	[string [(ver)]] [**LOC**AL]	

string Zeichenfolge, die einen Dateinamen angibt.

string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

ver Versionsnummer der Datei.

Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer.
Wird LOCAL angegeben, darf keine Versionsnummer angegeben werden.

LOCAL Der angegebene Dateiname wird als arbeitsdateispezifischer Dateiname der aktuellen Arbeitsdatei vermerkt (expliziter lokaler Eintrag). Existiert bei Ausführung von @READ 'file' bzw. @GET 'file' noch kein expliziter lokaler Eintrag, wird der angegebene Dateiname zum lokalen Dateinamen (impliziter lokaler Eintrag).

Wird LOCAL nicht angegeben, wird der angegebene Dateiname als globaler Eintrag vermerkt.

Der lokale @FILE-Eintrag wird gelöscht durch

– die Eingabe von @FILE LOCAL ohne string.
– das vollständige Löschen der Arbeitsdatei mit @DELETE, Format 1.
– das Schließen einer real eröffneten Datei mit @CLOSE.

Der globale @FILE-Eintrag wird durch die Eingabe von @FILE ohne Operanden gelöscht.

Beispiel

```
  23.00 ..........................................................................
file 'bsp.file' (*) ; get .....................................0000.00:001(0)
```

Für die folgenden @GET und @SAVE wird der Dateiname BSP.FILE samt Stern als Versionsnummer voreingestellt.

Anschließend wird mit @GET die Datei BSP.FILE eingelesen.

```
   1.00 EINS.....................................................................
   2.00 ZWEI.....................................................................
   3.00 DREI.....................................................................
   4.00 VIER.....................................................................
   5.00 FUENF.....................................................................
   6.00 .....................................................................




%  EDT0902 FILE 'BSP.FILE' VERSION 002
delete 1-2 ; save ..........................................0001.00:001(0)
```

In der Arbeitsdatei wird der Zeilenbereich 1 bis 2 gelöscht und anschließend der Inhalt der Arbeitsdatei mit @SAVE in die Datei BSP.FILE geschrieben.

```
   3.00 DREI.....................................................................
   4.00 VIER.....................................................................
   5.00 FUENF.....................................................................
   6.00 .....................................................................




%  EDT0903 FILE 'BSP.FILE' IS IN THE CATALOG, FCBTYPE = ISAM
y  EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO) ..................0001.00:001(0)
```

```
   3.00 DREI.....................................................................
   4.00 VIER.....................................................................
   5.00 ACHT.....................................................................
   6.00 .........................................................................
%  EDT0902 FILE 'BSP.FILE' VERSION 003
.....................................................................0003.00:001(0)
```

@FSTAT Kataloginformationen abfragen

Mit @FSTAT kann der Benutzer abfragen, welche Dateien unter einer bestimmten Benutzerkennung vorhanden sind und welche Eigenschaften diese Dateien haben.

Die Informationen können

– am Bildschirm ausgegeben werden,
– in eine Arbeitsdatei geschrieben werden.

Die Liste ist alphabetisch sortiert.

Operation	Operanden	F-Modus / L-Modus
@**FS**TAT	{['file'] / str-var} [[**T**O] ln [(inc)]] [{**S**HORT / **LONG**}]	

Das Schlüsselwort TO kann nur dann weggelassen werden, wenn ’file’ angegeben wird

file — Auswahl der Dateien, die ausgegeben werden sollen. file entspricht pfadname des BS2000-Kommandos SHOW-FILE-ATTRIBUTES. Der Operand wird vom EDT nicht geprüft, sondern unverändert an das System weitergegeben. Es sind alle Angaben erlaubt, die auch im Systemkommando für den Operanden FILE-NAME gegeben werden können.

Wird keine Datei mit entsprechendem Namen gefunden, meldet der EDT einen Fehler. In EDT-Prozeduren kann der EDT-Fehlerschalter mit @IF, Format 1 abgefragt werden.

Bei Angabe einer CATID im Operand file wird die Liste der Dateinamen mit CATID und USERID ausgegeben (siehe SHORT).

str-var — Die Auswahl der Dateien, die ausgegeben werden sollen, kann auch durch eine Zeichenfolgevariable (#S1-#S20) angegeben werden.

ln — Zeilennummer, ab der die Dateinamen (Kataloginformationen) in die aktuelle Arbeitsdatei geschrieben werden. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

Ist ln nicht angegeben, wird das Ergebnis
– im L-Modus am Bildschirm ausgegeben.
– im Stapelbetrieb auf SYSOUT ausgegeben.
– im F-Modus in die Arbeitsdatei 9 geschrieben, die vorher gelöscht wird.

inc Schrittweite, aus der die auf ln folgenden Zeilennummern gebildet werden sollen. Wird inc nicht angegeben, wird die implizit gegebene Schrittweite verwendet.

SHORT Es wird nur eine Liste der Dateinamen mit CATID und USERID ausgegeben.

Wird der Pfadname vollqualifiziert angegeben, so wird der Dateiname so ausgegeben, wie er eingegeben wurde.

LONG Zusätzlich zu den Dateinamen werden weitere Kataloginformationen ausgegeben.

Falls ln nicht angegeben wurde und @PAR INFORMATION=ON eingeschaltet ist, wird im F-Modus eine Überschriftzeile zur Beschreibung der Kataloginformationen ausgegeben.

Spalte	**Überschrift**	**Bedeutung**
1-7	SIZE	Anzahl der PAM Pages
8	P	Datei auf privatem oder gemeinschaftlichem Datenträger
9-62	FILENAME	Dateiname mit CATID und USERID
63-69	LAST PP	Last Page
71-78	CR-DATE	Erstellungsdatum (YY-MM-DD)
80	S	SHARE-Attribut (Y/N)
81	A	ACCESS-Attribut (W/R)
83-86	FCB	FCB-Typ (SAM/ISAM/PAM/BTAM)
88	R	READ-PASS-Attribut (Y/N)
89	W	WRITE-PASS-Attribut (Y/N)

Wird bei @FSTAT weder SHORT noch LONG angegeben, so erfolgt die Ausgabe der Kataloginformationen wie bisher, d.h. es wird eine Liste der Dateinamen ohne CATID und USERID ausgegeben.

Bei der Angabe von LONG im F-Modus beträgt die Ausgabelänge für jede Datei 89 Zeichen und überschreitet damit die Arbeitsfensterbreite von maximal 80 Zeichen (bei @PAR INDEX=OFF).

Ist ln angegeben, wird die aktuelle Zeilennummer verändert, wenn eine Zeile angelegt wurde, deren Nummer größer als die bisherige höchste Zeilennummer ist.

Beispiel

```
  23.00 ..........................................................................
fstat '*bsp*' long ; edit long ................................0001.00:001(0)
```

Ausführliche Informationen über alle Dateien, deren Namen die Zeichenfolge BSP enthalten, werden ausgegeben.

Um die Informationen komplett im Datenfenster zu sehen, wird EDIT LONG angegeben.

```
0000003 :N:$USER.BSP.FSTAT                                  0000003 94-01-12 N
W ISAM NN.....................................................................
0000021 :N:$USER.BSP.FSTAT.1                                0000021 94-01-11 N
W SAM  NN.....................................................................
0000015 :N:$USER.EDT.FSTAT.BSP                              0000015 94-01-10 N
W SAM  NN.....................................................................
0000006 :N:$USER.PROG.BSP                                   0000006 94-01-12 Y
R PAM  NY.....................................................................




.............................................................0001.00:001(9)
```

Die Informationen wurden in der Arbeitsdatei 9 abgelegt.

@GET Einlesen einer ISAM-Datei

Mit @GET wird eine ISAM-Datei von Platte oder Band ganz oder teilweise in die aktuelle Arbeitsdatei eingelesen bzw. kopiert.

Die Datei ist nur während der Ausführung von @GET physikalisch geöffnet. Bearbeitet wird die eingelesene Kopie der ursprünglichen ISAM-Datei.

Für die einzulesende ISAM-Datei nimmt der EDT standardmäßig variable Satzlänge an (Einlesen einer Datei mit fester Satzlänge siehe Abschnitt „Bearbeiten von ISAM-Dateien mit vom Standard abweichenden Attributen“ auf Seite 41ff.).

Operation	Operanden	F-Modus / L-Modus
@**G**ET	['file'] [(ver)] [range*] [:col:] [**N**ORESEQ]	

Soll der ISAM-Schlüssel als Zeilennummer interpretiert werden, so muß NORESEQ zwingend angegeben werden.

file Dateiname. Besteht noch kein lokaler @FILE-Eintrag für den Dateinamen, so wird der angegebene Dateiname eingetragen. Fehlt der Operand file, so wird, falls vorhanden, der lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE). Ist die ISAM-Datei nicht vorhanden, wird @GET mit einer Fehlermeldung abgewiesen.

Wenn der Dateikettungsname EDTISAM einer Datei zugeordnet ist, genügt die Angabe /, um diese Datei einzulesen (siehe Abschnitt „Dateibearbeitung“ auf Seite 40ff.).

ver Versionsnummer der Datei.

Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer. Wird * angegeben, erscheint vor dem Einlesen die aktuelle Versionsnummer auf dem Bildschirm. Wird eine falsche Versionsnummer angegeben, erscheint die richtige auf dem Bildschirm und diese Datei wird eingelesen.

range* Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

Fehlt range*, werden alle Zeilen eingelesen.

col Spaltenbereich bestehend aus:
- einer oder mehreren (durch Komma getrennt) Spalten (z.B. 10,15,8)
- einem oder mehreren (durch Komma getrennt) zusammenhängenden Spaltenbereichen (z.B. 15-25,18-23)
- einer Kombination von einzelnen Spalten und Spaltenbereichen (z.B. 10,14-29,23-50,17)

Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge eingelesen.

Wiederholungen und Überlappungen von Spalten und Spaltenbereichen sind erlaubt.

NORESEQ Die Zeilennummern werden aus den ISAM-Schlüsseln der eingelesenen ISAM-Datei gebildet. Dabei können Zeilen mit bereits vorhandener Zeilennummer überschrieben werden.

Im F-Modus-Arbeitsfenster werden nur 6-stellige Zeilennummern ausgegeben. Es werden deshalb nur die ersten 6 Stellen des ISAM-Schlüssels dargestellt.

Nach der Bearbeitung kann die Datei mit @SAVE (siehe @SAVE) zurückgeschrieben werden.

Ist die angegebene Datei eine SAM-Datei, wird intern ein @READ auf diese Datei gemacht. Dies wird durch eine Meldung angezeigt. range* bzw. NORESEQ werden ignoriert.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @GET der Coded Character Set Name (CCSN) der Datei berücksichtigt.

Die @GET-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei gleich dem im EDT aktuell eingestellten ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

@GETJV Wert einer Jobvariablen lesen

Mit @GETJV kann der Wert einer Jobvariablen

– am Bildschirm ausgegeben werden,
– in eine Arbeitsdatei geschrieben werden,
– einer Zeichenfolgevariablen zugeordnet werden.

Ist das Subsystem „Jobvariablen-Support“ nicht installiert, wird diese Anweisung mit der Fehlermeldung % EDT5254 JVS NOT IN SYSTEM abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@GETJV	[string] [=line]	

string Zeichenfolge, die einen vollqualifizierten Jobvariablen-Namen angibt.

string kann angegeben werden:
– explizit als Zeichenfolge in Hochkomma
– implizit über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Wird string nicht angebenen, wird die Jobvariable mit dem Kettungsnamen *EDTLINK angesprochen.

line Nummer der Zeile, in die der Wert der Jobvariablen geschrieben werden soll.
line kann auch durch Zeilennummervariable (#L0-#L20), durch symbolische Zeilennummern (z.B. %.$) oder als Zeichenfolgevariable (#S0-#S20) angegeben werden.
Wird line nicht angegeben, wird der Wert der Jobvariablen am Bildschirm ausgegeben.

Ist line angegeben, wird die aktuelle Zeilennummer verändert, wenn eine Zeile angelegt wurde, deren Nummer größer als die bisherige höchste Zeilennummer ist.

Beispiel

```
1.     @GETJV '$SYSJV.DATE' = 1 ---------------------------------- (01)
1.     @GETJV '$SYSJV.DATE' = #S01 ------------------------------- (02)
1.     @CREATE 2 : 'HEUTE IST DER '
1.     @CREATE 2 : 2,#S01:7-8:,'.',#S01:4-5:,'.',#S01:1-2:   --------- (03)
1.     @PRINT
1.0000 90-09-24267
2.0000 HEUTE IST DER 24.09.90
```

(01) Das aktuelle Datum wird in Zeile 1 geschrieben.

(02) Das aktuelle Datum wird in der Zeichenfolgevariablen #S01 abgelegt.

(03) Mit @CREATE wird Zeile 2 erzeugt.

@GETLIST Einlesen von Elementen einer Listenvariablen

Mit @GETLIST werden die Elemente einer Listenvariable ganz oder teilweise in die aktuelle Arbeitsdatei eingelesen.

In Systemen, in denen das Subsystem SDF-P nicht installiert ist, wird @GETLIST mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@**GETL**IST	string [range*] [:col:]	

string — Zeichenkette, die den Namen der Listenvariable angibt.
string kann angegeben werden:
- explizit als Zeichenfolge in Hochkomma
- implizit über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Die Elemente der Listenvariablen müssen vom Typ STRING sein, anderenfalls wird die Anweisung mit einer Fehlermeldung abgebrochen.

range* — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

range* gibt dem Benutzer die Möglichkeit, die Elemente einer Listenvariable teilweise einzulesen. Hierbei erfolgt eine gedachte Zuordnung von Elementnummern und Zeilennummern, wobei 0.0001 für das 1. Element in der Liste steht, 0.0002 für das 2. Element, usw. Die eingelesenen Elemente werden an den Inhalt der Arbeitsdatei oder der durch @OPEN eröffneten Datei angehängt. Sie erhalten dort die Zeilennummer, die sich aus der aktuellen Zeilennummer und der aktuellen Schrittweite ergibt.
range* bezieht sich auf die Elementnummer. Mit 0.0001-0.0005 werden die ersten 5 Elemente der Liste eingelesen.

Fehlt range*, werden alle Elemente eingelesen.

col Spalten oder Spaltenbereiche
Wird kein Spaltenbereich angegeben, wird das Element in voller Länge eingelesen.
Ist der Inhalt des Elementes länger als 256 Zeichen, werden nur die ersten 256 Zeichen übernommen. Eine Warnung wird ausgegeben.
Wiederholungen und Überlappungen von Spalten und Spaltenbereichen sind erlaubt.
Ist das Listenelement leer oder kürzer als die niedrigste angegebene Spalte, wird dafür keine Zeile angelegt.

Wird die maximale Zeilennummer erreicht, wird die Anweisung abgebrochen und eine Fehlermeldung ausgegeben.

Vergabe der Zeilennummern

Die Zeilennummern werden abhängig von der aktuellen Zeilennummer und der aktuellen Schrittweite vergeben. Bei leerer Arbeitsdatei sind die aktuelle Zeilennummer und die aktuelle Schrittweite standardmäßig 1.

In SDF-P V2.0 braucht der EDT zusätzlich einen Puffer von 8 Speicher-Seiten, den er vom System zusätzlichh anfordert.

@GETVAR Lesen einer S-Variablen

Mit @GETVAR kann man den Inhalt einer S-Variablen vom Typ STRING und vom Typ INTEGER:

– am Bildschirm ausgeben lassen.
– einer Zeichenfolgevariablen als Wert zuweisen (max. 256 Zeichen).

In Systemen, in denen das Subsystem SDF-P nicht installiert ist, wird @GETVAR mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@**GETV**AR	{ string [=line | =int-var] / **SYSEDT** }	

string — Zeichenfolge, die den Namen einer einfachen S-Variablen angibt.
string kann angegeben werden:
– explizit als Zeichenfolge in Hochkomma
– implizit über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

line — Nummer der Zeile, in die der Wert der S-Variablen vom Typ STRING geschrieben werden soll.
line kann auch durch Zeilennummernvariablen (#L00 bis #L20), symbolisch (z.B. %,$) oder als Zeichenkettenvariable (#S00 bis #S20) angegeben werden.

int-var — Ganzzahlvariable (#I0 bis #I20), in die der Inhalt der S-Variablen übernommen werden soll.
Ist der Typ der S-Variablen nicht INTEGER, wird die Abarbeitung abgebrochen und eine Fehlermeldung ausgegeben.

Ist weder int-var noch line angegeben, wird der Inhalt der S-Variablen im Dialog auf den Bildschirm und im Stapelbetrieb auf SYSLST ausgegeben.

Wenn eine Zeile angelegt wurde, deren Nummer größer als die bisherige höchste Zeilennummer ist, wird die aktuelle Zeilennummer. verändert.

SYSEDT — Den Zeichenfolgevariablen #S00 bis #S20 werden die Inhalte der S-Variablen SYSEDT-S00 bis SYSEDT-S20 zugewiesen. Wird der Operand SYSEDT angegeben, werden keine Meldungen über Erfolg oder Mißerfolg bei der Wertzuweisung und auch keine Meldung % EDT5341 ausgegeben, wenn der Inhalt der S-Variablen länger als 256 Zeichen ist. Es werden keine Fehlerschalter gesetzt.

@GOTO Springen zu Zeilennummern in Prozeduren

Mit @GOTO wird während der Ausführung einer Prozedur auf eine angegebene Zeile innerhalb dieser Prozedur gesprungen.

Operation	Operanden	@PROC
@**GO**TO	ln	

ln Zeilennummer (z.B. 5).
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

Die mit @GOTO angesprungene Zeile muß in der zugehörigen Prozedur existieren. Wird versucht eine Zeile anzuspringen, die nicht existiert, kommt es zu der Fehlermeldung
% EDT4974 LINE NOT IN PROCEDURE FILE.

Sollen in EDT-Prozeduren Zeilen über @GOTO angesprungen werden, ist es bei diesen Zeilen sinnvoll, mit @ln immer die Zeilennummer anzugeben.

Beispiel

```
   1.     @SET #I3 = 1 ------------------------------------------------ (01)
   1.     @PROC 3
   1.     @1 ---------------------------------------------------------- (02)
   1.     @ @IF #I3 > 5 RETURN ---------------------------------------- (03)
   2.     @ @STATUS = #I3
   3.     @ @SET #I3 = #I3+1
   4.     @ @GOTO 1
   5.     @END
   1.     @DO 3 ------------------------------------------------------- (04)
#I03= 0000000001
#I03= 0000000002
#I03= 0000000003
#I03= 0000000004
#I03= 0000000005
   1.
```

(01) Der Ganzzahlvariablen #I3 wird der Wert 1 zugewiesen.

(02) Angabe der Zeilenummer bei der Zeile, die über @GOTO angesprungen wird.

(03) Bei Ausführung der Prozedur in der Arbeitsdatei 3 soll der Wert, den die Ganzzahlvariable #I3 hat, solange um 1 erhöht und angegeben werden, bis er größer als 5 ist. Realisiert wird dies durch eine Schleife. Am Ende der Schleife wird mit @GOTO zum Schleifenanfang gesprungen.

(04) Die Prozedur in Arbeitsdatei 3 wird ausgeführt.

@HALT Beenden des EDT

@HALT beendet

- den EDT-Lauf,
- den Bildschirmdialog nach @DIALOG,
- den EDT als Unterprogramm mit oder ohne Übergabe eines Textes.

@HALT bewirkt:

- Beenden des EDT-Laufes
  - im Bildschirmdialog, wenn dieser mit dem Kommando START-EDT, START-PROGRAM $EDT oder START-PROGRAM *MOD($EDTLIB,EDTC) eingeleitet wurde.
  - in BS2000-Systemprozeduren mit EDT-Aufruf.
  - bei einem Unterprogrammaufruf, wenn es in der CMD-Funktion (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]; Ausführen einer EDT-Anweisung oder EDT- Anweisungsfolge) beim Abarbeiten einer Anweisungsfolge gegeben wird.
- Beenden des Bildschirmdialogs nach @DIALOG
  - mit Rückkehr in eine Systemprozedur.
  - mit Fortführung eines Unterprogrammaufrufes.

Operation	Operanden	F-Modus / L-Modus
@**H**ALT	[**ABNORMAL**] [message]	

ABNORMAL Wurde der EDT als Hauptprogramm aufgerufen, wird der EDT abnormal beendet. In Prozeduren wird auf den nächsten JOB-STEP aufgesetzt oder die Verarbeitung in einem ERROR-BLOCK angestoßen.

Wurde EDT als Unterprogramm aufgerufen, wird der Zeichenstring als Teil der message an das rufende Programm übergeben und ein spezieller Returncode gesetzt.

message Zeichenfolge, die bei Aufruf des EDT als Unterprogramm an das aufrufende Programm übergeben wird. Sie beginnt beim ersten Zeichen ungleich Leerzeichen nach @HALT und endet mit dem Anweisungsende.
Im F-Modus erkennt der EDT das Anweisungsende auch durch ein Semikolon (;) in message.

Zwischen @HALT und message muß mindestens ein Leerzeichen stehen.

Die Länge von message im F-Modus wird durch die Anweisungslänge in der Anweisungszeile begrenzt. (maximal zwei Fortsetzungszeilen).
Es werden aber max. 80 Zeichen an das aufrufende Programm übergeben.

Dieser Operand darf nur angegeben werden, wenn der EDT als Unterprogramm aufgerufen wird. Wurde der EDT nicht als Unterprogramm aufgerufen, wird @HALT message mit einer Fehlermeldung abgewiesen.

Existieren noch ungesicherte Arbeitsdateien, so werden nach der Meldung:
% EDT0900 EDITED FILE(S) NOT SAVED! die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben.

Zusätzlich wird ausgegeben, falls vorhanden, entweder

- ein lokaler @FILE-Eintrag
  - explizit definiert durch @FILE LOCAL, oder
  - implizit definiert durch @READ, @GET, @OPEN (Format 1)
- oder der Bibliotheks- und Elementsname eines mit
  - @OPEN (Format 2) eröffneten Bibliothekselements
- oder der Dateiname einer mit
  - @OPEN (Format 2) eröffneten SAM-oder ISAM-Datei
  - @XOPEN eröffneten POSIX-Datei

Danach folgt die Anfrage an den Benutzer:

% EDT0904 TERMINATE EDT? REPLY (Y=YES, N=NO)

N: Im F-Modus erscheint das Arbeitsfenster wieder. Der Benutzer kann ungesicherte Dateien schließen und zurückschreiben.

Y: Die ungesicherten, virtuellen Dateien gehen verloren. Der EDT wird beendet.

Durch Einschalten von Auftragsschalter 4 vor dem EDT-Aufruf wird die Sicherungsabfrage unterdrückt. Die Sicherungsabfrage unterbleibt auch, wenn der F-Modus mit @DIALOG aufgerufen wurde.

Beispiel

```
%  EDT0900 EDITED FILE(S) NOT SAVED!
  LOCAL FILE ( 0) :
  LOCAL FILE ( 1) :
  LOCAL FILE ( 4) : L= EDT164
                    E= HALT,X
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)
```

@IF Abfrage von Zeichenfolgen, Zeilennummern, Zahlen und Schaltern

Mit @IF kann man

- abfragen, ob in einer vorangegangenen Anweisung ein EDT- oder DVS-Fehler aufgetreten ist (Format 1).
- Zeichenfolgen, Zeilennummern oder Ganzzahlen miteinander vergleichen (Format 2).
- abfragen, ob die aktuelle Arbeitsdatei leer ist (Format 3).
- feststellen, ob bei einem vorangegangenen @ON ein Treffer festgestellt wurde (Format 3).
- die 32 Auftrags- und 32 Benutzerschalter abfragen, ob sie ein- oder ausgeschaltet sind (Format 4).

@IF (Format 1) Abfrage von Fehlerschaltern

Mit diesem Format wird geprüft, ob zuvor EDT- oder DVS-Fehler aufgetreten sind. Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.

EDT-Fehler treten beispielsweise durch die Eingabe einer nicht korrekten EDT-Anweisung auf. Der DVS-Fehlerschalter kann bei Anweisungen gesetzt werden, bei denen auf Dateien zugegriffen wird (z.B. @WRITE Format 1) oder zeigt Fehler bei Systemzugriffen an.

Operation	Operanden	L-Modus / @PROC
@IF	{ **E**RRORS / **N**O ERRORS / **D**MS ERRORS / **N**O **D**MS ERRORS } : text	

text Beliebige Zeichenfolge.

Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol, werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   - text steht in der aktuellen Zeile;
   - die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   - vorhandene Tabulatorzeichen werden berücksichtigt.

2. Anweisungssymbol, werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   - kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   - Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Wird text nicht angegeben, ist @IF wirkungslos.

ERRORS text wird ausgeführt, wenn zuvor der EDT-Fehlerschalter gesetzt wurde.

NO ERRORS text wird ausgeführt, wenn der EDT-Fehlerschalter nicht gesetzt ist.

DMS ERRORS
text wird ausgeführt, wenn zuvor der DVS-Fehlerschalter gesetzt wurde.

NO DMS ERRORS
text wird ausgeführt, wenn der DVS-Fehlerschalter nicht gesetzt ist.

- Vor der zu prüfenden Anweisung müssen die Fehlerschalter zurückgesetzt werden (siehe @RESET). @IF kann sonst ein unerwünschtes Ergebnis liefern, da auch bereits vorhergegangene Anweisungen den EDT- oder DVS-Schalter gesetzt haben könnten.
- Innerhalb von EDT-Prozeduren (@DO) darf @IF ERRORS nicht zur Trefferabfrage nach @ON eingesetzt werden. Es ist dazu @IF Format 3 zu verwenden.

@IF (Format 2) Abfrage von Zeichenfolgen, Zeilennummern und Zahlen

Dieses Format von @IF vergleicht

- Zeileninhalte oder Zeichenfolgevariablen miteinander,
- Zeilennummern oder Zeilenummernvariablen miteinander,
- Ganzzahlvariablen miteinander.

Trifft die Vergleichsbedingung zu, wird

- zu einer Zeile der Prozedur verzweigt (GOTO ln),
- die Ausführung der aktuellen Prozedur abgebrochen (RETURN).

Trifft die Vergleichsbedingung nicht zu, setzt der EDT die Abarbeitung der Prozedur in der Zeile fort, die auf @IF folgt.

Operation	Operanden	@PROC
@IF	{ [**S**] string1 rel string2 / ln1 rel ln2 / [**I**] int1 rel int2 } { **G**OTO ln / **R**ETURN }	

S
: Wird nur dann zwingend benötigt, wenn in string1 und string2 Zeilennummern ohne Spaltenangaben stehen. Dann kann der EDT nämlich nicht mehr unterscheiden, ob der Inhalt der Zeilen oder die Zeilennummern verglichen werden sollen.

  Beispielsweise werden mit @IF#L1 = #L2... die Zeilennummern #L1 und #L2 miteinander verglichen. Will man jedoch die Inhalte der Zeilen #L1 und #L2 miteinander vergleichen, muß @IF S #L1 = #L2 angegeben werden.

string1
string2
: Miteinander zu vergleichende Zeichenfolgen.

  string kann angegeben werden:
  - explizite Angabe in Hochkomma,
  - implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

  Beispielsweise ist 'HUGO' eine zulässige Zeichenfolge. Befindet sich in der Zeichenfolgevariablen #S18 der Text 'ABCD456DEF', so ist die Angabe der Zeichenfolge '456ABCE' gleichwertig mit der Angabe #S18:5-7,1-3,9:.

ln1
ln2 Miteinander zu vergleichende Zeilennummern, es werden nicht die Zeileninhalte verglichen. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

I Muß nur dann angegeben werden, wenn mit int1 eine Zahl angegeben wird (sonst erkennt der EDT nicht, ob es sich um eine Zeilennummer oder um eine Ganzzahl handelt).

int1
int2 Miteinander zu vergleichende Ganzzahlen.

Es kann jeweils eine ganze (positive oder negative) Zahl oder eine Ganzzahlvariable (#I0,...,#I20) angegeben werden.

rel Vergleichsrelation:

Symbol			**Erläuterung**
GT	bzw.	>	größer als
LT	bzw.	<	kleiner als
GE	bzw.	>=	größer oder gleich
LE	bzw.	<=	kleiner oder gleich
EQ	bzw.	=	gleich
NE	bzw.	<>	ungleich

ln könnte sowohl %+3L als auch %+3 sein. Bei einer nachfolgenden Vergleichsrelation LE kann das zu Interpretationsschwierigkeiten führen. Deshalb ist es generell empfehlenswert, die mathematischen Zeichen für die Vergleichsrelationen zu benutzen.

GOTO ln Trifft die Vergleichsoperation zu, wird in der Prozedur zur angegebenen Zeile ln verzweigt.

RETURN Trifft die Vergleichsoperation zu, wird die aktuelle Prozedur abgebrochen.

Vergleich von Zeichenfolgen

Der Vergleich zweier Zeichenfolgen hängt von der Länge dieser Zeichenfolgen ab (eine Zeichenfolge der Länge Null ist zugelassen).

Fall 1: Beide Zeichenfolgen haben dieselbe Länge.

Von links nach rechts werden die sich entsprechenden Zeichen der beiden Zeichenfolgen miteinander verglichen.
Auf diese Weise erreicht man entweder ein ungleiches Zeichenpaar, oder aber die beiden Zeichenfolgen werden als identisch erkannt. Für den Fall der Ungleichheit an einer Stelle steht die Ungleichheit der beiden Zeichenfolgen fest. Der EDT interpretiert die beiden ungleichen Zeichen gemäß ihres EBCDI-Codes als Dualzahlen. Größer ist die Zeichenfolge, deren Zeichen die größere Dualzahl darstellt.

Fall 2: Die beiden Zeichenfolgen haben unterschiedliche Länge.

Der Vergleich läuft im Prinzip wie im Fall 1 ab. Wenn der EDT alle Zeichen in der Länge der kürzeren Zeichenfolge betrachtet und dabei kein ungleiches Zeichenpaar gefunden hat, ist die längere der beiden Zeichenfolgen die größere. Wird bereits vorher ein ungleiches Zeichenpaar gefunden, interpretiert der EDT die beiden ungleichen Zeichen gemäß ihres EBCDI-Codes als Dualzahlen. Größer ist die Zeichenfolge, deren Zeichen die größere Dualzahl darstellt.

Sind die beiden Zeichenfolgen unterschiedlich lang, kann niemals Gleichheit der beiden Zeichenfolgen vorliegen.

Beispiel 1

```
   4.     @PRINT
   1.0000 BITTE NICHT LACHEN
   2.0000 WEGEN DIESES BEISPIELS
   3.0000 BITTE NICHT LACHEN
   4.     @SET #S0 = 'ERSTE ZEILE = LETZTE ZEILE'
   4.     @SET #S1 = 'ERSTE ZEILE UNGLEICH LETZTE ZEILE'   ------------- (01)
   4.     @SET #I9 = 2
   4.     @PROC 3
   1.     @ @IF S %+#I9 = $-2L GOTO 4 ---------------------------------- (02)
   2.     @ @PRINT #S1 N
   3.     @ @RETURN
   4.     @ @PRINT #S0 N
   5.     @END
   4.     @DO 3 -------------------------------------------------------- (03)
ERSTE ZEILE = LETZTE ZEILE
   4.     @ON 1 DELETE 'NICHT'
   4.     @DO 3 -------------------------------------------------------- (04)
ERSTE ZEILE UNGLEICH LETZTE ZEILE
   4.
```

(01) Die Zeichenfolgevariablen #S0 und #S1 und die Ganzzahlvariable #I9 werden mit Inhalt versehen.

(02) Bei Ausführung der Arbeitsdatei 3 werden an dieser Stelle nicht Zeilennummern, sondern Zeileninhalte miteinander verglichen.

(03) Die Ausführung der in Arbeitsdatei 3 abgelegten Anweisungen führt zum Vergleich der Zeilen 3 (%+#I9) und 1 ($-2L also 3-2). Da die Inhalte dieser beiden Zeilen identisch sind, wird zu der Zeile 4 der Arbeitsdatei 3 verzweigt.

(04) Da sich inzwischen der Inhalt von Zeile 1 geändert hat, wird nicht zu der Zeile 4 der Arbeitsdatei 3 verzweigt.

Beispiel 2

```
   1.     @SET #S4 = 'M' ------------------------------------------- (01)
   1.     @PROC 4
   2.     @ @PRINT #S4 N ------------------------------------------- (02)
   2.     @ @CREATE #S4: 'M',#S4
   3.     @ @IF #S4 < 'M'*8 GOTO 1
   4.     @END
   1.     @DO 4
M
MM
MMM
MMMM
MMMMM
MMMMMM
MMMMMMM
   1.
```

(01) Die Zeichenfolgevariable #S4 erhält das Zeichen M als Inhalt.

(02) In Arbeitsdatei 4 wird folgende Prozedur eingegeben: Der Inhalt von #S4 soll ausgegeben werden. Danach soll vor den momentanen Inhalt von #S4 der Buchstabe M gestellt werden.

Für den Fall, daß der Inhalt von #S4 kleiner als MMMMMMMM ist, soll wieder von vorne begonnen werden.

Beispiel 3

```
9.     @PRINT
1.0000 ABC
2.0000 WER
3.0000 ABC
4.0000 WILL
5.0000 ABC
6.0000 HIER
7.0000 ABC
8.0000 MITMACHEN?
9.     @PROC 6
1.     @ @IF ! <> 'ABC' GOTO 3 ------------------------------------ (01)
2.     @ @CREATE !: '*' * 20
3.     @ @CONTINUE
4.     @END
9.     @DO 6,!=%,$ ------------------------------------------------ (02)
9.     @PRINT
1.0000 ********************
2.0000 WER
3.0000 ********************
4.0000 WILL
5.0000 ********************
6.0000 HIER
7.0000 ********************
8.0000 MITMACHEN?
9.
```

(01) In der Arbeitsdatei 6 werden die Zeilennummern über das Schleifensymbol ! angesprochen. Sollte der Inhalt der sich gerade hinter dem Symbol ! verbergenden Zeile von ABC verschieden sein, soll sich an diesem Zeileninhalt nichts ändern. Im anderen Fall soll der Zeileninhalt durch ******************** ersetzt werden.

(02) Arbeitsdatei 6 wird ausgeführt. Hierbei sollen alle Zeichen der aktuellen Arbeitsdatei nacheinander durch das Schleifensymbol ! angesprochen werden.

Beispiel 4

```
   4.     @PRINT
   1.0000 BITTE NICHT LACHEN
   2.0000 WEGEN DIESES BEISPIELS
   3.0000 ES IST ZU EINFACH
   4.     @SET #S0 = 'VERGLEICH POSITIV'
   4.     @SET #S1 = 'VERGLEICH NEGATIV'   ------------------------------ (01)
   4.     @SET #I9 = 1
   4.     @PROC 1 ------------------------------------------------------- (02)
   1.     @ @IF %+#I9 = $-1L GOTO 4 ------------------------------------- (03)
   2.     @ @PRINT #S1 N
   3.     @ @RETURN
   4.     @ @PRINT #S0 N
   5.     @END
   4.     @DO 1 --------------------------------------------------------- (04)
VERGLEICH POSITIV
   4.     @SET #I9 = 2
   4.     @DO 1 --------------------------------------------------------- (05)
VERGLEICH NEGATIV
   4.
```

(01) Die Zeichenfolgevariablen #S0 und #S1 werden mit Inhalt versehen. Der Ganzzahlvariablen #I9 wird der Wert 1 zugewiesen.

(02) Arbeitsdatei 1 wird geöffnet.

(03) Bei einem später gegebenen @DO 1 bewirkt diese Zeile, daß die Zeilennummern % + #I9 und $ – 1L miteinander verglichen werden.

Mit % wird die erste Zeilennummer - also 1 - angesprochen.

Mit $ wird die letzte Zeilennummer - also 3 - angesprochen.

Mit $-1L wird die vorletzte Zeilennummer - also 2 - angesprochen.

(04) Die Prozedur in Arbeitsdatei 1 wird ausgeführt. Zu diesem Zeitpunkt ist die dort genannte Relation % + #I9 = $ – 1L wahr, da 1+1 = 3–1 wahr ist.

(05) Zu diesem Zeitpunkt ist die in der Arbeitsdatei 1 genannte Relation % + #I9 = $ - 1L falsch, da 1+2 = 3–1 falsch ist.

Beispiel 5

```
4.     @PRINT
1.0000 BITTE NICHT LACHEN
2.0000 WEGEN DIESES BEISPIELS
3.0000 ES IST ZU EINFACH
4.     @SET #L3 = 5
4.     @PROC 2
1.     @ @IF %+6-#L3 <> $-* RETURN ---------------------------------- (01)
2.     @ @CREATE $+1: 'ODER AUCH NICHT'
3.     @ @PRINT
4.     @END
4.     @$-1
2.     @DO 2 -------------------------------------------------------- (02)
2.     @1
1.     @DO 2 -------------------------------------------------------- (03)
1.0000 BITTE NICHT LACHEN
2.0000 WEGEN DIESES BEISPIELS
3.0000 ES IST ZU EINFACH
4.0000 ODER AUCH NICHT
1.
```

(01) Ist bei Ausführung der Arbeitsdatei 2 die hier genannte Relation nicht erfüllt (<> steht für ungleich), wird der Prozedurablauf an dieser Stelle abgebrochen.

(02) Die Arbeitsdatei 2 wird ausgeführt. Wegen * = $ – 1 = 2 ist die Aussage % + 6 – #L3 <> $ – * gleichwertig mit 1+6–5 <> 3–2 und damit wahr. Deshalb wird die Ausführung der Arbeitsdatei abgebrochen.

(03) Zu diesem Zeitpunkt gilt * = 1 und somit ist die in Arbeitsdatei 2 stehende Relation % + 6 – #L3 <> $ – * falsch, da 1+6–5 = 3–1. Deshalb werden auch die restlichen Anweisungen der Arbeitsdatei 2 ausgeführt.

Beispiel 6

```
   1.     @SET #I3 = 1 ------------------------------------------------ (01)
   1.     @PROC 7
   1.     @ @IF #I3 > 5 RETURN
   2.     @ @STATUS = #I3          ------------------------------------ (02)
   3.     @ @SET #I3 = #I3+1
   4.     @ @GOTO 1
   5.     @END
   1.     @DO 7 ------------------------------------------------------- (03)
#I03= 0000000001
#I03= 0000000002
#I03= 0000000003
#I03= 0000000004
#I03= 0000000005
   1.
```

(01) Der Ganzzahlvariablen #I3 wird der Wert 1 zugewiesen.

(02) Mit der Prozedur in Arbeitsdatei 7 sollen die Werte für die Ganzzahlvariable #I3 solange ausgegeben (@STATUS = #I3) und erhöht werden (#I3 + 1), bis zum 1. Mal #I3 einen Wert größer als 5 angenommen hat.

(03) Die Arbeitsdatei 7 wird ausgeführt.

@IF (Format 3) Abfrage @ON-Treffern oder von leeren Arbeitsdateien

Format 3 von @IF fragt ab, ob der EDT bei der letzten Ausführung von @ON einen Treffer festgestellt hat oder die aktuelle Arbeitsdatei leer ist. In Abhängigkeit vom Ergebnis der Abfrage

– verzweigt der EDT zu einer Zeile innerhalb der Prozedur (GOTO ln),
– bricht der EDT die Ausführung der aktuellen Prozedur ab (RETURN),
– setzt der EDT die Ausführung der Prozedur in der Zeile fort, die @IF folgt.

Operation	Operanden	@PROC
@IF	{ **.T**RUE**.** [rel cl] / **.F**ALSE**.** / **.E**MPTY**.** } { **G**OTO ln / **R**ETURN }	

.TRUE. GOTO ln bzw. RETURN wird ausgeführt, wenn bei der letzten Ausführung von @ON ein Treffer festgestellt wurde.

Wurden rel und cl angegeben, wird noch nicht verzweigt. Der EDT vergleicht zuerst noch die Nummer der Spalte, in der der erste festgestellte Treffer begann mit der durch cl angegebenen Spaltennummer. Erst wenn dieser Vergleich positiv ausfällt, wird GOTO ln bzw. RETURN ausgeführt.

rel Vergleichsoperation:

Symbol			**Erläuterung**
GT	bzw.	>	größer als
LT	bzw.	<	kleiner als
GE	bzw.	>=	größer oder gleich
LE	bzw.	<=	kleiner oder gleich
EQ	bzw.	=	gleich
NE	bzw.	<>	ungleich

cl Spaltennummer (Ganzzahl zwischen 1 und 256 bzw. Ganzzahlvariable). Die Spaltennummer wird mit der Nummer der Spalte verglichen, in der bei dem letzten @ON der erste Treffer begann.

.FALSE. GOTO ln bzw. RETURN wird ausgeführt, wenn bei der letzten Ausführung von @ON kein Treffer festgestellt wurde.

.EMPTY. GOTO ln bzw. RETURN wird ausgeführt, wenn die aktuelle Arbeitsdatei leer ist, d.h. keine Datenzeilen enthält.

ln Zeilennummer (z.B. 5).
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

Wurde die Bedingung nicht erfüllt, setzt der EDT die Ausführung der Prozedur in der Zeile fort, die auf @IF folgt.

Beispiel 1

```
5.     @PRINT
1.0000 WER
2.0000 WILL
3.0000 HIER
4.0000 MITMACHEN
5.     @PROC 8
1.     @ @ON ! FIND 'I'
2.     @ @IF .FALSE. GOTO 4   ---------------------------------------- (01)
3.     @ @CREATE !: '*'*20
4.     @ @CONTINUE
5.     @END
5.     @DO 8,!=%,$ ------------------------------------------------- (02)
5.     @PRINT
1.0000 WER
2.0000 ********************
3.0000 ********************
4.0000 ********************
5.
```

(01) In der Arbeitsdatei 8 werden die Zeilennummern über das Schleifensymbol ! angesprochen. Sollte in einer der damit angesprochenen Zeilen der Buchstabe I nicht enthalten sein, bleibt diese Zeile unverändert. Im anderen Fall soll der Zeileninhalt durch ********** ersetzt werden.

(02) Die Arbeitsdatei 8 wird ausgeführt. Dabei sollen alle Zeilen der Hauptdatei nacheinander durch das Schleifensymbol ! angesprochen werden.

Beispiel 2

```
5.     @PRINT
1.0000 WER
2.0000 WILL
3.0000 HIER
4.0000 MITMACHEN?
5.     @PROC 9
1.     @ @ON ! FIND 'ER'
2.     @ @IF .TRUE. = 3 GOTO 4
3.     @ @GOTO 5                              -------------------------- (01)
4.     @ @SUFFIX ! WITH ' DENN UNBEDINGT'
5.     @ @CONTINUE
6.     @END
5.     @DO 9,!=%,$ ------------------------------------------------- (02)
5.     @PRINT
1.0000 WER
2.0000 WILL
3.0000 HIER DENN UNBEDINGT
4.0000 MITMACHEN?
5.
```

(01) In der Prozedur in Arbeitsdatei 9 werden die Zeilennummern über das Schleifensymbol ! angesprochen. Sollte in einer der damit angesprochenen Zeilen in den Spalten 3 bis 4 die Zeichenfolge ER stehen, ist ihr die Zeichenfolge DENN UNBEDINGT anzuhängen. Im anderen Fall bleibt die Zeile unverändert.

(02) Die Prozedur von Arbeitsdatei 9 wird ausgeführt. Dabei werden alle Zeilen der Hauptdatei nacheinander durch das Schleifensymbol ! angesprochen.

@IF (Format 4) Abfrage von Auftrags- und Benutzerschaltern

Format 4 von @IF fragt ab, welche Auftrags- bzw. Benutzerschalter ein- oder ausgeschaltet sind (siehe auch @SETSW und Abschnitt „Auftragsschalter“ auf Seite 60ff.). In Abhängigkeit vom Ergebnis der Abfrage

– verzweigt der EDT zu einer Zeile innerhalb der Prozedur (GOTO ln),
– bricht der EDT die Ausführung der Prozedur ab (RETURN),
– setzt der EDT die Ausführung der Prozedur in der Zeile fort, die @IF folgt.

Operation	Operanden	@PROC
@IF	{ **ON** / **OF**F } = [**U**] int { **G**OTO ln / **R**ETURN }	

ON Der EDT prüft, ob ein Schalter gesetzt ist.

OFF Der EDT prüft, ob ein Schalter nicht gesetzt ist.

U Bei Angabe von U werden die Benutzerschalter abgefragt. Sonst werden die Auftragsschalter abgefragt.

int Nummer des Schalters, dessen Stellung geprüft werden soll. Es ist eine Ganzzahl zwischen 0 und 31 anzugeben (bzw. eine Ganzzahlvariable). Falls vor der Schalternummer der Parameter U angegeben ist, wird statt des Auftragsschalters int der Benutzerschalter int der eigenen Kennung geprüft.

ln Zeilennummer (z.B. 5).
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

Beispiel

```
   1.     @SET #S2 = 'SCHALTER 15 IST AUS'
   1.     @SET #S3 = 'SCHALTER 15 IST AN'
   1.     @PROC 8
   1.     @ @IF ON = 15 GOTO 4
   2.     @ @PRINT #S2 N
   3.     @ @RETURN              ---------------------------------------- (01)
   4.     @ @PRINT #S3 N
   5.     @END
   1.     @SETSW OFF = 15 ------------------------------------------------ (02)
   1.     @DO 8 ---------------------------------------------------------- (03)
SCHALTER 15 IST AUS
   1.     @SETSW ON = 15 ------------------------------------------------- (04)
   1.     @DO 8 ---------------------------------------------------------- (05)
SCHALTER 15 IST AN
   1.
```

(01) Mit der Prozedur in Arbeitsdatei 8 soll bei gesetztem Prozeßschalter 15 die Zeichenfolgevariable #S3 ausgegeben werden, anderenfalls aber die Zeichenfolgevariable #S2.

(02) Schalter 15 wird zurückgesetzt.

(03) Die Prozedur von Arbeitsdatei 8 wird ausgeführt.

(04) Schalter 15 wird gesetzt.

(05) Die Arbeitsdatei 8 wird ausgeführt.

@INPUT Eingabemodus festlegen bzw. Prozedur starten

Mit @INPUT kann man

– eine @INPUT-Prozedur aus einer SAM- oder ISAM-Datei ganz oder bereichsweise einlesen und abarbeiten (Format 1).

– eine @INPUT-Prozedur aus einer Datei oder einem Bibliothekselement starten (Format 2).

– den Eingabemodus bei der Eingabe im L-Modus festlegen (Format 3).

@INPUT (Format 1)
Starten einer @INPUT-Prozedur aus einer SAM- oder ISAM-Datei

Mit diesem Format wird eine @INPUT-Prozedur gestartet. Eine SAM- oder ISAM-Datei wird in den Benutzeradreßraum eingelesen und die eingelesenen Anweisungen und Textzeilen werden sofort abgearbeitet.

Operation	Operanden	F-Modus / L-Modus
@**I**NPUT	'file' [(ver)] [range*] [:col:] [{**R**ECORD**S** / **KEY**}] [**P**RINT]	

file — Name der SAM- oder ISAM-Datei, die eingelesen und abgearbeitet werden soll.

ver — Versionsnummer der Datei.
Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer. Wird eine falsche Versionsnummer angegeben, wird die @INPUT-Prozedur trotzdem eingelesen und abgearbeitet.

range* — Zeilenbereich, der in der ISAM- oder SAM-Datei abgearbeitet werden soll. Fehlt range*, werden alle Zeilen der Datei bearbeitet.

<table>
<tr><td colspan="3">file ist eine</td></tr>
<tr><td>ISAM-Datei</td><td colspan="2">SAM-Datei</td></tr>
<tr><td rowspan="3">range* kann sein:
– eine einzelne Zeile, z.B. 5
– ein zusammenhängender Zeilenbereich, z.B. 6-12
– eine Folge mehrerer, durch Komma voneinander getrennter Zeilen bzw. Zeilenbereiche, z.B. 6-12, 5, 19.
Zeilen bzw. Zeilenbereiche dürfen auch wiederholt angegeben werden, z.B. 5, 5, 8-10, 9.</td><td colspan="2">range* kann sein:
– eine einzelne Zeile, z.B. 5 bzw. .0005
– ein zusammenhängender Zeilenbereich, z.B. 6-12 bzw. .0006-.0012</td></tr>
<tr><td colspan="2">Ist RECORDS angegeben?</td></tr>
<tr><td>Ja</td><td>Nein</td></tr>
<tr><td>Der zu lesende Zeilenbereich ist mit Hilfe der logischen Zeilennummern anzusprechen.
Die logische Zeilennummer der 1. Zeile ist 0.0001, die der 2. Zeile 0.0002 usw.</td><td>Die Angabe von range* ist nur sinnvoll, wenn die INPUT-Datei mit @WRITE unter Benutzung von KEY erstellt wurde. In einer solchen Datei bilden die ersten 8 Zeichen jeder Zeile die Nummer, die die Zeile in der virtuellen Datei hatte. range* bezieht sich auf diese Nummern, d.h. bei Angabe von range* interpretiert der EDT die ersten 8 Zeichen jeder Zeile nicht als Zeileninhalt, sondern als Zeilennummer. Beispielsweise wird mit 5 die Zeile angesprochen, die mit 00050000 beginnt.</td></tr>
</table>

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Ihr Wert bezieht sich dann aber nicht auf die mit file angegebene Datei, sondern auf die aktuelle virtuelle bzw. mit @OPEN eröffnete Datei.

col — Spaltenbereich bestehend aus:
- einer oder mehreren (durch Komma getrennten) Spalten
- (z.B. 10,15,8)
- einem oder mehreren (durch Komma getrennten) zusammenhängenden Spaltenbereichen (z.B. 15-25,18-23)
- einer Kombination von einzelnen Spalten und Spaltenbereichen (z.B. 10,14-29,23-50,17)

Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge eingelesen.

KEY — Ist bei SAM-Dateien anzugeben, die mit @WRITE unter Verwendung von KEY erstellt wurden. Die ersten 8 Zeichen einer jeden Zeile sind in einer solchen Datei die Nummer, die die Zeile in der virtuellen Datei hatte. Durch die Angabe von KEY bei @INPUT werden diese Nummern beim Lesen der Datei nicht als Zeileninhalt, sondern als Zeilennummer interpretiert. Andernfalls würde der EDT jede Zeile der INPUT-Datei als Textzeile betrachten.

RECORDS — Ist bei SAM-Dateien anzugeben, wenn man einen Zeilenbereich mit der logischen Zeilennummer auswählt.

PRINT — Gibt jede Zeile, die in der ISAM- bzw. SAM-Datei gelesen wird, am Bildschirm aus.

Durch die Angabe von range* und col können gezielt Teile der @INPUT-Prozedur ausgewählt werden.

@INPUT darf weder in @INPUT- noch in @DO-Prozeduren abgesetzt werden.

Die Abarbeitung einer @INPUT-Prozedur wird abgebrochen, wenn
- @RETURN abgearbeitet wurde.
- in einer @IF-Anweisung mit Operand RETURN der Vergleich zutrifft.
- in einer Anweisung ein DVS-Fehler aufgetreten ist.

In der Datei enthaltene Schlüssel (KEY) werden nicht auf Richtigkeit überprüft. Die Zeilennummer wird aus den ersten 8 Zeichen einer SAM-Datei gebildet. Ist die daraus errechnete Zeilennummer kleiner als der angegebene Zeilenbereich, oder kann aus den ersten 8 Zeichen einer SAM-Datei kein Schlüssel gebildet werden, wird die Zeile ignoriert. Ist die errechnete Zeilennummer größer als der angegebene Zeilenbereich, werden diese und die nachfolgenden Zeilen ignoriert.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @INPUT der Coded Character Set Name (CCSN) der Datei berücksichtigt.

Die @INPUT-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei gleich dem im EDT aktuell eingestelltem ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

Beispiel

```
6.     @PRINT
1.0000 @DELETE
2.0000 ICH BIN ZEILE 1
3.0000 ICH BIN DIE ZWEITE ZEILE
4.0000 @PRINT 1
5.0000 @PRINT 2
6.     @WRITE 'SAM-INP' KEY ------------------------------------------- (01)
6.     @SAVE 'ISAM-INP' ----------------------------------------------- (02)
6.     @INPUT 'SAM-INP' & --------------------------------------------- (03)
1.0000 ICH BIN ZEILE 1
2.0000 ICH BIN DIE ZWEITE ZEILE
3.     @INPUT 'ISAM-INP' 1-3,5,4 -------------------------------------- (04)
2.0000 ICH BIN DIE ZWEITE ZEILE
1.0000 ICH BIN ZEILE 1
3.
```

(01) Der Inhalt der Arbeitsdatei wird als SAM-Datei geschrieben, wobei vor jede Zeile ein Schlüssel zu legen ist, der aus der jeweiligen Zeilennummer errechnet wird.

(02) Der Inhalt der Arbeitsdatei wird noch einmal geschrieben, diesmal aber als ISAM-Datei.

(03) Die komplette Datei SAM-INP wird eingelesen und ausgeführt. Da diese Datei mit @WRITE unter Benutzung von KEY erstellt wurde, ist entweder KEY oder range* (hier: &) anzugeben. Anderenfalls erfolgt keine Umrechnung der gespeicherten Schlüssel in die Zeilennummern.

(04) Die Zeilen 1-3, 5, 4 der Datei ISAM-INP sollen in dieser Reihenfolge eingelesen und ausgeführt werden.

@INPUT (Format 2) Starten einer @INPUT-Prozedur aus einer Bibliothek oder Datei

Mit diesem Format wird eine @INPUT-Prozedur aus einer Bibliothek bzw. einer Datei gestartet. Ein Bibliothekselement oder eine Datei wird in den Benutzeradreßraum eingelesen und die eingelesenen Anweisungen und Textzeilen werden sofort abgearbeitet.

Operation	Operanden	F-Modus / L-Modus
@**I**NPUT	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2 } [**P**RINT]	

LIBRARY = path1 ([E[LEMENT]=]elemname [(vers)][,elemtyp])
: Name des Elements mit Angabe des Bibliotheknamens.

ELEMENT = elemname [(vers)][,elemtyp]
: Namen des Elements ohne Angabe des Bibliotheknamens. Voraussetzung ist die Voreinstellung des Bibliotheknamens mit @PAR.

path1
: Name der Bibliothek. path1 kann auch über Zeichenfolgevariable angegeben werden.
Wird path1 nicht angegeben, wird die mit @PAR LIBRARY voreingestellte Bibliothek verwendet.

elemname
: Name des Elements. elemname kann auch über Zeichenfolgevariable angegeben werden.

vers
: Versionsbezeichnung des gewünschten Elements (siehe Handbuch „LMS“ [13]). Wird vers nicht angegeben, oder *STD, wird die höchste vorhandene Version des Elementes gewählt.

elemtyp
: Typ des Elements. elemtyp kann auch über Zeichenfolgevariable angegeben werden.
Zulässige Typangaben: S, M, P, J, D, X, *STD und freie Typnamen mit entsprechendem Basistyp. Falls nicht angegeben, wird der in @PAR ELEMENT-TYPE voreingestellte Wert verwendet.

Wird ein freier Typnamen verwendet, so liegt es in der Verantwortung des Benutzers, daß der zugehörige Basistyp einem zulässigen Typ S, M, P, J, D oder X entspricht.

Typ	Elementinhalt
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats

*STD
Typ S ist die Voreinstellung nach Aufruf des EDT. Mit @PAR kann eine andere zulässige Typangabe als Voreinstellung festgelegt werden.

FILE = path2 Einlesen einer BS2000-Datei.

path2 Name der Datei, die als @INPUT-Prozedur eingelesen werden soll. path2 kann auch über Zeichenfolgevariable angegeben werden.

PRINT Ist PRINT angegeben, wird jede gelesene Zeile am Bildschirm ausgegeben.

@INPUT darf weder in @INPUT- noch in @DO-Prozeduren abgesetzt werden.

Die Abarbeitung einer @INPUT-Prozedur wird abgebrochen, wenn
– @RETURN abgearbeitet wurde.
– in einer @IF-Anweisung mit Operand RETURN der Vergleich zutrifft.
– in einer Anweisung ein DVS-Fehler aufgetreten ist.

Berechnung der Zeilennummern

Die Datensätze werden beim Einfügen nach drei Methoden numeriert:

1. Standardnumerierung mit Standardschrittweite 1.0000 (z.B. 21.0000, 22.0000, 23.0000 ... 99.0000) oder
2. Numerierung mit festgelegten Schrittweiten gemäß @PAR INCREMENT oder
3. Automatische Numerierung und Umnumerierung,
   wenn die Schrittweite zu groß gewählt wurde, um die einzufügenden Datensätze aufnehmen zu können. Der EDT wählt dann eine Schrittweite, die um Faktor 10 kleiner ist als die Standardschrittweite (1.) bzw. festgelegte Schrittweite (2.). Mit der kleineren Schrittweite wird versucht, die einzufügenden Sätze zu numerieren.
   Dieser Vorgang wird solange wiederholt, bis die kopierten Sätze erfolgreich eingelesen werden können oder der EDT die minimale Schrittweite von 0.01 gewählt hat.

Umnumerierung bei @PAR RENUMBER=ON:
Wenn mit der minimalen Schrittweite 0.01 die kopierten Datensätze nicht eingefügt werden können, numeriert der EDT automatisch die Zeilennummern der hinter dem Zielort bereits bestehenden Sätze mit der Schrittweite 0.01 um.

Kann nicht genügend Platz gefunden werden, wird kein Satz eingefügt und eine Fehlermeldung ausgegeben. Die Zeilennummer des ersten Datensatzes beim Kopieren in eine leere Datei errechnet der EDT aus 0 + Standardschrittweite bzw. 0 + festgelegte Schrittweite (@PAR INCREMENT).

– Bei @PAR INCREMENT mit einer Schrittweite < 0.01 ist zu beachten, daß Zeilennummern von eingelesenen, kopierten oder eingefügten Zeilen im F-Modus nicht vollständig ausgegeben werden (6-stellige Zeilennummernanzeige).
Werden diese unvollständig ausgegebenen Zeilennummern in @INPUT verwendet, kann dies zu unvorhersehbaren Ergebnissen führen.

– Wird die Anweisung im L-Modus eingegeben und wird eine Zeile angelegt, deren Nummer größer als die bisher höchste Zeilennummer ist, so wird die aktuelle Zeilennummer verändert.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @INPUT der Coded Character Set Name (CCSN) der Datei (Bibliothekselement) berücksichtigt.

Die @INPUT-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei (Bibliothekselement) gleich dem im EDT aktuell eingestelltem ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

@INPUT (Format 3) Festlegen des Eingabemodus des EDT

Mit @INPUT, Format 3 bestimmt der Benutzer, daß der EDT Texteingaben interpretieren soll:

– als Folge von abdruckbaren Zeichen.
– als Folge von Hexadezimalzeichen (EBCDIC oder ISO).
– als Folge von Binärzeichen.

Operation	Operanden	L-Modus
@**I**NPUT	{[**C**HAR] / **H**EX|**X** [**I**SO] / **B**INARY}	

CHAR Der EDT interpretiert alle Texteingaben als Folge von abdruckbaren Zeichen.

HEX,X Der EDT interpretiert alle Texteingaben als Folge von Hexadezimalzeichen. Wird eine ungerade Anzahl von hexadezimalen Zeichen eingegeben, wird links eine Null hinzugefügt.
Es können nur Zeilen mit maximal 128 Zeichen eingegeben werden, da die einzugebende Zeile maximal 256 Zeichen lang sein kann.

ISO Die Codierung der Hexadezimaleingabe wird in ISO-Verschlüsselung (ASCII) erwartet. Die Daten selbst werden dann in EBCDIC-Verschlüsselung in die Arbeitsdatei aufgenommen.
Die Umwandlungstabelle entspricht der Zuordnung von Coded Character Set EDF03IRV zu ISO646 internationaler 7-Bit-Code (bzw. EDF041 zu ISO8859-1).
Ist ISO nicht abgegeben, wird die Eingabe in EBCDIC erwartet.

BINARY Der EDT interpretiert alle Texteingaben als Folge von Binärzeichen.
Ist die Anzahl der eingegebenen Zeichen nicht ein Vielfaches von 8, wird links die entsprechende Zahl von Nullen hinzugefügt.
Es können nur Zeilen mit maximal 32 Zeichen eingegeben werden, da die einzugebende Zeile maximal 256 Zeichen lang sein kann.

Nach dem Aufruf des EDT ist CHAR voreingestellt.

i Anweisungen müssen stets als Folge von abdruckbaren Zeichen eingegeben werden.

@LIMITS Zeilennummern und Anzahl der Zeilen ausgeben

Mit @LIMITS gibt der EDT für die aktuelle Arbeitsdatei aus:

– die niedrigste vergebene Zeilennummer
– die höchste vergebene Zeilennummer
– die Anzahl der Zeilen

Die Ausgabe erfolgt auf SYSOUT.

Operation	Operanden	F-Modus / L-Modus
@**LIM**ITS		

Beispiel

```
   4.     @PRINT
   1.0000 A
   2.0000 B
   3.0000 C
   4.     @LIMITS
   1.0000 TO    3.0000          3 LINES ---------------------------------- (01)

   4.     @COPY 1-3 TO 99.01 ------------------------------------------- (02)

 100.03   @LIMITS
   1.0000 TO   99.0300          6 LINES ---------------------------------- (03)

 100.03
```

(01) Die niedrigste und höchste vergebene Zeilennummer und die Zeilenanzahl werden ausgegeben.

(02) Die Zeilen 1-3 sollen nach 99.01, 99.02 und 99.03 kopiert werden.

(03) Nun sind die niedrigste bzw. höchste vergebene Zeilennummern 1.0000 bzw. 99.0300. Die Zeilenanzahl beträgt 6.

@LIST Ausdrucken von Arbeitsdateiinhalten

Mit @LIST können beliebige Teile einer Arbeitsdatei über den Drucker ausgegeben werden.

Operation	Operanden	F-Modus / L-Modus
@**L**IST	[rng [:domain] [**X**] [**N**] [**C** [int]|**P** int] [**I**] [**S**]] [,...]	

rng — Zeilenbereich, bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

Fehlt rng, wird die gesamte Datei ausgegeben. rng muß angegeben werden, wenn X, N, C int, P int, I oder S verwendet werden.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte der Rest der Zeile ausgegeben.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile ausgegeben.

X — Die Zeilen werden hexadezimal ausgedruckt. Hierbei werden die Zeilen nur bis zu maximal 128 Zeichen ausgedruckt.

N — Die Zeilennummern werden beim Ausdrucken unterdrückt.

C — Das erste Zeichen steuert das Ausdrucken, wird selbst aber nicht ausgedruckt.

Inhalt des ersten Zeichens	Steuerfunktion
X’C1’	Seitenvorschub vor dem Drucken
X’40’ oder X’00’	Einfacher Zeilenvorschub vor dem Drucken
X’41’ oder X’01’	Doppelter Zeilenvorschub vor dem Drucken
X’42’ oder X’02’	3facher Zeilenvorschub vor dem Drucken
X’4F’ oder X’0F’	16facher Zeilenvorschub vor dem Drucken

P — Steuert den Seitenvorschub

int — Zahl zwischen 0 und 256

C int — Wird eine Zahl zwischen 1 und 256 angegeben, wird vom EDT nach dem Ausdrucken von int Zeilen ein Seitenvorschub gemacht. Wird für int die Zahl 0 angegeben, wird keine Prüfung vorgenommen. Wird int ohne C angegeben, wird nach genau int Zeilen ein Seitenvorschub gemacht, bzw. bei int=0 wird kein Seitenvorschub gemacht.

P int — Es wird nach genau int Zeilen ein Seitenvorschub gemacht. Wird für int die Zahl 0 angegeben, wird kein Seitenvorschub gemacht.
Bei Verwendung von P muß rng angegeben werden.

I — Mit dem Ausdrucken wird sofort begonnen.
I ist nur im Dialog erlaubt.

Wird I nicht angegeben, so findet die Ausgabe auf SYSLST statt, und falls SYSLST keiner Datei zugeordnet ist, wird mit dem Ausdrucken erst nach LOGOFF begonnen.

S — Unterdrückt die Leerzeilen, die üblicherweise der ersten ausgedruckten Zeile vorausgehen.

Beispiel

```
   6.     @PRINT
   1.0000 MIT DEM @LIST-KOMMANDO
   2.0000 WIRD DER INHALT
   3.0000 EINER ARBEITSDATEI
   4.0000 IN JEDER GEWUENSCHTEN
   5.0000 FORM ZU PAPIER GEBRACHT.
   6.     @LIST ------------------------------------------------------- (01)
   6.     @LIST 4-5 N ------------------------------------------------- (02)
   6.     @LIST & X --------------------------------------------------- (03)
   6.     @LIST & I --------------------------------------------------- (04)
%  SCP0810 SPOOLOUT OF FILE
':A:$BENUTZER.S.SPS.5660.01.23.89023.115714'ACCEPTED
: TSN: '5666', PNAME: 'NAME'
   6.
```

(01) Der komplette Inhalt der Arbeitsdatei soll ausgedruckt werden. Dies geschieht erst nach LOGOFF.

Druckausgabe

```
1.0000 MIT DEM @LIST-KOMMANDO
2.0000 WIRD DER INHALT
3.0000 EINER ARBEITSDATEI
4.0000 IN JEDER GEWUENSCHTEN
5.0000 FORM ZU PAPIER GEBRACHT.
```

(02) Die Zeilen 4 bis 5 sollen nach LOGOFF ohne Zeilennummer ausgedruckt werden.

Druckausgabe

```
IN JEDER GEWUENSCHTEN
FORM ZU PAPIER GEBRACHT.
```

(03) Alle Zeilen sollen nach LOGOFF in hexadezimaler Form ausgedruckt werden.

Druckausgabe

```
1.0000 D4C9E340C4C5D4407CD3C9E2E360D2D6D4D4C1D5C4D6
2.0000 E6C9D9C440C4C5D940C9D5C8C1D3E3
3.0000 C5C9D5C5D940C1D9C2C5C9E3E2C4C1E3C5C9
4.0000 C9D540D1C5C4C5D940C7C5E6E4C5D5E2C3C8E3C5D5
5.0000 C6D6D9D440E9E440D7C1D7C9C5D940C7C5C2D9C1C3C8E34B
```

(04) Alle Zeilen sollen sofort ausgedruckt werden.

Druckausgabe

```
1.0000 MIT DEM @LIST-KOMMANDO
2.0000 WIRD DER INHALT
3.0000 EINER ARBEITSDATEI
4.0000 IN JEDER GEWUENSCHTEN
5.0000 FORM ZU PAPIER GEBRACHT.
```

@LOAD Programm laden

@LOAD bewirkt, daß

– die EDT-Sitzung beendet wird
– das angegebene Programm geladen wird

Operation	Operanden	F-Modus / L-Modus
@**LO**AD	string	

string Zeichenfolge, die den Namen des Programms angibt, das geladen werden soll.
string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Existieren noch ungesicherte Arbeitsdateien, so werden nach der Meldung: % EDT0900 EDITED FILE(S) NOT SAVED! die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben.

Zusätzlich wird ausgegeben, falls vorhanden, entweder

– ein lokaler @FILE-Eintrag
  – explizit definiert durch @FILE LOCAL, oder
  – implizit definiert durch @READ, @GET, @OPEN (Format 1)
– oder der Bibliotheks- und Elementsname eines mit
  – @OPEN (Format 2) eröffneten Bibliothekselements
– oder der Dateiname einer mit
  – @OPEN (Format 2) eröffneten SAM-oder ISAM-Datei
  – @XOPEN eröffneten POSIX-Datei

Danach folgt die Anfrage an den Benutzer:

% EDT0904 TERMINATE EDT? REPLY (Y=YES, N=NO)

N: Im F-Modus erscheint das Arbeitsfenster wieder. Der Benutzer kann ungesicherte Dateien schließen und zurückschreiben.

Y: Die ungesicherten, virtuellen Dateien gehen verloren. Der EDT wird beendet, das angegebene Programm gestartet.

Wurde eine Datei real mit @OPEN eröffnet, entfällt diese Abfrage. Der EDT schließt die Datei durch ein implizites @CLOSE.

Die Sicherungsabfrage kann unterdrückt werden, indem man vor dem Aufruf des EDT den Auftragsschalter 4 einschaltet.

Beispiel

```
   1.00 @LOAD bewirkt, dass ..................................................
   2.00 -  die EDT-Sitzung beendet, ...........................................
   3.00 -  das angegebene Programm geladen wird. ..............................
   4.00 ......................................................................









load '$lms'.....................................................0001.00:001(0)
```

Der EDT soll beendet und der LMS geladen werden.

```
%  EDT0900 EDITED FILE(S) NOT SAVED!
 LOCAL FILE ( 0) :
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)?y
%  BLS0500 PROGRAM 'LMS', VERSION 'V3.0A' OF 'yy-mm-dd' LOADED.
/resume-program
%  LMS0310 LMS VERSION V03.0A00 LOADED
CTL=(CMD)                                                    PRT=(OUT)
$
```

Da die Arbeitsdatei noch nicht gesichert wurde, fragt der EDT wie bei @HALT nach, ob er tatsächlich beendet werden soll.

Da @LOAD und nicht @EXEC angegeben wurde, wird mit Schrägstrich angezeigt, daß weitere Systemkommandos erwartet werden. Erst mit dem Kommando RESUME-PROGRAM wird der LMS gestartet.

@LOG Protokollsteuerung

@LOG steuert die Protokollierung der Eingaben im Stapelbetrieb und im Dialog.

Die Ausgabe kann erfolgen:

– nach SYSLST (Schnelldrucker)
– nach SYSLSTnn oder einer SYSLSTnn zugewiesenen Datei
– in eine Listenvariable

Operation	Operanden	F-Modus / L-Modus
@LOG	[{ **A**LL / **C**OMMANDS / **N**ONE }] [{ **SYSLST** / **SYSLST** nn / LIST-**VAR**=chars }]	

ALL Alle Eingaben im L-Modus (Text und Anweisungen), die über RDATA oder über die Datensichtstation eingegeben werden, sollen protokolliert werden.
Bei Eingaben im F-Modus-Dialog wird die Eingabe in der Anweisungszeile (bei Anweisungskettung in einzelne Anweisungen getrennt) protokolliert.

COMMANDS Nur Anweisungen sollen protokolliert werden.

NONE Nichts soll protokolliert werden.

Die Voreinstellung beim Aufruf des EDT im Stapelbetrieb ist abhängig von Auftragsschalter 4:
– Bei SETSW ON=4 wird @LOG NONE eingestellt.
– Bei SETSW OFF=4 wird @LOG COMMANDS eingestellt.

Wird EDT nicht im Batch aufgerufen, ist @LOG NONE voreingestellt.

SYSLST Die Protokollierung erfolgt auf SYSLST. Dies ist die Voreinstellung.

SYSLST nn nn = 1,...99
Die Protokollierung erfolgt auf die Datei, die SYSLSTnn zugewiesen ist.

LIST-VAR Die Protokollierung erfolgt in eine Listenvariable.

chars Zeichenfolge, die den Namen einer Listenvariable angibt. Die S-Variable muß vorher definiert werden, z.B. mit DECLARE-VARIABLE chars, MULTI-ELEMENT=LIST. Die einzelnen Elemente der Liste müssen vom Typ ANY oder STRING sein. Die Liste wird durch die einzelnen Protokollzeilen am Ende erweitert.

Ist keiner der Operanden SYSLST, SYSLSTnn oder VAR angegeben, bleibt der Ausgabeort erhalten.

Die Anweisung wird auch im Testmodus ausgeführt.

@LOWER Groß-/Kleinschreibung bei der Bildschirmein-/ausgabe

Mit @LOWER wird festgelegt, ob der EDT eingegebene Kleinbuchstaben in Großbuchstaben umsetzt oder nicht.

@LOWER wirkt global für alle Arbeitsdateien, unabhängig vom Arbeitsfenster, in dem @LOWER eingegeben wurde.

Operation	Operanden	F-Modus / L-Modus
@**LOW**ER	[**O**N] | **OF**F	

ON Der EDT unterscheidet zwischen Groß- und Kleinbuchstaben. Zeichenfolgen werden verarbeitet, wie sie eingegeben werden.

OFF Der EDT codiert eingegebene Kleinbuchstaben a, ..., z in Großbuchstaben A, ..., Z um und gibt diese am Bildschirm aus. Dies gilt auch für Zeichenfolgen, die als Operanden in Anweisungen stehen. Die Umlaute ä, ö und ü werden nicht umcodiert. Sie werden als Kleinbuchstaben in die Arbeitsdatei übernommen.

Im F-Modus werden in der Arbeitsdatei enthaltene Kleinbuchstaben bei der Ausgabe am Bildschirm als Schmierzeichen dargestellt. Im L-Modus werden sie abdruckbar dargestellt.

Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.
Wird @LOWER ohne Operanden eingegeben, wird @LOWER ON eingestellt.
Wird @LOWER innerhalb eines Eingabe-Blocks (siehe @BLOCK) angegeben, wird der Umcodierungsmodus erst nach Abarbeitung aller Anweisungen des Eingabeblocks wirksam. @LOWER sollte daher immer am Ende einer Eingabe stehen.

Ist XHCS im System vorhanden, wird zur Umsetzung von Klein- in Großbuchstaben die zum Coded Character Set (CCS) zugehörige Umsetztabelle verwendet.

Kleinbuchstabenbehandlung bei der Ausführung von EDT-Anweisungen

– Bei @LOWER OFF werden bei Eingaben vom Bildschirm in allen Arbeitsmodi Kleinbuchstaben in Großbuchstaben umgesetzt.
  Bei Eingaben, die von SYSDTA-Dateien, aus @INPUT- bzw. Prozedurdateien oder über die Unterprogrammschnittstelle gelesen werden, erwartet der EDT Anweisungen in Großbuchstaben.
– Bei @LOWER ON werden im F-Modus-Dialog nur die Kleinbuchstaben im Operanden string nicht in Großbuchstaben umgesetzt. in den restlichen Arbeitsmodi (L-Modus, Prozedurbetrieb, ...) werden die Kleinbuchstaben in den Operanden string, text und param nicht umgesetzt.

@MOVE Übertragen von Zeilenbereichen

Mit @MOVE wird eine Zeile oder ein Zeilenbereich in einen anzugebenden Bereich übertragen. Der Sendebereich wird gelöscht. Die Übertragung kann innerhalb der aktuellen Arbeitsdatei oder aus einer beliebigen Arbeitsdatei erfolgen.

Es kann nicht aus einer Arbeitsdatei übertragen werden, die gerade als EDT-Prozedur (siehe @DO) abgearbeitet wird (aktive Arbeitsdatei).

Operation	Operanden	F-Modus / L-Modus
@**M**OVE	rng [(procnr)] [**T**O ln1 [(inc)] [:] [ln2]] [,...]	

Wird aus der aktuellen Arbeitsdatei übertragen, müssen die Operanden TO und ln1 immer angegeben werden. Wird aus einer anderen Arbeitsdatei übertragen, werden bei Weglassen dieses Operanden die Zeilennummern des Sendebereichs beibehalten.

rng — Zeilenbereich, bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

Die symbolischen Zeilennummern beziehen sich auf die aktuelle Arbeitsdatei, d.h. die Werte der symbolischen Zeilennummern entsprechen den Zeilennummern der aktuellen Arbeitsdatei und nicht der Arbeitsdatei, aus der übertragen wird.

procnr — Arbeitsdatei (0-22), aus der übertragen wird. Wird procnr nicht angegeben, wird aus der aktuellen Arbeitsdatei übertragen.

ln1 — Nummer der ersten Zeile des Empfangsbereichs.

Die Nummern der folgenden Zeilen des Empfangsbereichs errechnet der EDT, indem er die jeweilige Zeilennummer um die für den Empfangsbereich geltende Schrittweite erhöht. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
Fehlt inc, wird mit ln1 implizit die Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.

ln1 darf auch durch Zeilennummervariablen oder symbolisch angegeben werden.

inc	aktuelle Schrittweite des Empfangsbereichs. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
:	Dieses Trennzeichen kann entfallen, wenn inc angegeben ist und dadurch ln1 und ln2 eindeutig voneinander getrennt sind.
ln2	Nummer der letzten Zeile des Empfangsbereichs. @MOVE überträgt zeilenweise. Beim Erreichen dieser Obergrenze wird der Übertragungsvorgang beendet, unabhängig davon, ob noch zu übertragende Zeilen vorhanden sind oder nicht. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. ln2 darf auch durch Zeilennummervariablen oder symbolisch angegeben werden. Wird ln2 nicht angegeben, können unbeabsichtigt Zeilen überschrieben werden.

Zuerst wird der ganze Sendebereich übertragen und danach gelöscht.

Im Falle einer Überlappung von Sende- und Empfangsbereich wird Zeile für Zeile übertragen und gelöscht.

Aktuelle Schrittweite und Zeilennummer

Die aktuelle Schrittweite wird von @MOVE nicht verändert. inc bestimmt nur die Schrittweite zwischen den übertragenen Sätzen. Es bezieht sich nicht auf die aktuelle Schrittweite.

Wird die Schrittweite inc zu groß gewählt, können im Empfangsbereich Zeilen überschrieben werden.

Die aktuelle Zeilennummer im L-Modus wird nur dann verändert, wenn eine Zeile angelegt wird, deren Nummer größer als die bisher höchste Zeilennummer ist.

Beispiel

```
   1.00 DIESE ZEILE WIRD NICHT BEWEGT.....................................
   2.00 DIE ZEILE 2 UND DIE ZEILE 3.......................................
   3.00 UND DIE ZEILE 4 WERDEN............................................
   4.00 DES OEFTEREN BEWEGT. .............................................
   5.00 ..................................................................
```

```
set 90:die zeile wird nie ueberschrieben.....................0001.00:001(0)
```

Die Zeile 90 wird neu angelegt.

```
   1.00 DIESE ZEILE WIRD NICHT BEWEGT.....................................
   2.00 DIE ZEILE 2 UND DIE ZEILE 3.......................................
   3.00 UND DIE ZEILE 4 WERDEN............................................
   4.00 DES OEFTEREN BEWEGT. .............................................
  90.00 DIE ZEILE WIRD NIE UEBERSCHRIEBEN.................................
  91.00 .....................................................................




move 2-4 to 20.............................................0001.00:001(0)
```

Die Zeilen 2 bis 4 sollen in den Bereich ab Zeile 20 übertragen werden. Als Schrittweite des Empfangsbereichs ist implizit der Wert 1 angegeben.

```
   1.00 DIESE ZEILE WIRD NICHT BEWEGT.....................................
  20.00 DIE ZEILE 2 UND DIE ZEILE 3.......................................
  21.00 UND DIE ZEILE 4 WERDEN............................................
  22.00 DES OEFTEREN BEWEGT. .............................................
  90.00 DIE ZEILE WIRD NIE UEBERSCHRIEBEN.................................
  91.00 .....................................................................




move 20-22 to 100 (5).....................................0001.00:001(0)
```

Die Zeilen 20, 21 und 22 wurden mit der impliziten Schrittweite 1 neu angelegt und die Zeilen 2, 3 und 4 gelöscht.

Die Zeilen 20-22 sollen nach 100, 105 und 110 übertragen werden.

```
   1.00 DIESE ZEILE WIRD NICHT BEWEGT.....................................
  90.00 DIE ZEILE WIRD NIE UEBERSCHRIEBEN.................................
 100.00 DIE ZEILE 2 UND DIE ZEILE 3.......................................
 105.00 UND DIE ZEILE 4 WERDEN............................................
 110.00 DES OEFTEREN BEWEGT. .............................................
 111.00 .....................................................................




move 100-.$ to 82(5) : 89.................................0001.00:001(0)
```

Der Zeilenbereich ab Zeile 100 bis Arbeitsdateiende (100-.$) soll mit der expliziten Schrittweite 5 in den Zeilenbereich ab Zeile 82 übertragen werden. Das Angeben der Obergrenze 89 bewirkt, daß die Zeile 90 nicht überschrieben wird.

```
  1.00 DIESE ZEILE WIRD NICHT BEWEGT.....................................
 82.00 DIE ZEILE 2 UND DIE ZEILE 3.......................................
 87.00 UND DIE ZEILE 4 WERDEN............................................
 90.00 DIE ZEILE WIRD NIE UEBERSCHRIEBEN..................................
110.00 DES OEFTEREN BEWEGT. ..............................................
111.00 ..................................................................
```

Die Zeile 110 wurde nicht übertragen, da sonst die angegebene Obergrenze überschritten worden wäre.

@NOTE Kommentierung von EDT-Prozeduren

Mit @NOTE kann man EDT-Prozeduren kommentieren. @NOTE verursacht bei der Ausführung keine Aktion.

Operation	Operanden	L-Modus / @PROC
@**N**OTE	[comment]	

comment Kommentar; Beliebiger Text.

@ON Dateibearbeitung mit Suchbegriff

@ON überprüft einen anzugebenden Zeilenbereich auf das Vorhandensein eines Suchbegriffs. 11 Formate von @ON stehen zur Verfügung, um die Datenzeilen zu bearbeiten, in denen der Suchbegriff vorhanden ist. Wird der Suchbegriff gefunden, wird eine der folgenden Aktionen ausgeführt:

Ausgeben und Markieren

Format 1: Ausgeben der Zeile, die den Suchbegriff enthält, auf den Bildschirm (im Dialog) oder den Drucker (im Stapelbetrieb). (Seite 330ff.)

Format 2: Ausgeben der Nummer der Spalte, in der der Suchbegriff beginnt, oder der Zeilenlängen im angegebenen Zeilenbereich. Die Ausgabe erfolgt auf den Bildschirm. (Seite 334ff.)

Format 3: Überprüfen, ob ein Suchbegriff vorhanden ist. Die Nummer der Zeile mit Suchbegriff (erster Treffer) wird festgehalten. (Seite 338ff.)

Format 4: Markieren der Zeile, die den Suchbegriff enthält, mit einer Satzmarkierung (Vorbereitung für Format 5). (Seite 340ff.)

Kopieren mit und ohne Markierung

Format 5: Kopieren der Zeilen mit der angegebenen Satzmarkierung. (Seite 343ff.)

Format 6: Kopieren der Zeilen, die den Suchbegriff enthalten. (Seite 346ff.)

Ersetzen und Einfügen von Zeichenfolgen

Format 7: Ersetzen der mit dem Suchbegriff bestimmten Zeichenfolge durch eine andere Zeichenfolge. (Seite 350ff.)

Format 8: Ersetzen oder Einfügen einer Zeichenfolge vor oder nach dem Suchbegriff. (Seite 353ff.)

Löschen von Zeichenfolgen

Format 9: Löschen des Suchbegriffs aus dem Text. (Seite 357ff.)

Format 10: Löschen des Zeileninhaltes vor oder nach dem Suchbegriff. (Seite 360ff.)

Format 11: Löschen der Zeile, in der der Suchbegriff gefunden wird. (Seite 362ff.)

Angabe des Suchbegriffs in @ON

Die Angabe des Suchbegriffs search in @ON ist möglich durch:

– direkte Angabe zwischen Hochkommas oder Anführungszeichen

– indirekte Angabe mit Hilfe einer Zeilennummer-, Zeilennummervariablen oder Zeichenfolgevariablen.

Der einfachste Suchbegriff ist die konstante Zeichenfolge (z.B. 'ABC'). Er wird im Suchobjekt durch eine Teilkette mit demselben Textwert befriedigt.

Verwendung von Jokerzeichen im Suchbegriff

Neben konstanten Zeichen können auch variable, sogenannte Jokerzeichen, angegeben werden. Es gibt zwei Jokerzeichen:

asterisk (Standardwert *) ersetzt eine beliebig lange, auch leere Zeichenfolge. Es wird durch die kürzest mögliche Teilkette der überprüften Zeile befriedigt. Wird es mehrmals nebeneinander angegeben, wird es wie ein einziger asterisk ausgeführt; z.B.: 'ABC**F' ist gleichwertig mit 'ABC*F'.

slash (Standardwert /) ersetzt genau ein Zeichen.

Ist das Schlüsselwort PATTERN angegeben, werden die Jokerzeichen als variable Zeichen interpretiert, und es wird eine Mustererkennung ('Pattern Matching') durchgeführt (siehe auch Beispiel zu @ON, Format 6).

Fehlt das Schlüsselwort PATTERN, werden die Jokerzeichen als einfache, konstante Zeichen behandelt.

Beispiel

on & print 'AB*C'	zeigt alle Zeilen, die genau die Zeichenfolge AB*C enthalten.
on & print pattern 'AB*C'	zeigt die Zeilen, die die Zeichenfolgen ABC, ABXC, ABCDEFG, ABXXXXXXC etc. enthalten.

In jedem Suchbegriff dürfen mehrere Jokerzeichen verwendet werden. Ein nur aus Jokerzeichen bestehender Suchbegriff ist ebenfalls zulässig. Mit @SYMBOLS können die Jokerzeichen auch umdefiniert werden.

Negatives Suchen

Durch die Angabe des Schlüsselwortes NOT werden die Sätze angesprochen, die den Suchbegriff nicht enthalten (siehe auch Beispiel zu @ON, Format 1).

Indirekte Angabe des Suchbegriffs

– Der Suchbegriff steht in einer Zeile, deren Zeilennummer angegeben werden muß.

– Der Suchbegriff steht in einer Zeile, deren Zeilennummer in einer Zeilenummernvariable (#L0 bis #L20) steht. Die Zeilennummervariable ist anschließend in der @ON-Anweisung anzugeben.

– Der Suchbegriff (z.B. 'ABC') wird einer Stringvariablen (#S0 - #S20) zugewiesen (siehe @SET). Die Zeichenfolgevariable ist anschließend in der @ON-Anweisung anzugeben.

```
@SET #S0 = 'AB*C//D'
@ON  & PRINT PATTERN #S0
```

Der Suchbegriff wird bei indirekter Angabe so behandelt, als wäre er in Hochkommas eingeschlossen. Die Angabe in Anführungszeichen ist nicht möglich.

Bedeutung der Begrenzer eines Suchbegriffs

Linke Begrenzung des Suchbegriffs durch			
Hochkomma		Anführungszeichen	
Rechte Begrenzung des Suchbegriffs durch		Rechte Begrenzung des Suchbegriffs durch	
Hochkomma	Anführungszeichen	Hochkomma	Anführungszeichen
Der EDT stellt das Vorhandensein des Suchbegriffs fest, wenn – der Suchbegrif vorhanden ist.	Der EDT stellt das Vorhandensein des Suchbegriffs fest, wenn – der Suchbegriff vorhanden ist und – auf den Suchbegriff ein Textbegrenzerzeichen folgt oder der Suchbegriff am Zeilenende steht.	Der EDT stellt das Vorhandensein des Suchbegriffs fest, wenn – der Suchbegriff vorhanden ist und – vor dem Suchbegriff ein Textbegrenzerzeichen steht oder der Suchbegriff am Zeilenanfang beginnt.	Der EDT stellt das Vorhandensein des Suchbegriffs fest, wenn – der Suchbegriff vorhanden ist und – vor dem Suchbegriff ein Textbegrenzerzeichen steht oder der Suchbegriff am Zeilenanfang beginnt und – auf den Suchbegriff ein Textbegrenzerzeichen folgt oder der Suchbegriff am Zeilenende steht.
Beispiel: 1. ABCD 2. A,BCD 3. ABC,D 4. A,BC,D Mit @ON &... ' BC' ... werden Treffer in allen 4 Zeilen festgestellt.	Beispiel: 1. ABCD 2. A,BCD 3. ABC,D 4. A,BC,D Mit @ON &... ' BC"... wird in der 3. und 4. Zeile 1 Treffer festgestellt.	Beispiel: 1. ABCD 2. A,BCD 3. ABC,D 4. A,BC,D Mit @ON &... "BC' ... wird nur in der 2. und 4. Zeile 1 Treffer festgestellt.	Beispiel: 1. ABCD 2. A,BCD 3. ABC,D 4. A,BC,D Mit @ON &... "BC"... wird nur in der 4. Zeile 1 Treffer festgestellt.

Wurden mit @QUOTE an Stelle der Hochkommas andere Zeichen vereinbart, gelten die oben angegebenen Regeln für diese Zeichen.

Zu Beginn einer EDT-Sitzung stehen folgende Textbegrenzerzeichen zur Verfügung:

Leerzeichen (X'40') sowie +.!*();–/,?:'="

Die Menge der Textbegrenzerzeichen kann mit @DELIMIT verändert werden.

Bedeutung der Begrenzer bei einem Suchbegriff mit Jokerzeichen

Der Suchbegriff kann eingeschlossen werden zwischen:

Hochkommas	Hochkomma und Anführungszeichen	Anführungszeichen
@ON & P PATTERN 'ABC*'	@ON & P PATTERN 'ABC*" @ON & P PATTERN "ABC*'	@ON & P PATTERN "ABC*"

Steht das Jokerzeichen asterisk im Suchbegriff neben einem Anführungszeichen, dann reicht der Trefferstring bis zum nächsten Textbegrenzerzeichen. Gibt es kein Textbegrenzerzeichen, dann enthält der Trefferstring den Rest der Zeile.

Steht das Jokerzeichen asterisk neben einem Hochkomma, dann enthält der Trefferstring die kürzestmögliche Zeichenfolge.

Beispiel

Zeile enthält xxx_abcd_yyy

Suchbegriff	Trefferstring
'abc*"	abcd (weil _ der nächste Textbegrenzer ist
'abc*'	abc (weil die kürzestmögliche Teilkette genommen wird)

Soll ein Suchbegriff die Textbegrenzerzeichen selbst enthalten, so muß für ' ein '' und für " ein "" eingegeben werden.

Beispiel

'Dies ist eine "" beliebige"" Zeichenfolge.'
(Es wird gesucht nach der Zeichenfolge Dies ist eine "beliebige" Zeichenfolge.)

Arbeitsweise des EDT mit Suchbegriffen

Die Reihenfolge, in der der EDT die Zeilen nach einem Suchbegriff durchsucht und die Art der Suche, sind abhängig von folgenden Operanden:

range	Zeilenbereich (mehrere Teilbereiche), der durchsucht wird
domain	Spaltenbereich, in dem gesucht wird
F	der EDT sucht nur den ersten Treffer
ALL	der EDT sucht alle Treffer in einer Zeile
int	der EDT sucht nach dem int-ten Treffer in einer Zeile
R	der EDT durchsucht die Zeilen von rechts nach links

Die Zeilen werden standardmäßig von links nach rechts durchsucht. Der EDT überprüft die Zeilen in der Reihenfolge, in der sie mit range angegeben sind. Die folgenden Übersichten erläutern die Aktionen des EDT in Abhängigkeit von F, range, int und ALL.

<table>
<tr><td colspan="3">Ist der Operand F angegeben?</td></tr>
<tr><td colspan="2">Ja</td><td>Nein</td></tr>
<tr><td colspan="2">Sind im Operanden range mehrere durch Komma voneinander getrennte Zeilenbereiche (bzw. Zeilen) angegeben?</td><td rowspan="2">Jede Zeile des angegebenen Bereichs bzw. der angegebenen Bereiche wird durchsucht.</td></tr>
<tr><td>Ja</td><td>Nein</td></tr>
<tr><td>Wenn der EDT den Suchbegriff findet, bricht er das Suchen in dem momentan überprüften Zeilenbereich ab. Die restlichen Zeilen dieses Bereichs werden nicht mehr durchsucht.
Der EDT setzt die Suche im nächsten angegebenen Zeilenbereich fort.</td><td>Wenn der EDT den Suchbegriff findet, bricht er die Suche ab. Die restlichen Zeilen werden nicht mehrdurchsucht.</td><td></td></tr>
</table>

<table>
<tr><td colspan="4">Ist der Operand int angegeben?</td></tr>
<tr><td colspan="2">Ja</td><td colspan="2">Nein</td></tr>
<tr><td colspan="2" rowspan="2">Erst wenn der EDT den Suchbegriff in der überprüften Zeile zum int-ten Mal findet, wertet er das als Treffer.

Beispiel:
1. AAAAA
Mit @ON 1 ' AA' ,3 wird in den Spalten 3 bis 4 ein Treffer festgestellt.</td><td colspan="2">Ist der Operand ALL angegeben?</td></tr>
<tr><td>Ja</td><td>Nein</td></tr>
<tr><td colspan="2">Ist der Operand ALL angegeben?</td><td>Wenn der EDT den Suchbegriff findet, führt er die bei @ON geforderten Aktionen aus.
Anschließend setzt der EDT die Überprüfung der Zeile ab der Spalte fort, die ursprünglich auf den Suchbegriff folgte.</td><td>Sobald der EDT den Suchbegriff zum ersten Mal findet, bricht er die Überprüfung der Zeile ab.</td></tr>
<tr><td>Ja</td><td>Nein</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Wenn der EDT einen Treffer feststellt, führt er die bei @ON geforderten Aktionen aus.
Anschließend setzt der EDT die Überprüfung der Zeile ab der Spalte fort, die ursprünglich auf den Suchbegriff folgte.
Jede weitere Übereinstimmung mit dem Suchbegriff, die der EDT in der Zeile findet, wertet er als Treffer.</td><td>Sobald der EDT den ersten Treffer festgestellt hat, bricht er die Überprüfung der Zeile ab.</td></tr>
</table>

Festhalten eines Treffers

Der EDT hält fest, ob ein Treffer festgestellt wurde oder nicht:

Abfrage des Treffers mit @IF

– Mit @IF (siehe @IF, Format 3) kann abgefragt werden, ob bei der Ausführung des letzten @ON ein Treffer festgestellt wurde.

– Die Meldung NO MATCH IN RANGE kann nur abgefragt werden, wenn die Prozedur mit @DO...PRINT aufgerufen wurde. Der EDT-Fehlerschalter ist nur gesetzt, wenn die Meldung tatsächlich am Bildschirm ausgegeben wurde.

Festhalten der Trefferzeile

– Die Nummer der Zeile, in der der EDT den 1. Treffer feststellt, wird in der Zeilennummervariablen #L0 und unter dem Zeilennummern-Symbol ? festgehalten. Wird der Treffer in einer Zeichenfolgevariablen festgestellt, bleiben die Werte von #L0 und ? unverändert.
– Die Nummer der Spalte, in der beim ersten festgestellten Treffer der Suchbegriff beginnt, wird in der Ganzzahlvariablen #I0 gespeichert; die Nummer der Spalte, in der sie endet, in der Ganzzahlvariablen #I1. Das gilt auch für einen in einer Zeichenfolgevariablen festgestellten Treffer.

Beispiel

Zeile enthält XXX_ABCD_YYY

Suchbegriff	**Trefferstring**	**Inhalt von I0 und #I1**
’ABC*“	ABCD	#I0 = 5; #I1 = 8
’ABC*’	ABC	#I0 = 5; #I1 = 7
“*BCD’	ABCD	#I0 = 5; #I1 = 8
’*BCD’	BCD	#I0 = 6; #I1 = 8

Mit @PRINT #L0:#I0-#I1: kann man im L-Modus den Treffer auf dem Bildschirm ausgeben lassen. Wird kein Treffer festgestellt, bleiben die Werte von ?, #L0, #I0 und #I1 unverändert.

Festhalten eines Treffers beim negativen Suchen

Beim ersten Satz, in dem der Suchbegriff nicht vorkommt, wird in der Ganzzahlvariablen #I0 die Anfangsposition und in der Ganzzahlvariablen #I1 die Endposition des überprüften Spaltenbereichs gespeichert.

Wenn die Endposition des Spaltenbereichs größer als die Satzlänge ist, wird in #I1 die Satzlänge gespeichert.

@ON (Format 1) Ausgeben der Zeileninhalte mit dem Suchbegriff

Dieses Format von @ON bewirkt, daß der EDT den Inhalt jeder Zeile ausgibt, in der der Suchbegriff festgestellt wird. Im Dialog erfolgt die Ausgabe am Bildschirm (SYSOUT), bei Stapelverarbeitung auf den Drucker (SYSLST).

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **P**RINT [{**A**LL / **F**}] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**S**] [**N**]	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht. Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL — Kann zwar angegeben werden, ist aber wirkungslos.

F — Die Überprüfung jedes angegebenen Zeilenbereichs wird jeweils nach dem 1. Treffer abgebrochen.

R — Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.
Wird R angegeben, so muß PRINT mindestens mit PR abgekürzt werden.

NOT — Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).

PATTERN — Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.

search — Suchbegriff. Dieser kann angegeben werden:
- direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder
- indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.

int — Erst das int-te Auftreten des Suchbegriffs in einer Zeile ist als Treffer zu werten.

S — Die Leerzeile, die der ersten auszugebenden Zeile vorausgeht, wird unterdrückt.

N — Die Trefferzeilen werden ohne zugehörige Zeilennummer ausgegeben.

Beispiel

```
   1.00 ALLE ZEILEN, IN.....................................................
   2.00 DENEN EIN TREFFER VORKOMMT,..........................................
   3.00 SOLLEN AUSGEGEBEN WERDEN. ..........................................
   4.00 KOMMT KEIN..........................................................
   5.00 TREFFER VOR, SO WIRD NICHTS.........................................
   6.00 AUSGEGEBEN. ........................................................
   7.00 .....................................................................




on & print f 'treffer'.....................................0001.00:001(1)
```

Die erste Zeile, die die Zeichenfolge TREFFER enthält, soll ausgegeben werden.

```
   2.0000 DENEN EIN TREFFER VORKOMMT,
PLEASE ACKNOWLEDGE
```

```
   1.00 ALLE ZEILEN, IN.....................................................
   2.00 DENEN EIN TREFFER VORKOMMT,..........................................
   3.00 SOLLEN AUSGEGEBEN WERDEN. ...........................................
   4.00 KOMMT KEIN..... .....................................................
   5.00 TREFFER VOR, SO WIRD NICHTS..........................................
   6.00 AUSGEGEBEN. ..........................................................
   7.00 .....................................................................




on & print 'treffer'.....................................0001.00:001(1)
```

Alle Zeilen sollen ausgegeben werden, in denen die Zeichenfolge TREFFER vorkommt.

```
    2.0000 DENEN EIN TREFFER VORKOMMT,
    5.0000 TREFFER VOR, SO WIRD NICHTS
 PLEASE ACKNOWLEDGE









create 100: #l0 :#i0-#i1:.....................................0001.00:001(1)
```

Bei mehreren Treffern sind die Inhalte der Zeilennummervariablen #L0 und der Ganzzahlvariablen #I0 und #I1 für den ersten gefundenen Treffer gültig, d.h. TREFFER in Zeile 2.

```
   1.00 ALLE ZEILEN, IN.....................................................
   2.00 DENEN EIN TREFFER VORKOMMT,..........................................
   3.00 SOLLEN AUSGEGEBEN WERDEN.............................................
   4.00 KOMMT KEIN..........................................................
   5.00 TREFFER VOR, SO WIRD NICHTS..........................................
   6.00 AUSGEGEBEN. ..........................................................
 100.00 TREFFER...............................................................
 101.00 .....................................................................




on &:2 print 'treffer'.....................................0001.00:001(1)
```

Alle Zeilen sollen ausgegeben werden, die ab Spalte 2 die Zeichenfolge TREFFER beinhalten.

```
    2.0000 DENEN EIN TREFFER VORKOMMT,
 PLEASE ACKNOWLEDGE
```

```
   1.00 ALLE ZEILEN, IN.....................................................
   2.00 DENEN EIN TREFFER VORKOMMT,..........................................
   3.00 SOLLEN AUSGEGEBEN WERDEN............................................
   4.00 KOMMT KEIN..........................................................
   5.00 TREFFER VOR, SO WIRD NICHTS..........................................
   6.00 AUSGEGEBEN. .........................................................
 100.00 TREFFER..............................................................
 101.00 .....................................................................




on & print 'en',3.............................................0001.00:001(1)
```

Die Zeilen sollen ausgegeben werden, die mindestens dreimal die Zeichenfolge EN enthalten.

```
    3.0000 SOLLEN AUSGEGEBEN WERDEN.
 PLEASE ACKNOWLEDGE
```

```
   1.00 ALLE ZEILEN, IN.....................................................
   2.00 DENEN EIN TREFFER VORKOMMT,..........................................
   3.00 SOLLEN AUSGEGEBEN WERDEN............................................
   4.00 KOMMT KEIN..........................................................
   5.00 TREFFER VOR, SO WIRD NICHTS..........................................
   6.00 AUSGEGEBEN. .........................................................
 100.00 TREFFER..............................................................
 101.00 .....................................................................




on & print not 'treffer'.....................................0001.00:001(1)
```

Es sollen alle Zeilen ausgegeben werden, die nicht die Zeichenfolge TREFFER enthalten.

```
    1.0000 ALLE ZEILEN, IN
    3.0000 SOLLEN AUSGEGEBEN WERDEN.
    4.0000 KOMMT KEIN
    6.0000 AUSGEGEBEN.
 PLEASE ACKNOWLEDGE
```

@ON (Format 2) Ausgeben der Anfangsspalte eines Suchbegriffs

Dieses Format von @ON bewirkt, daß der EDT auf dem Bildschirm die Zeilennummern und die Nummern der Spalten ausgibt, in denen die Suchbegriffe beginnen. Wird search nicht angegeben, gibt der EDT die Zeilennummern und die Länge jeder Zeile des angegebenen Zeilenbereichs aus.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **CO**LUMN [{**A**LL / **F**}] [**R**] [**P**ATTERN] [search [,int]]	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen
- (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL — Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort.

F — Es wird nur der 1. Treffer jedes angegebenen Zeilenbereichs angezeigt. Wird weder ALL noch F angegeben, so werden alle 1. Treffer jeder Zeile angezeigt.

R — Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.

PATTERN — Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.

search — Suchbegriff. Dieser kann angegeben werden:
- direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder
- indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.

int — Erst das int-te Auftreten des Suchbegriffs in einer Zeile ist als Treffer zu werten.

Beispiel

```
   1.00 WIE LANG IST ZEILE 1 ?.....................................................
   2.00 UND ZEILE 2 ?..............................................................
   3.00 WER WEISS DIE LAENGE VON ZEILE 3 ?..........................................
   4.00 ...........................................................................









on & column.....................................................0001.00:001(1)
```

Die Länge aller Zeilen soll ausgegeben werden.

```
    1.0000 022
    2.0000 013
    3.0000 034
 PLEASE ACKNOWLEDGE
```

```
   1.00 WIE LANG IST ZEILE 1 ?.....................................................
   2.00 UND ZEILE 2 ?..............................................................
   3.00 WER WEISS DIE LAENGE VON ZEILE 3 ?..........................................
   4.00 ...........................................................................









on 1 column r 'e '.....................................................0001.00:001(1)
```

Für die Zeile 1 soll die Spalte ausgegeben werden, in der zum ersten Mal von rechts die Zeichenfolge E␣ auftritt.

```
    1.0000 018
 PLEASE ACKNOWLEDGE
```

```
   1.00 WIE LANG IST ZEILE 1 ?.....................................................
   2.00 UND ZEILE 2 ?..............................................................
   3.00 WER WEISS DIE LAENGE VON ZEILE 3 ?..........................................
   4.00 ...........................................................................









on 1 column all 'e '..................................................0001.00:001(1)
```

Für Zeile 1 sollen alle Spalten ausgegeben werden, in denen die Zeichenfolge E␣ auftritt.

```
    1.0000 003 018
 PLEASE ACKNOWLEDGE
```

```
   1.00 WIE LANG IST ZEILE 1 ?.................................................
   2.00 UND ZEILE 2 ?..........................................................
   3.00 WER WEISS DIE LAENGE VON ZEILE 3 ?.....................................









on & column 'e',3.............................................0001.00:001(1)
```

Zu allen Zeilen, die mindestens dreimal die Zeichenfolge E enthalten, soll die Spaltennummer des Treffers (drittes Auftreten des Suchbegriffs E) ausgegeben werden.

```
    1.0000 018
    3.0000 013
 PLEASE ACKNOWLEDGE
```

```
   1.00 WIE LANG IST ZEILE 1 ?.................................................
   2.00 UND ZEILE 2 ?..........................................................
   3.00 WER WEISS DIE LAENGE VON ZEILE 3 ?.....................................
   4.00 .......................................................................









on & column all 'e'...........................................0001.00:001(1)
```

Alle Zeilen- und Spaltennummern, in denen der Suchbegriff vorkommt, sollen ausgegeben werden.

```
    1.0000 013 015 018
    2.0000 006 009
    3.0000 002 006 013 017 020 027 030
 PLEASE ACKNOWLEDGE
```

@ON (Format 3) Zeilennummer der 1. Trefferzeile suchen

Mit diesem Format von @ON wird festgestellt, ob der Suchbegriff in dem angegebenen Bereich vorkommt und wo er zum ersten Mal auftritt. Die Zeilennummer der Trefferzeile wird in #L0, die Trefferspalten in #I0 und #I1 festgehalten.

Operation	Operanden	L-Modus / @PROC
@**O**N	range [:domain] **F**IND [{**A**LL / **F**}] [**R**] [**NOT**] [**P**ATTERN] search [,int]	

range Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort.

F — Die Überprüfung des Zeilenbereichs wird nach dem 1. Treffer abgebrochen. Die restlichen Zeilen werden nicht mehr überprüft.
Die Angabe von ALL oder F ist bei diesem Format zwar erlaubt, jedoch ohne Bedeutung, da nur der 1. Treffer festgehalten wird.

R — Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.

NOT — Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).
#I0 und #I1 werden nicht gesetzt.

PATTERN — Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.

search — Suchbegriff. Dieser kann angegeben werden:
- direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder
- indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.

int — Erst das int-te Auftreten eines Suchbegriffs in einer Zeile ist als Treffer zu werten.

Beispiel

```
1.     ************************   ------------------------------------ (01)
2.     E1**E2**E3**E4**E5**E6**
3.     @ON & FIND 'E',4 ---------------------------------------------- (02)
3.     @PRINT #L0:#I0 ------------------------------------------------ (03)
2.0000 E4**E5**E6**
3.
```

(01) 2 Zeilen werden in die virtuelle Datei gebracht.

(02) In jeder Zeile ist das vierte E zu finden. Der EDT zeigt nach außen hin keine Reaktion. Er hat aber im Trefferfall in #L0 die Nummer der Trefferzeile gespeichert und in #I0 die Spaltennummer, in der der Suchbegriff beginnt, sowie in #I1 die Spaltennummer, in der dieser endet.

(03) Hierdurch wird die 1. Trefferzeile, beginnend bei der Trefferspalte, ausgegeben.

@ON (Format 4) Markieren der Zeilen mit Suchbegriff

Dieses Format bewirkt, daß alle Sätze, die den Suchbegriff enthalten, mit der Satzmarkierung m gekennzeichnet werden. Im F-Modus wird das Arbeitsfenster auf den 1. Treffersatz positioniert.

Bereits vorhandene Satzmarkierungen (z.B. durch vorhergegangene @ON) bleiben erhalten. Sie können aber mit @DELETE MARK gelöscht werden.

Die Satzmarkierung kann zum Kopieren (@ON Format 5) oder zum Positionieren innerhalb der Arbeitsdatei (siehe Abschnitt „+ / – Positionieren in der Arbeitsdatei“ auf Seite 108ff. Format 2) verwendet werden.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range* [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**NOT**] [**P**ATTERN] search [,int] **M**ARK [m]	

range* Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen
- (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL	Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Die Angabe ist zulässig, aber ohne Bedeutung, da ein Satz nur einmal markiert werden kann.
F	Die Überprüfung jedes Zeilenbereichs wird nach dem jeweils 1. Treffer abgebrochen.
R	Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.
NOT	Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).
PATTERN	Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.
search	Suchbegriff. Dieser kann angegeben werden: – direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder – indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.
int	Erst das int-te Auftreten eines Suchbegriffs in einer Zeile ist als Treffer zu werten.
MARK m	Nummer der Markierung (1, ..., 9). Im F-Modus muß MARK m nicht angegeben werden. Es wird in diesem Fall die Markierung 1 angenommen.

i Bezieht sich die Anweisung auf eine real durch @OPEN eröffnete Datei, erfolgt keine Markierung. Es wird lediglich das Arbeitsfenster auf den 1. Treffersatz positioniert. Die explizite Angabe von MARK m wird mit einer Fehlermeldung abgewiesen.

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................









on & find 'str.' mark 2.....................................0001.00:001(1)
```

Die Sätze, die den Suchbegriff STR. enthalten, sollen mit der Satzmarkierung 2 markiert werden. Der EDT positioniert automatisch auf den 1. Treffersatz.

```
   3.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................









+(2).....................................................0003.00:001(1)
```

Das Arbeitsfenster wurde auf Zeile 3 positioniert, da diese den 1. Treffersatz enthielt.

Mit +(2) soll zum nächsten Satz mit Satzmarkierung 2 geblättert werden.

```
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................
```

@ON (Format 5) Kopieren der markierten Zeilen

Alle mit der angegebenen Satzmarkierung markierten Sätze werden in die angegebene Arbeitsdatei kopiert.

Arbeitsdateien, die gerade als EDT-Prozedur (siehe @DO) abgearbeitet werden (aktive Arbeitsdatei), können nicht als Ausgabedatei verwendet werden.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range* [:domain] **F**IND [**A**LL / **F**] [**NOT**] **M**ARK m [**C**OPY [**T**O]] (procnr) [**K**EEP] [**O**LD]	

Nur die markierten Sätze werden auf Vorhandensein der Markierung m geprüft.

range* Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.

Zeichenfolgevariablen dürfen nicht angegeben werden.

domain Spaltenbereich.

Die Angabe ist zulässig, aber ohne Bedeutung, da nur nach Satzmarkierungen gesucht wird.

ALL Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Die Angabe ist zulässig, aber ohne Bedeutung, da ein Satz nur einmal kopiert wird.

F Es wird nur der erste gefundene Satz von jedem angegebenen Zeilenbereich mit der angegebenen Satzmarkierung kopiert.
Wird F in Verbindung mit NOT angegeben, wird der erste gefundene markierte Satz von jedem Zeilenbereich kopiert, der nicht die Satzmarkierung m hat.

NOT Es werden alle markierten Zeilen kopiert, die nicht die Satzmarkierung m haben.

MARK m Nummer der Satzmarkierung (1, ..., 9).

procnr Nummer der Arbeitsdatei (0-22), in die kopiert wird.

procnr darf nicht die aktuelle Arbeitsdatei sein. Werden Treffer festgestellt und wird OLD nicht angegeben, wird der Inhalt der Zieldatei procnr vor dem Kopieren gelöscht. Wird kein Treffer festgestellt, bleibt der Inhalt von procnr unverändert. Eine aktive Arbeitsdatei (siehe @DO) kann nicht als Ausgabedatei verwendet werden.

KEEP Die Zeilennummern der Treffersätze werden beim Kopieren beibehalten. Wird KEEP nicht angegeben, erstellt der EDT die Zieldatei ab der aktuellen Zeile mit der aktuellen Schrittweite.

OLD Der Inhalt der Zieldatei procnr wird vor dem Kopieren nicht gelöscht. Eventuell existierende Zeilen mit der selben Zeilennummer in der Zieldatei werden überschrieben. Werden Treffer festgestellt und wird OLD nicht angegeben, wird der Inhalt der Zieldatei procnr vor dem Kopieren gelöscht.

Ist kein Satz markiert oder wird kein Satz mit der angegebenen Markierung gefunden, wird folgende Meldung ausgegeben: % EDT0901 NO MATCH IN RANGE

i Wird dieses Format in der Arbeitsdatei 0 eingegeben und ist eine ISAM-Datei durch @OPEN real eröffnet, so wird die Anweisung mit der Fehlermeldung % EDT4935 MAIN FILE OPENED REAL abgewiesen.

Beispiel 1

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
on 1-5 find mark 2 copy to (3) keep ; 3.........................0001.00:001(1)
```

Der Zeilenbereich 1 bis 5 soll auf die Satzmarkierung 2 überprüft und die Treffersätze unter Beibehaltung der Zeilennummer in die Arbeitsdatei 3 kopiert werden. Anschließend soll in die Arbeitsdatei 3 verzweigt werden.

```
   3.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND.............................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG.........................
   5.00 ..........................................................................
```

```
.....................................................................0003.00:001(3)
```

Beispiel 2

Es sollen alle Zeilen kopiert werden, die nicht markiert sind. Dazu müssen in einem ersten Schritt alle Zeilen mit einer noch nicht vergebenen Satzmarkierung (z.B. 9) markiert werden (@ON Format 4). Anschließend werden dann alle Zeilen kopiert, die nicht die Satzmarkierung (z.B. 1 bis 3) haben (@ON Format 5).

```
@ON & FIND PATTERN '*' MARK 9
@ON & FIND NOT MARK 1 COPY TO (1) KEEP
@PROC 1
@ON & FIND NOT MARK 2 COPY TO (2) KEEP
@PROC 2
@ON & FIND NOT MARK 3 COPY TO (3) KEEP
@PROC 3
```

Arbeitsdatei 1 enthält alle Zeilen, die nicht die Satzmarkierung 1 haben.
Arbeitsdatei 2 enthält alle Zeilen, die nicht die Satzmarkierung 1 oder 2 haben.
Arbeitsdatei 3 enthält alle Zeilen, die nicht die Satzmarkierung 1 oder 2 oder 3 haben.

@ON (Format 6) Kopieren der Zeilen mit dem Suchbegriff

Mit diesem Format werden alle Sätze, die den Suchbegriff enthalten, in die angegebene Arbeitsdatei kopiert. Werden Treffer festgestellt, wird standardmäßig vor dem Kopiervorgang der Inhalt der Zieldatei procnr gelöscht. Wird kein Treffer festgestellt oder wird OLD angegeben, bleibt der Inhalt unverändert.

Arbeitsdateien, die gerade als EDT-Prozedur (siehe @DO) abgearbeitet werden (aktive Arbeitsdatei), können nicht als Ausgabedatei verwendet werden.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range* [:domain] **F**IND [{**A**LL / **F**}] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**C**OPY [**T**O]] (procnr) [**K**EEP] [**O**LD]	

range* Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL	Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Die Angabe ist zulässig, aber ohne Bedeutung, da ein Satz nur einmal kopiert wird.
F	In jedem angegebenen Zeilenbereich wird nur der erste gefundene Suchbegriff kopiert.
R	Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.
NOT	Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).
PATTERN	Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.
search	Suchbegriff. Dieser kann angegeben werden: – direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder – indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.
int	Erst das int-te Auftreten eines Suchbegriffs in einer Zeile ist als Treffer zu werten.
procnr	Nummer der Arbeitsdatei (0-22), in die kopiert wird. procnr darf nicht die aktuelle Arbeitsdatei sein. Werden Treffer festgestellt und wird OLD nicht angegeben, wird der Inhalt der Zieldatei procnr vor dem Kopieren gelöscht. Wird kein Treffer festgestellt, bleibt der Inhalt von procnr unverändert. Eine aktive Arbeitsdatei (siehe @DO) kann nicht als Ausgabedatei verwendet werden.
KEEP	Die Zeilennummern der Treffersätze werden beim Kopieren beibehalten. Wird KEEP nicht angegeben, erstellt der EDT die Zieldatei ab der aktuellen Zeile mit der aktuellen Schrittweite.
OLD	Der Inhalt der Zieldatei procnr wird vor dem Kopieren nicht gelöscht. Eventuell existierende Zeilen mit der selben Zeilenummer in der Zieldatei werden überschrieben. Werden Treffer festgestellt und wird OLD nicht angegeben, wird der Inhalt der Zieldatei procnr vor dem Kopieren gelöscht.

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
   2.00 HOFER    MARIA     GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................









on & find 'str.' copy to (5) ; 5.................................0001.00:001(1)
```

Sätze mit dem Suchbegriff STR. sollen unter Beibehaltung der Zeilennummern in die Arbeitsdatei 5 kopiert werden.
Anschließend soll in die Arbeitsdatei 5 verzweigt werden.

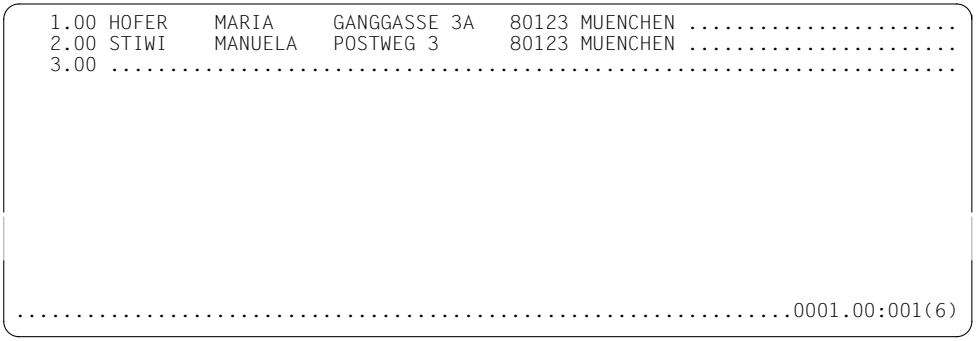
```
   1.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND..........................
   2.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   3.00 ..........................................................................









1.............................................................0001.00:001(5)
```

Es wird in die Arbeitsdatei 1 verzweigt.

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN ......................
   2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN ......................
   3.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND..........................
   4.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG......................
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   6.00 ..........................................................................









on & :10-20: find pattern 'm*a' copy to (6) ; 6..................0001.00:001(1)
```

Alle Sätze der Personen, deren Vornamen mit M beginnen und mit A enden werden in die Arbeitsdatei 6 kopiert.
Anschließend wird in die Arbeitsdatei 6 verzweigt.

```
   1.00 HOFER    MARIA     GANGGASSE 3A   80123 MUENCHEN ......................
   2.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN ......................
   3.00 ................................................................................
```



@ON (Format 7) Ersetzen des Suchbegriffs

Im Trefferfall bewirkt dieses Format, daß der Suchbegriff durch eine Zeichenfolge ersetzt wird.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **C**HANGE [{**A**LL / **F**}] [**R**] [**P**ATTERN] search [,int] [**T**O] string	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte der Rest der Zeile durchsucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile durchsucht.

ALL — Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Zuvor wird der Suchbegriff durch string ersetzt.

F — Die Überprüfung jedes angegebenen Zeilenbereichs wird nach dem jeweils ersten Treffer abgebrochen.
Wird ALL oder F nicht angegeben, wird jeder erste Treffer pro Zeile gewertet.

R — Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.

PATTERN — Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.

search — Suchbegriff. Dieser kann angegeben werden:
– direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder
– indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.

int — Erst das int-te Auftreten eines Suchbegriffs in einer Zeile ist als Treffer zu werten. Für int können Werte zwischen 1 und 256 angegeben werden. Standardwert ist 1.

string — Zeichenfolge.
string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

string ersetzt den Suchbegriff. Wird ein Leerstring angegeben, wird der Suchbegriff gelöscht!

Beispiel

```
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................
   2.00 WER ISTIST HIER RR WIE OSKAR ?..........................................
   3.00 ........................................................................









on 2 change 'r',3 to 1:19-21 ; on 2 change 'ist' to ''............0001.00:001(1)
```

Mit dem ersten @ON soll in Zeile 2 nach dem dritten Auftreten des Zeichens R gesucht werden. Dieses R soll durch die Zeichenfolge aus Zeile 1 in den Spalten 19 bis 21, d.h. STU ersetzt werden.

Mit dem zweiten @ON soll in Zeile 2 das erste Auftreten der Zeichenfolge IST durch die leere Zeichenfolge ersetzt, also gelöscht werden (Alternative zu @ON Format 9).

```
   1.00 ABCDEFGHIJKLMNOPQRSTUVWXYZ.............................................
   2.00 WER IST HIER STUR WIE OSKAR ?..........................................
   3.00 ........................................................................
```

@ON (Format 8) Ersetzen oder Einfügen vor oder nach dem Suchbegriff

Dieses Format von @ON bietet zwei Möglichkeiten den Zeileninhalt zu verändern. Die mit string angegebene Zeichenfolge

– ersetzt (CHANGE) den Zeileninhalt vor (PREFIX) oder nach (SUFFIX) dem Suchbegriff oder

– wird vor (PREFIX) oder nach (SUFFIX) dem Suchbegriff eingefügt (INSERT).

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **F**IND [{**A**LL / **F**}] [**R**] [**P**ATTERN] search [,int] {**C**HANGE / **I**NSERT} {**P**REFIX / **S**UFFIX} string	

range Zeilenbereich, bestehend aus:
– einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
– einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
– einer Kombination von einzelnen Zeilen und Zeilenbereichen
– (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain Spaltenbereich bestehend aus:
– einer einzelnen Spalte (z.B. 10-10)
– einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
– darf nicht kleiner als die erste sein,
– kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL	Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Zuvor wird der Text vor oder nach dem Suchbegriff durch string ersetzt bzw. string vor oder nach dem Suchbegriff eingefügt.
F	Die Überprüfung jedes angegebenen Zeilenbereichs wird nach dem jeweils ersten Treffer abgebrochen. Wird ALL oder F nicht angegeben, wird jeder erste Treffer pro Zeile gewertet.
R	Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.
PATTERN	Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.
search	Suchbegriff. Dieser kann angegeben werden: – direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder – indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.
int	Erst das int-te Auftreten des Suchbegriffs in einer Zeile ist als Treffer zu werten.
CHANGE	Der vor oder nach dem Suchbegriff stehende Text wird durch die mit string angegebene Zeichenfolge ersetzt.
INSERT	Die mit string angegebene Zeichenfolge wird vor oder nach dem Suchbegriff eingefügt.
PREFIX	Die mit string angegebene Zeichenfolge ersetzt den Zeileninhalt vor dem Suchbegriff oder wird vor dem Suchbegriff eingefügt.
SUFFIX	Die mit string angegebene Zeichenfolge ersetzt den Zeileninhalt nach dem Suchbegriff oder wird nach dem Suchbegriff eingefügt.
string	Zeichenfolge. string kann angegeben werden: – explizite Angabe in Hochkomma, – implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich). string ersetzt den Text vor oder nach dem Suchbegriff bzw. wird vor oder nach dem Suchbegriff eingefügt. Wird PREFIX bzw. SUFFIX ganz ausgeschrieben und als string eine Zeichenfolge angegeben, muß zwischen PREFIX bzw. SUFFIX und dem Hochkomma ein Leerzeichen stehen.

Beispiel 1

```
  23.00 ..........................................................................
fstat '$user2.bsp.'.....................................0001.00:001(1)
```

Alle mehrfachbenutzbaren Dateien der Benutzerkennung USER2, die mit dem teilqualifizierten Namen BSP. beginnen, sollen aufgelistet werden.

```
   1.00 BSP.1.....................................................................
   2.00 BSP.2.....................................................................
   3.00 BSP.3.....................................................................
   4.00 BSP.4.....................................................................
   5.00 ..........................................................................









on & find 'bsp.' insert prefix '@read ''$user2.'.................0001.00:001(9)
```

Jedem teilqualifizierten Namen BSP. soll die Zeichenfolge @read $user2. vorangestellt werden.

```
   1.00 @READ '$USER2.BSP.1.......................................................
   2.00 @READ '$USER2.BSP.2.......................................................
   3.00 @READ '$USER2.BSP.3.......................................................
   4.00 @READ '$USER2.BSP.4.......................................................
   5.00 ..........................................................................









1 ; do 9.....................................................0001.00:001(9)
```

Die vier Dateien BSP.1 bis BSP.4 werden hintereinander in die Arbeitsdatei 1 eingelesen.

Beispiel 2

```
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 A11B11C11D11E11F11G11H11.....................................................
   2.00 A1111B1111C1111D1111E1111.....................................................
   3.00 .....................................................................................









on 1-2 find r '11',4 insert suffix '++++'.........................0001.00:001(1)
```

Im Zeilenbereich 1 bis 2 soll von rechts nach links nach dem vierten Auftreten der Zeichenfolge 11 gesucht und im Trefferfall dahinter ++++ eingefügt werden.

```
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 A11B11C11D11E11++++F11G11H11.....................................................
   2.00 A1111B1111C1111D1111++++E1111.....................................................
   3.00 .....................................................................................









on &:4 find '111',3 change suffix '####'.........................0001.00:001(1)
```

In Zeile 1 trat der Suchbegriff zum vierten Mal von rechts in den Spalten 14–15 auf.
In Zeile 2 trat der Suchbegriff zum ersten Mal in den Spalten 24-25, zum zweiten Mal in den Spalten 23-24, zum dritten Mal in den Spalten 22-23 und zum vierten Mal als Treffer in den Spalten 19-20 auf. Hinter dem Treffer wurde die Zeichenfolge ++++ eingefügt.

Nun soll in der gesamten Arbeitsdatei ab Spalte 4 nach dem dritten Auftreten der Zeichenfolge 111 gesucht werden. Im Trefferfall ist der nachfolgende Text zu ersetzen durch ####.

```
   1.00 A11B11C11D11E11++++F11G11H11.....................................................
   2.00 A1111B1111C111####.....................................................
   3.00 .....................................................................................
```

In Zeile 1 trat der Suchbegriff nicht auf.
In Zeile 2 trat der Suchbegriff beginnend ab Spalte 4 zum ersten Mal in den Spalten 7–9, zum zweiten Mal in den Spalten 8–10 und zum dritten Mal als Treffer in den Spalten 12–14 auf. Nach dem Treffer wurde der Zeilenrest durch die Zeichenfolge #### ersetzt.

@ON (Format 9) Löschen des Suchbegriffs

Im Trefferfall bewirkt dieses Format von @ON, daß der Suchbegriff gelöscht wird. Der restliche Zeileninhalt des Satzes bleibt erhalten.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **D**ELETE [**A**LL | **F**] [**R**] [**P**ATTERN] search [,int]	

range Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Zuvor löscht er den gefundenen Suchbegriff.

F — Die Überprüfung jedes angegebenen Zeilenbereichs wird nach dem jeweils ersten Treffer abgebrochen.
Wird ALL oder F nicht angegeben, wird jeder erste Treffer pro Zeile gewertet.

R — Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.

PATTERN — Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.

search — Suchbegriff. Dieser kann angegeben werden:
- direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder
- indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.

int — Erst das int-te Auftreten des Suchbegriffs in einer Zeile ist als Treffer zu werten.

Beispiel

```
   1.00 XXXYYYZZZ *** XXXYYYZZZ ### XXXYYYZZZ %%%.............................
   2.00 AAA XXXYYYZZZ BBB XXXYYYZZZ CCC XXXYYYZZZ.............................
   3.00 ...........................................................................
```

```
on & delete f r 'xxxyyyzzz'.....................................0001.00:001(1)
```

In der gesamten Arbeitsdatei sollen die Zeilen von rechts nach links nach der Zeichenfolge XXXYYYZZZ durchsucht werden. Beim ersten Auftreten des Suchbegriffs soll diese gelöscht und das Suchen beendet werden.

```
   1.00 XXXYYYZZZ *** XXXYYYZZZ ###  %%%.....................................
   2.00 AAA XXXYYYZZZ BBB XXXYYYZZZ CCC XXXYYYZZZ.............................
   3.00 .....................................................................
```

```
on & delete all 'xxxyyyzzz'.....................................0001.00:001(1)
```

In Zeile 1 wurde der Suchbegriff ab Spalte 29 zum ersten Mal von rechts aus gefunden und gelöscht.

Anschließend soll in der gesamten Arbeitsdatei der Suchbegriff XXXYYYZZZ gelöscht werden.

```
   1.00  ***  ###  %%%.....................................................
   2.00 AAA  BBB  CCC.....................................................
   3.00 .....................................................................
```

@ON (Format 10) Löschen des Zeileninhaltes vor oder nach dem Suchbegriff

Im Trefferfall bewirkt dieses Format von @ON, daß der Inhalt der Trefferzeile vor (PREFIX) oder nach (SUFFIX) dem Suchbegriff gelöscht wird.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **F**IND [{**A**LL / **F**}] [**R**] [**P**ATTERN] search [,int] **D**ELETE {**P**REFIX / **S**UFFIX}	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht. Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL — Der EDT setzt nach Feststellung eines Treffers die Untersuchung der Zeile fort. Zuvor löscht er den vor dem Suchbegriff stehenden Text.

F — Die Überprüfung jedes angegebenen Zeilenbereichs wird nach dem jeweils ersten Treffer abgebrochen.

R	Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.
PATTERN	Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.
search	Suchbegriff. Dieser kann angegeben werden: – direkt in Form einer Zeichenfolge, die in Hochkommas eingeschlossen ist, oder – indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.
int	Erst das int-te Auftreten des Suchbegriffs in einer Zeile ist ein Treffer.
PREFIX	Der Inhalt der Trefferzeile vor dem Suchbegriff wird gelöscht.
SUFFIX	Der Inhalt der Trefferzeile nach dem Suchbegriff wird gelöscht.

Beispiel

```
   1.00 ABABAB ABABAB ABABAB ABABAB.........................................
   2.00 ABABABABABABABABABABABAB...........................................
   3.00 ...................................................................
                                                                           
                                                                           
                                                                           
                                                                           
                                                                           
                                                                           
                                                                           
on %.-.$ find 'ab'*3 , 4 delete prefix.....................0001.00:001(1)
```

Im gesamten Zeilenbereich (%.-.$) soll in jeder Zeile beim vierten Auftreten der Zeichenfolge ABABAB (AB*3,4) der dem Treffer vorausgehende Text gelöscht werden.

```
   1.00 ABABAB.............................................................
   2.00 ABABABABABABABABAB.................................................
   3.00 ...................................................................
```

In Zeile 1 wurde der Suchbegriff zum vierten Mal als Treffer ab Spalte 22 gefunden.
In Zeile 2 überlagerte sich das Auftreten des Suchbegriffs:
der Suchbegriff wurde ab Spalte 1 zum ersten Mal, ab Spalte 3 zum zweiten Mal, ab Spalte 5 zum dritten Mal und ab Spalte 7 als Treffer zum vierten Mal gefunden.

@ON (Format 11) Löschen der Zeile mit dem Suchbegriff

Dieses Format von @ON bewirkt, daß die ganze Zeile gelöscht wird, die den Suchbegriff enthält.

Operation	Operanden	F-Modus / L-Modus
@**O**N	range [:domain] **F**IND [{ **A**LL / **F** }] [**R**] [**NOT**] [**P**ATTERN] search [,int] **D**ELETE	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte im Rest der Zeile gesucht. Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird in der gesamten Zeile gesucht.

ALL — kann zwar angegeben werden, ist aber wirkungslos.

F — Nur der erste gefundene Satz jedes angegebenen Zeilenbereichs wird gelöscht.

Wird ALL oder F nicht angegeben, wird jeder erste Treffer pro Zeile gewertet.

R — Die Zeilen werden von rechts nach links durchsucht. Standardmäßig werden sie von links nach rechts durchsucht.

NOT — Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).

PATTERN — Die in search vorkommenden aktuellen Zeichen für asterisk und slash werden als Jokerzeichen interpretiert.

search — Suchbegriff. Dieser kann angegeben werden:
- direkt in Form einer Zeichenfolge, die in Hochkomma eingeschlossen ist, oder
- indirekt, indem eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils evtl. mit Spaltenangabe) angegeben wird (z.B. 5:2-6: oder #L2 oder #S5:2-3:). Die Zeile mit der angegebenen Zeilennummer oder die Variable muß dann den gewünschten Suchbegriff enthalten.

int — Erst das int-te Auftreten des Suchbegriffs in einer Zeile ist als Treffer zu werten.

Beispiel

```
   1.00 1  ABC  2  ABC  3  ABC  4  ABC  5  ABC.................................
   2.00 1  ABC  2  ABC  3  ABC  4  ABC.......................................
   3.00 1  ABC  2  ABC  3  ABC...............................................
   4.00 1  ABC  2  ABC.......................................................
   5.00 1  ABC...............................................................
   6.00 1....................................................................
   7.00 .....................................................................




on & find 'abc',3 delete.....................................0001.00:001(1)
```

In der gesamten Arbeitsdatei sollen alle Zeilen gelöscht werden, in denen die Zeichenfolge ABC mindestens dreimal auftritt.

```
   4.00 1  ABC  2  ABC.....................................................
   5.00 1  ABC.....................................................
   6.00 1.....................................................
   7.00 .....................................................
```

```
on & : 7 find 'a' delete.....................................0004.00:001(1)
```

Die Zeilen 1, 2 und 3 wurden gelöscht.

Anschließend sollen in der gesamten Arbeitsdatei alle Zeilen gelöscht werden, in denen ab Spalte 7 das Zeichen A auftritt.

```
   5.00 1  ABC.....................................................
   6.00 1.....................................................
   7.00 .....................................................
```

```
on & find ' '*2 delete.....................................0004.00:001(1)
```

Zeile 4 wurde gelöscht.

Anschließend sollen in der gesamten Arbeitsdatei alle Zeilen gelöscht werden, in der zwei aufeinanderfolgende Leerzeichen ’␣␣’ auftreten.

```
   6.00 1.....................................................
   7.00 .....................................................
```

Zeile 5 wurde gelöscht.

@OPEN Öffnen und Einlesen einer Datei oder eines Bibliothekselementes

@OPEN bietet in 2 Formaten folgende Funktionen:

– Öffnen und Einlesen einer ISAM-Datei zur realen Bearbeitung (Format 1).

– Kopieren einer Datei in eine ISAM-Datei (Format 1).

– Öffnen und Einlesen eines Programm-Bibliothekselementes oder einer Datei (Format 2). Das Bibliothekselement bzw. die Datei wird in die aktuelle Arbeitsdatei eingelesen und kann dort bearbeitet werden.

@OPEN (Format 1)
Bearbeiten einer ISAM-Datei direkt auf Platte (reale Bearbeitung)

Die ISAM-Datei wird nicht in den Speicherbereich des EDT eingelesen. In die aktuelle Arbeitsdatei, die leer sein muß, wird nur der gerade bearbeitete Teil der ISAM-Datei eingelesen, also ein Arbeitsfensterinhalt. Jede Änderung wird nach dem Drücken der Taste DUE sofort in die Plattendatei übernommen. Die Datei bleibt bis zum Ende ihrer Bearbeitung durch @CLOSE physikalisch eröffnet.

Kopieren von Dateien und reale Bearbeitung von SAM-Dateien

Mit @OPEN kann eine zu bearbeitende SAM- oder ISAM-Datei (file1) in eine ISAM-Datei kopiert werden. Die Kopie der Datei (file2) wird anschließend eröffnet.
Eine SAM-Datei, die real bearbeitet werden soll, kann somit mit @OPEN vor der Bearbeitung in eine ISAM-Datei kopiert und eröffnet werden. Siehe dazu Abschnitt „Reale Bearbeitung einer SAM-Datei“ auf Seite 367.

Operation	Operanden	F-Modus / L-Modus
@**OP**EN	['file1'] [(ver)] [[**K**EY] [**A**S 'file2' [**O**VERWRITE]]]	

file1 — Dateiname. Falls der mit file1 vereinbarte Dateiname noch nicht existiert, wird eine Datei mit diesem Namen katalogisiert. file1 kann entfallen, wenn zuvor über @FILE ein Dateiname vereinbart worden ist. Ist über @FILE zuvor kein Dateiname vereinbart worden, muß file1 angegeben werden, sonst wird die @OPEN-Anweisung mit einer Fehlermeldung abgewiesen.

ver — Versionsnummer der Datei.
Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer.

KEY findet nur bei SAM-Dateien Anwendung, die mit @WRITE und KEY erstellt wurden. KEY bewirkt, daß die in der SAM-Datei festgehaltenen Schlüssel als Zeilennummern und nicht als Zeileninhalt interpretiert werden

file2 Dateiname. Falls der mit file2 vereinbarte Dateiname noch nicht existiert, wird eine Datei mit diesem Namen katalogisiert. Wird file2 angegeben, wird file1 in file2 kopiert und file2 eröffnet.
Ist OVERWRITE nicht angegeben, wird, falls file2 vorhanden und nicht leer ist, eine Warnung ausgegeben.
@OPEN wird nur dann ausgeführt, wenn
- der Benutzer die Warnung mit Y beantwortet,
- die Datei file1 nicht leer ist und
- der Dateiname file2 ungleich file1 ist.

OVERWRITE Unterdrückt die Abfrage % EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)? für eine vorhandene Datei file2. Eine Datei mit gleichem Namen wird überschrieben. Existiert die Datei file2 noch nicht, ist OVERWRITE wirkungslos.

ISAM-Dateien können nur in der Arbeitsdatei 0 real bearbeitet werden. Die Arbeitsdatei muß leer sein.

Nach @OPEN werden die Anweisungen @RUN, @RENUMBER und @COMPARE abgewiesen. Sätze von real eröffneten Dateien können nicht markiert werden (z.B. @ON Format 4) und werden nicht automatisch gesichert (siehe @AUTOSAVE).

Wird mit @OPEN eine falsche Versionsnummer ver genannt, so wird lediglich die aktuelle Versionsnummer ausgegeben, die @OPEN-Anweisung jedoch nicht durchgeführt.
Wird mit @OPEN die aktuelle Versionsnummer oder * angegeben, erscheint nach @CLOSE die um 1 erhöhte und damit die neueste Versionsnummer.

Schließen einer mit @OPEN eröffneten Datei

- Mit @CLOSE wird die Datei geschlossen und die Arbeitsdatei und der Eintrag für den lokalen Dateinamen gelöscht. Nachfolgende Eingaben beziehen sich nicht mehr auf die Plattendatei, sondern auf die Arbeitsdatei (virtuelle Datei).
- Mit einem zweiten @OPEN wird implizit ein @CLOSE ausgelöst. Es wird zuerst die erste Datei geschlossen und die Arbeitsdatei gelöscht. Erst dann wird die zweite Datei eröffnet.

Reale Bearbeitung von ISAM-Dateien

Die Zeilennummern in der Arbeitsdatei entsprechen den ISAM-Schlüsseln der durch @OPEN eröffneten ISAM-Datei.

Vom EDT wird nicht überprüft, ob irgendwelche Zeilen aus mehr als 256 Zeichen bestehen. Beim Einlesen einer derartigen Zeile gehen die Zeichen ab Spalte 257 verloren.

Wird file2 angegeben, wird eine Kopie der Originaldatei durch ein implizites COPY-FILE-Kommando erstellt. Eröffnet und bearbeitet wird die Kopie file2. Die Originaldatei file1 wird nicht eröffnet.

Reale Bearbeitung von ISAM-Dateien mit ISAM-Schlüssellänge kleiner als 8 Byte

Vor dem EDT-Aufruf ist in folgendem Systemkommando die Schlüssellänge anzugeben:

```
/SET-FILE-LINK LINK-NAME = EDTMAIN, FILE-NAME = dateiname, -
/    ACCESS-METHOD = ISAM(KEY-LENGTH = schlüssellänge)
```

Nach dem Aufruf des EDT wird die Datei mit @OPEN eröffnet. Dabei ist die Angabe '/' anstatt des Dateinamens nicht möglich.

Es empfiehlt sich, nach dem Schließen der Datei die Zuordnung des Dateikettungsnamens wieder aufzuheben mit dem Systemkommando

```
/REMOVE-FILE-LINK LINK-NAME = EDTMAIN
```

Standardmäßig interpretiert oder erzeugt der EDT ISAM-Schlüssel, die 8 Zeichen lang sind. Ist eine kürzere Schlüssellänge vereinbart, wird ein vorhandener ISAM-Schlüssel von links her verkürzt. Beispielsweise wird bei einer KEYLEN-Angabe von 4 die Zeilennummer 1234.5678 als ISAM-Schlüssel 5678 interpretiert. Die Eindeutigkeit eines ISAM-Schlüssels ist damit nicht mehr gewährleistet. Der Benutzer hat selbst für diese Eindeutigkeit zu sorgen, falls er ISAM-Schlüssel von weniger als 8 Zeichen wünscht.

Die Bearbeitung von ISAM-Dateien mit fester Satzlänge (RECORD-FORMAT=FIXED) wird nicht unterstützt.

Reale Bearbeitung einer SAM-Datei

Die SAM-Datei file1, die real bearbeitet werden soll, muß vor der Bearbeitung in eine ISAM-Datei file2 kopiert werden. Dazu ist in der @OPEN-Anweisung „AS 'file2'" anzugeben. Der EDT kopiert die SAM-Datei file1 in die ISAM-Datei file2 und eröffnet die Datei file2. Das Kopieren erfolgt durch ein implizites @READ auf die 1. Datei und ein nachfolgendes @SAVE mit Angabe des 2. Dateinamens. Der Dateikettungsname für die erste Datei lautet EDTSAM, der für die zweite Datei EDTMAIN.

Der ISAM-Schlüssel der ersten Zeile von file2 ist die vor Ausführung des @OPEN aktuelle Zeilennummer. Die ISAM-Schlüssel der folgenden Zeilen ergeben sich dadurch, daß der Schlüssel jeweils um den Wert erhöht wird, den die aktuelle Schrittweite vor Ausführung des @OPEN hatte (siehe @ln oder @SET, Format 6).

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar) ist mit @OPEN der Coded Character Set Name (CCSN) als Codemerkmal der Datei berücksichtigt.

Die @OPEN-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei gleich dem im EDT aktuell eingestelltem ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

@OPEN (Format 2) Eröffnen und Einlesen in die aktuelle Arbeitsdatei

Mit @OPEN (Format 2) wird ein Bibliothekselement oder eine SAM- bzw. ISAM-Datei eröffnet und in die aktuelle Arbeitsdatei eingelesen.

Dateien, deren Dateiattribute von den Standardwerten abweichen, können ohne vorhergehendes SET-FILE-LINK-Kommando eröffnet werden.

@OPEN (FORMAT 2) kann in den Arbeitsdateien 0 bis 22 eingegeben werden.

Operation	Operanden	F-Modus / L-Modus
@**OP**EN	{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) **E**LEMENT=elemname [(vers)][,elemtyp] **F**ILE=path2 [,**T**YPE=**I**SAM | **S**AM | **C**ATALOG] } [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	

Wird mehr als ein Operand angegeben, müssen die einzelnen Operanden durch Leerzeichen oder Komma voneinander getrennt werden.

LIBRARY = path1 ([E[LEMENT]=]elemname [(vers)][,elemtyp])
: Name des Elements mit Angabe des Bibliotheknamens.

ELEMENT = elemname [(vers)][,elemtyp]
: Name des Elements ohne Angabe des Bibliotheknamens. Voraussetzung ist die Voreinstellung des Bibliotheknamens mit @PAR.
Nach erfolgreichem @OPEN wird der Bibliotheksname ausgegeben.

path1
: Name der Bibliothek. path1 kann auch als Zeichenfolgevariable angegeben werden.
Wird path1 nicht angegeben, wird die mit @PAR LIBRARY voreingestellte Bibliothek verwendet.

elemname
: Name des Elements. elemname kann auch als Zeichenfolgevariable angegeben werden.

vers
: Versionsbezeichnung des gewünschten Elements (siehe Handbuch „LMS“ [13]). Wird vers nicht angegeben, oder *STD, wird die höchste vorhandene Version des Elementes gewählt.

elemtyp
: Typ des Elements. elemtyp kann auch als Zeichenfolgevariable angegeben werden. Zulässige Typangaben sind: S, M, P, J, D, X, *STD und freie Typnamen mit entsprechendem Basistyp. Falls nicht angegeben, wird der in @PAR ELEMENT-TYPE voreingestellte Wert verwendet.

Wird ein freier Typname verwendet, so liegt es in der Verantwortung des Benutzers, daß der zugehörige Basistyp einem zulässigen Typ S, M, P, J, D oder X entspricht.

Typ	Elementinhalt
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats

*STD
Typ S ist die Voreinstellung nach Aufruf des EDT. Mit @PAR kann eine andere zulässige Typangabe als Voreinstellung festgelegt werden.

FILE = path2 Eröffnen und Einlesen einer BS2000-Datei.

path2 Name der Datei (vollqualifizierter Dateiname), die geöffnet werden soll. path2 kann auch als Zeichenfolgevariable angegeben werden.

TYPE Festlegen der Zugriffsmethode der Datei.

= SAM Standardwert. Die zu öffnende Datei ist eine SAM-Datei.

= ISAM Die zu öffnende Datei ist eine ISAM-Datei.

= CATALOG
Die Attribute werden aus dem Katalogeintrag der existierenden Datei übernommen.

Die Zugriffsmethode wird durch das FCBTYPE-Attribut im Katalogeintrag bestimmt.
Dateien mit vom Standard abweichenden Attributen können bearbeitet werden (entspricht @READ und @WRITE mit vorheriger Zuweisung zum Linknamen EDTSAM, bzw. @GET und @SAVE mit vorheriger Zuweisung zum Linknamen EDTISAM). Dateien, die mit SET-FILE-LINK LINK-NAME=EDTSAM, ACCESS-METHOD=ISAM zu lesen sind, können nicht mit @OPEN geöffnet werden.

MODE Festlegen des Eröffnungsmodus des Bibliothekselements bzw. der Datei.

= ANY Standardwert. Ein existierendes oder neues Bibliothekselement bzw. eine Datei kann eröffnet werden.

= UPDATE Ein existierendes Bibliothekselement bzw. eine Datei wird zur Bearbeitung eröffnet.

= NEW Ein Bibliothekselement bzw. eine Datei wird neu angelegt.
Dabei darf derselbe Elementname in der Bibliothek bzw. die Datei noch nicht vorhanden sein.

= REPLACE
Der Inhalt eines existierenden Elementes bzw. der Datei soll ersetzt werden. Der Inhalt wird nicht in die Arbeitsdatei eingelesen.

Ist ein Bibliothekselement bzw. eine Datei bereits in einer anderen Arbeitsdatei eröffnet, erfolgt eine Fehlermeldung.

Berechnung der Zeilennummern beim Einlesen

Die Datensätze werden beim Einlesen nach drei Prinzipien numeriert:

1. Standardnumerierung mit Standardschrittweite 1.0000
   (z.B. 1.0000, 2.0000, 3.0000 ... 999.0000) oder
2. Numerierung mit festgelegter Schrittweite gemäß @PAR INCREMENT oder
3. Automatische Numerierung bei @PAR RENUMBER=ON:
   Sie erfolgt, wenn die Schrittweite zu groß ist, um die Datei einlesen zu können.
   Der EDT wählt eine Schrittweite, die um den Faktor 10 kleiner ist als die Standardschrittweite (1.) bzw. festgelegte Schrittweite (2.). Mit der kleineren Schrittweite werden bereits gelesene Sätze umnumeriert und die Zeilennummern der Sätze, die weiter eingelesen werden, berechnet.
   Dieser Vorgang wird solange wiederholt, bis die Datei erfolgreich eingelesen werden konnte oder die Datei selbst mit der minimalen Schrittweite von 0.0001 nicht eingelesen werden konnte (Abbruch des Einlesevorgangs mit Fehlermeldung).

   Bei einer Schrittweite < 0.01 ist zu beachten:

   Im F-Modus werden die Zeilennummern von eingelesenen, kopierten oder eingefügten Zeilen nicht vollständig ausgegeben (6-stellige Zeilennummernanzeige).
   Werden diese unvollständig ausgegebenen Zeilennummern in Anweisungen verwendet (@COPY, usw.), kann dies zu Fehlern führen.

   Die aktuelle Zeilennummer wird auf den Wert der letzten eingelesenen Zeile plus der aktuellen Schrittweite gesetzt.

Ist die bearbeitete Bibliothek Teil einer Dateigenerationsgruppe, muß der Typ des Bibliothekselements zuerst mit @PAR ELEMENT-TYPE = elementtyp vereinbart werden.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @OPEN der Coded Character Set Name (CCSN) als Codemerkmal der Datei (Bibliothekselement) berücksichtigt.

Die @OPEN-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei (Bibliothekselement) gleich dem im EDT aktuell eingestelltem ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

Beispiel

OPEN LIBRARY = PROGLIB (ELEMENT = TEST)

Das Bibliothekselement TEST der Programmbibliothek PROGLIB wird eröffnet und in die aktuelle Arbeitsdatei eingelesen. Diese Arbeitsdatei muß leer sein.

OPEN ELEMENT = PROC.EX(3), J

Vor dieser Anweisung muß mit @PAR LIBRARY = libname die Programmbibliothek zugewiesen werden, in der das Element PROC.EX abgespeichert ist. Das Bibliothekselement PROC.EX mit einer Prozedur als Inhalt (Elementtyp J) wird eröffnet. Es wird die dritte Version des Elements in die aktuelle Arbeitsdatei eingelesen. Das Element bleibt geöffnet.

@P-KEYS Belegen programmierbarer Tasten

Mit @P-KEYS kann man

– die programmierbaren Tasten der Tastatur belegen (@P-KEYS)
– die vom EDT als Standardbelegung angenommenen Funktionen der belegten Tasten ausgeben (@P-KEYS SHOW)

Operation	Operanden	F-Modus / L-Modus
@P-KEYS	[**S**HOW]	

SHOW Die Funktionen der P-Tastenbelegung werden ausgegeben, wie sie vom EDT mit @P-KEYS belegt werden. Dieser Operand wird im Dialogbetrieb an allen Datensichtstationen unterstützt, die zum Typ 8160 kompatibel sind.

Die P-Tasten werden folgendermaßen belegt:

```
***  MEANING OF THE P-KEYS                                                ***
P1 : position CURSOR to 1st command line
P2 : position CURSOR to 2nd command line
P3 :
P4 : skip to next page in first window
P5 : skip to previous page in first window
P6 : skip to next page in first window for corrections
P7 : skip to next page in second window
P8 : skip to previous page in second window
P9 : skip to next page in second window for corrections
P10: skip to the next mark in the first window
P11: skip to the previous mark in the first window
P12: position CURSOR eight characters to the right
P13: skip to the next mark in the second window
P14: skip to the previous mark in the second window
P15:
P16:
P17:
P18:
P19:
P20:
--------------------------------------------------------------------------------
                         PRESS DUE1      FOR RETURN
```

– Wird mit @PAR SPLIT die Bildschirmaufteilung verändert oder mit @VDT an einer Datensichtstation 9763 das Format gewechselt, ist @P-KEYS erneut einzugeben.
– An der DSS 3270 wird @P-KEYS mit einer Fehlermeldung abgewiesen.

@PAGE Seitenvorschub

Diese Anweisung bewirkt einen Seitenvorschub auf SYSLST.

Operation	Operanden	F-Modus / L-Modus
@**PA**GE		

Mit @LIST kann die Maximalzahl der auszudruckenden Zeilen pro Seite auf einen Wert zwischen 1 und 256 gesetzt werden. Mit @PAGE wird diese Vereinbarung zurückgenommen und der Standardwert 65 angenommen.

@PAR Eingabe von Voreinstellwerten

Mit @PAR werden Voreinstellwerte festgelegt.

Die Voreinstellwerte beziehen sich auf Funktionen für die Dateibearbeitung:

Einschalten oder Ausschalten von Funktionen:

- EDIT LONG-Modus (EDIT LONG)
- Hexadezimal-Modus (HEX)
- Klein- oder Großbuchstabenumsetzung (LOWER)
- Überschreibmodus (EDIT FULL)
- Schreibschutz auf Satzebene (PROTECTION) (nur für EDT-Unterprogrammschnittstelle)
- Spaltenzähleranzeige (SCALE)
- Informationszeilenanzeige (INFORMATION)
- Zeilennummernanzeige (INDEX)
- Bildschirmoptimierung (OPTIMIZE)
- automatische Umnumerierung (RENUMBER)
- zwei Bildschirm-Arbeitsfenster (SPLIT)

Voreinstellen von Werten:

- Arbeitsdatei, für die @PAR gilt (fwkfv/GLOBAL)
- Satztrennzeichen (SEPARATOR)
- Voreinstellung der Codierungsart (CODE)
- Standardwert für die Angabe des Elementtyps eines Bibliothekselementes (ELEMENT-TYPE)
- Schrittweite zwischen zwei Zeilennummern (INCREMENT)
- Voreinstellung für eine Bibliothek (LIBRARY)
- maximale Satzlänge im F-Modus-Datenfenster (LIMIT)
- Struktursymbol für das Strukturblättern (STRUCTURE)
- Voreinstellung für einen Programmnamen (SDF-PROGRAM)

Operation	Operanden	F-Modus / L-Modus
@PAR	[fwkfv | **GL**OBAL] [,] [**E**DIT[[-] **LONG**] [=**ON**] | =**O**FF] [,] [**HEX** [=**ON**] | =**O**FF] [,] [**LOW**ER [=**ON**] | =**O**FF] [,] [**E**DIT[-]**FULL** [=**ON**] | =**O**FF] [,] [**PROT**ECTION [=**ON**] | =**O**FF] [,] [**SC**ALE [=**ON**] | =**O**FF] [,] [**INF**ORMATION [=**ON**] | =**O**FF] [,] [**I**NDEX [=**ON**] | =**O**FF] [,] [**OPT**IMIZE [=**ON**] | =**O**FF] [,] [**REN**UMBER [=**ON**] | =**O**FF] [,] [**SPLIT** =n fwkfv | =**O**FF] [,] [**SEP**ARATOR ='char' | =**O**FF] [,] [**C**ODE =**E**BCDIC | =**I**SO] [,] [[**ELEMENT**] [-] **TYP**E =elemtyp | =***STD**] [,] [**INC**REMENT =inc] [,] [**L**IBRARY =path] [,] [**LIM**IT =cl] [,] [**STRUC**TURE ='char'] [,] [**SDF-PROG**RAM =name | =***NONE**]	

Für die Operanden EDIT LONG, HEX, LOWER, EDIT FULL, PROTECTION, SCALE und INFORMATION stimmt der Voreinstellwert nicht mit dem Standardwert überein. Der Standardwert für diese Operanden ist ON.

Werden keine Operanden angegeben, werden alle arbeitsdateispezifischen Werte so eingestellt, wie nach dem Aufruf des EDT.

Die Kommas vor den Operanden werden nur angegeben, wenn 2 oder mehr Operanden in einer @PAR-Anweisung angegeben werden sollen.

fwkfv Die Arbeitsdateivariable ($0-$9) gibt an, auf welche Arbeitsdatei (0-9) sich @PAR bezieht.
SPLIT, OPTIMIZE, RENUMBER und SEPARATOR wirken immer auf alle 10 Arbeitsdateien (0-9). Für SPLIT wird die Angabe von fwkfv ignoriert. Ohne die Angabe einer Arbeitsdateivariablen gilt @PAR für die aktuelle Arbeitsdatei.

GLOBAL GLOBAL bezieht sich für RENUMBER, ELEMENT-TYPE, INCREMENT, CODE, LOWER, LIBRARY und LIMIT auf die Arbeitsdateien 0 bis 22. Für die anderen Operanden bezieht sich GLOBAL auf die Arbeitsdateien 0 bis 9, da diese nur im F-Modus Bedeutung haben. Für den Operanden SPLIT und SDF-PROGRAM hat GLOBAL keine Bedeutung.

Wird fwkfv oder GLOBAL als Operand angegeben, muß dieser als erster angegeben werden.

EDIT LONG Bestimmt, ob Sätze mit mehr als 80 Zeichen (DSS 3270: 77 Zeichen) vollständig im Arbeitsfenster dargestellt werden.

=ON Die Sätze werden vollständig im Arbeitsfenster dargestellt (maximal 256 Zeichen).

=OFF Es werden maximal 80 Zeichen (DSS 3270: 77 Zeichen) im Arbeitsfenster dargestellt. Nach dem EDT-Aufruf ist der Wert OFF voreingestellt.

HEX Ein- und Ausschalten des Hexadezimal-Modus. Im Hexadezimal-Modus werden pro Datensatz 4 Bildschirmzeilen ausgegeben. Die erste Zeile enthält die Zeichen in der Form, in der sie am Bildschirm abbildbar sind, oder Schmierzeichen. Die Zeilen 2 und 3 enthalten den Hexadezimal-Code der Zeichen von Zeile 1. Der Hexadezimal-Code steht vertikal unterhalb des Zeichens. In Zeile 4 wird eine Spaltenzählerzeile als Trennzeile zwischen den Datensätzen ausgegeben.

=ON Einschalten des Hexadezimal-Modus.
Bei @PAR CODE=ISO wird der Hexadezimalwert der EDT-Daten in ISO-Codierung (ASCII) ausgegeben. Die Eingaben müssen in den Hexadezimalzeilen in ISO-Codierung eingegeben werden.

=OFF Ausschalten des Hexadezimal-Modus.

Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

LOWER Legt fest, ob der EDT eingegebene Kleinbuchstaben in Großbuchstaben umsetzt oder nicht.

=ON Der EDT unterscheidet zwischen Groß- und Kleinbuchstaben. Text und Zeichenfolgen werden verarbeitet, wie sie eingegeben wurden.

=OFF Der EDT setzt eingegebene Kleinbuchstaben in Großbuchstaben um. Kleinbuchstaben in der Datei werden bei der Ausgabe am Bildschirm als Schmierzeichen dargestellt. Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

Nähere Information siehe @LOWER.

EDIT FULL Bestimmt, ob das Datenfenster und die Markierungsspalte gleichzeitig auf überschreibbar gestellt werden sollen. Der Überschreibmodus ist nur wirksam mit eingeschalteter Zeilennummernanzeige (@PAR INDEX=ON).

=ON Das Datenfenster und die Markierungsspalte sind gleichzeitig auf überschreibbar gestellt. Es ist möglich, eine Zeile zu markieren und gleichzeitig Daten in dieser Zeile zu ändern. Dadurch kann z.B. auch durch eine O-Markierung auf eine noch nicht angelegte Datenzeile kopiert werden.

Durch den Wechsel in den Hexadezimal-Modus (@PAR HEX=ON), in den EDIT LONG-Modus (@PAR EDIT LONG=ON) oder Ausschalten der Zeilennummernanzeige (@PAR INDEX=OFF) wird diese Einstellung nicht zurückgenommen, sondern nur inaktiviert.

Solange der Schreibschutz (@PAR PROTECTION=ON) eingestellt ist, wird die Eingabe von @PAR EDIT FULL=ON ignoriert.

=OFF Standardverarbeitung. Es kann in einer Bildschirmzeile entweder die Markierungsspalte oder der Datenteil beschrieben werden.

PROTECTION Einschalten des Schreibschutzes auf Satzebene. Dieser Operand ist nur im Zusammenhang mit der Anwendung der EDT-Unterprogrammschnittstelle sinnvoll. Dann können vom Anwenderprogramm aus Sätze durch Markierung schreibgeschützt oder überschreibbar dargestellt werden (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]).

=ON Es werden entsprechend gekennzeichnete Sätze im F-Modus Dialog schreibgeschützt dargestellt, bzw. anders gekennzeichnete Sätze im F-Modus-Dialog automatisch auf überschreibbar gestellt.
Ein eventuell eingestelltes EDIT FULL wird zurückgesetzt.

=OFF Die vom Anwenderprogramm angegebene Voreinstellung (Schreibschutz oder Überschreibbarkeit) ist nicht wirksam.
Nach Aufruf des EDT ist der Wert OFF voreingestellt.

SCALE Ausgeben eines Spaltenzählers (Zeilenlineal) im Datenfenster (gilt nicht im EDIT LONG-Modus).

=ON Der Spaltenzähler erscheint als 1. Zeile nach einer eventuell vorhandenen Informationszeile und gibt die aktuellen Spaltennummern des Arbeitsfensters an (z.B. nach horizontalem Verschieben des Arbeitsfensters). Falls ein Tabulator definiert ist (siehe @TABS), wird eine weitere Bildschirmzeile ausgegeben, in der die aktuellen Positionen des Tabulators mit „I“ angezeigt werden.Ist ein Tabulatorzeichen definiert (siehe @TABS), so wird es in der Markierungsspalten-Position abgebildet.

=OFF Ausschalten des Spaltenzählers und des eventuell vorhandenen Tabulator-Anzeigelineals.Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

INFORMATION

Ausgeben einer Informationszeile im Datenfenster (gilt nicht im EDIT-LONG-Modus).

Es wird entweder ein lokaler @FILE-Eintrag ausgegeben
- explizit definiert durch @FILE, oder
- implizit definiert durch @READ, @GET, @OPEN

oder ein Bibliotheks- und Elementname eines mit
- @OPEN (Format 2) eröffneten Bibliothekselements

oder ein POSIX-Dateiname einer mit
- @XOPEN eröffneten POSIX-Datei. Wenn der Dateiname zu lang ist, wird er am Anfang mit ... abgekürzt.

oder eine Überschriftszeile in Arbeitsdatei 9, falls
- @FSTAT LONG, @STAJV LONG oder @SHOW ohne Zielangabe im F-Modus-Dialog gegeben wird und der Inhalt nicht geändert wurde.

=ON Die Informationszeile erscheint als erste Zeile im Arbeitsfenster noch vor einem eventuell eingeschaltetem Zeilenlineal.
Falls keiner der oben genannten Namen und auch keine Überschriftszeile definiert ist, bleibt das Namensfeld leer.

=OFF Ausschalten der Informationszeile.

Nach dem Aufruf des EDT ist der Wert OFF voreingestellt.

INDEX Aus- und Einschalten der Zeilennummernanzeige. Dabei wird das Format des Bildschirmes verändert.

=ON Einschalten der Zeilennummernanzeige (EDT Standardformat).

Im Arbeitsfenster werden die 6-stellige Zeilennummernanzeige, ein Leerzeichen und 72 Zeichen (DSS 3270: 69 Zeichen) ausgegeben.

=OFF Ausschalten der Zeilennummernanzeige. Im Arbeitsfenster werden 80 Zeichen (DSS 3270: 77 Zeichen) pro Datenzeile ausgegeben.
Die erste Spalte jeder Zeile ist überschreibbar (Markierungsspalte).

Durch @PAR INDEX=ON wird der EDIT LONG-Modus ausgeschaltet.

Da bei der DSS 3270 die Feldtrennzeichen einen Platz am Bildschirm belegen, ist der für Daten nutzbare Teil entsprechend kürzer.

OPTIMIZE Ein- und Ausschalten der Bildschirmausgabeoptimierung. Vor jeder Bildschirmausgabe vergleicht der EDT den auszugebenden mit dem alten Bildschirm. Standardmäßig gibt er aus Gründen der Ausgabeverbesserung (Optimierung) nur den veränderten Text aus. Der unveränderte Text im alten Bildschirm bleibt unangetastet erhalten.

=ON Es werden nur geänderte Zeileninhalte ausgegeben (Standardwert)

=OFF Der gesamte Arbeitsfensterinhalt wird ausgegeben.

RENUMBER Ein- und Ausschalten der automatischen Umnumerierung der Zeilennummern. Die Anweisung wird wirksam bei @OPEN und @COPY, Format 2 sowie den Kurzanweisungen C, M und R. Siehe @COPY und @OPEN „Berechnung der Zeilennummern“.

=ON Die Zeilennummern einer Arbeitsdatei werden bei Bedarf umnumeriert (Standardwert). Dies führt der EDT aus, wenn beim Einlesen einer Datei (eines Bibliothekselementes) bzw. beim Kopieren oder Einfügen in eine Datei die Schrittweite nicht klein genug ist, um die Anweisung korrekt auszuführen.

=OFF Ausschalten der automatischen Umnumerierung. Die Zeilennummern einer Datei werden von EDT nicht verändert. Es erfolgt eine Meldung des EDT, wenn @OPEN oder @COPY nicht ausgeführt werden kann, weil die Schrittweite nicht klein genug ist, um alle Sätze aufnehmen zu können.

SPLIT Mit SPLIT kann ein zweites Arbeitsfenster am Bildschirm ausgegeben werden. Die Datei, in der diese Anweisung eingegeben wurde (aktuelle Arbeitsdatei), wird im oberen Arbeitsfenster gezeigt. Die mit fwkfv angegebene Arbeitsdatei wird im unteren Arbeitsfenster gezeigt.

=n fwkfv Ausgabe einer zweiten Arbeitsdatei mit folgenden Angaben:
n: Zeilenzahl ($2 \leq n \leq 22$) der unteren, zweiten Arbeitsdatei.
fwkfv: Arbeitsdateivariable ($0 bis $9), die angibt, welche Arbeitsdatei im unteren Teil des Bildschirmes gezeigt wird.
Die Dateiposition (Zeilen-, Spaltennummer) in dieser dort eingestellten Arbeitsdatei bleibt bei @PAR erhalten. Im weiteren Verlauf des EDT-Dialogs können die Dateipositionen der beiden Arbeitsdateien unabhängig verändert werden.

=OFF Ausblenden der zweiten Arbeitsdatei und Rücksetzen des verbleibenden Bildschirmes auf 24 Zeilen (Standard-Arbeitsfenster). Es bleibt der Bildschirm mit 24 Zeilen erhalten, in dessen Anweisungszeile die Anweisung gegeben wird. Wird @PAR SPLIT = OFF im oberen Arbeitsfenster eingegeben und stehen im unteren Arbeitsfenster Anweisungen, wird @PAR SPLIT mit einer Fehlermeldung abgelehnt.

SEPARATOR Festlegen des Satztrennzeichens zum „Auftrennen eines Datensatzes“ (siehe Seite 107).

=’char’ Das Satztrennzeichen char ist ein beliebiges alphanumerisches Zeichen oder Sonderzeichen. Es muß in Hochkomma angegeben werden. Die Hochkomma können durch @QUOTE umdefiniert werden.
Für Satztrennzeichen und Tabulatorzeichen sind unterschiedliche Zeichen zu wählen.

=OFF Aufheben einer Satztrennzeichen-Definition. Der Standardwert ist @PAR SEPARATOR = OFF, d.h. es ist kein Satztrennzeichen definiert.

CODE Voreinstellung der Codierungsart.
Für die Anweisungen @XOPEN, @XCOPY, @XWRITE wird der Standardwert für den Operanden CODE festgelegt. Im Hexadezimalmodus (@PAR HEX=ON bzw. HEX ON)) werden die EDT-Daten im Datenfenster entsprechend dieser Voreinstellung dargestellt.
Nach Aufruf des EDTs ist der Wert EBCDIC voreingestellt.

=EBCDIC Bei der Hexadezimal-Ausgabe wird der Sedezimalwert der EDT-Daten in EBCDIC-Codierung im Datenfenster ausgegeben.
Für die Anweisungen @XOPEN, @XCOPY, @XWRITE wird EBCDIC als Standardwert für den Operanden CODE festgelegt.

=ISO Bei der Hexadezimalausgabe wird der Sedezimalwert der EDT-Daten in ISO-Codierung im Datenfenster ausgegeben. Diese Codierung enspricht dem ASCII-Code. Die Darstellung zeigt, wie die Daten in POSIX-Dateien abgelegt werden, wenn sie mit @XWRITE (CODE=ISO) zurückgeschrieben werden.
Für die Anweisungen @XOPEN, @XCOPY, @XWRITE wird ISO als Standardwert für den Operanden CODE festgelegt.

ELEMENT-TYPE
Voreinstellung des Elementtyps.
Auf Elemente dieses Typs wird zugegriffen, wenn bei @COPY, @OPEN, @DELETE, @WRITE, @INPUT (Format 2) kein Elementtyp angegeben wurde.

=elemtyp Zulässige Typangaben: S, M, P, J, D, X, R, C, H, L, U, F, *STD und freie Typnamen mit entsprechendem Basistyp. elemtyp kann in der Form .str-var angegeben werden.

Typ	Elementinhalt
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats
R	Bindemodule
C	Lademodule
H	von H-Assembler erzeugt
L	vom BINDER erzeugt
U	von IFG erzeugt
F	von IFG erzeugt

*STD
Typ S ist die Voreinstellung nach Aufruf des EDT.
Wurde die Voreinstellung geändert, kann mit @PAR ELEMENT-TYPE = *STD die Voreinstellung wiederhergestellt werden.

INCREMENT

=inc Für @OPEN Format 2, @COPY Format 2 und @INPUT Format 2 wird der Wert für die Schrittweite zwischen den Datensätzen festgelegt. Im F-Modus wird die Schrittweite der Zeilennumerierung von nicht mit Daten belegten Bildschirmzeilen festgelegt. Der Standardwert für die Schrittweite ist 1.

Bei @PAR INCREMENT mit einer Schrittweite <0.01 ist zu beachten, daß im F-Modus Zeilennummern von eingelesenen, kopierten oder eingefügten Zeilen nicht vollständig ausgegeben werden (6-stellige Zeilennummernanzeige). Werden diese unvollständig ausgegebenen Zeilennummern in Anweisungen verwendet (@COPY, usw.), kann dies zu Fehlern führen.

LIBRARY

=path Voreinstellung eines Bibliotheknamens.
Auf die Elemente dieser Bibliothek wird zugegriffen, wenn bei @COPY, @OPEN, @WRITE, @INPUT nur der Elementname angegeben wurde. Falls bei @SHOW kein Bibliotheksname angegeben ist, wird das Inhaltsverzeichnis dieser PLAM-Bibliothek ausgegeben.
path kann in der Form .str-var angegeben werden.

LIMIT

=cl Dient zur Definition der maximalen Satzlänge im F-Modus-Datenfenster. Wird ein Satz eingegeben, dessen Länge den angegebenen Limitwert überschreitet, wird dieser Satz abgeschnitten und die Meldung % EDT2267 LINE TRUNCATED AFTER nnn CHARACTERS ausgegeben. Indirekte Satzänderungen durch Anweisungen etc. sind von dieser Prüfung nicht betroffen. Der zulässige Wertebereich für cl ist 1..256. Der Standardwert für cl ist 256.

STRUCTURE

='char' Das Struktursymbol ist ein Fluchtsymbol für das Strukturblättern (siehe Abschnitt „+ / – Positionieren des Arbeitsfensters nach der Strukturtiefe“ auf Seite 102ff.). Es muß in Hochkommas angegeben werden. Die Hochkommas können durch @QUOTE umdefiniert werden. Dieses Symbol kennzeichnet Sätze, die beim Strukturblättern ausgewertet werden sollen.
Ist ein Struktursymbol ungleich Leerzeichen definiert, werden nur die Sätze ausgewertet, die mindestens dieses Struktursymbol enthalten. Wird für das Struktursymbol ein Leerzeichen angegeben, so werden alle Sätze ausgewertet.
Der Standardwert des Struktursymbols ist '@' (z.B. für Columbus-Quellprogramme).

SDF-PROGRAM

Es wird für die Anweisung @SDFTEST und die Kurzanweisung t der Name eines Programms vordefiniert.
Datenzeilen, die mit // beginnen, werden als Anweisungen dieses Programms betrachtet.

Der interne Programmname kann mit SDF-A ermittelt werden, falls er nicht dem Namen des Programms entspricht wie z.B. bei C, BINDER, LMS, SORT:

ASSEMBH	INTERNAL-NAME=ASS$XT
PASCAL-XT	INTERNAL-NAME=PASXT21A
SDF-A	INTERNAL-NAME=SDAEDXT
SDF-U	INTERNAL-NAME=CMDEDIT

=name Interner Name eines Programms, maximal 8 stellig.
Der Name gilt für alle Arbeitsdateien global.

*NONE Mit *NONE wird die Definition zurückgenommen.
*NONE ist die Voreinstellung bei Start des EDTs.

@PARAMS Definieren von EDT-Parametern

@PARAMS definiert alle symbolischen Parameter, die innerhalb einer Prozedur benutzt werden.

Parameter in EDT-Prozeduren

Die Parameter können als Zeichenvariable betrachtet werden, die vor dem Ausführen einer EDT-Prozedur durch deren Wert ersetzt werden.

Ein Parameter beginnt mit dem Zeichen &. Ihm folgt ein Buchstabe, dem bis zu sechs weitere Buchstaben oder Ziffern folgen können. Es dürfen auch Kleinbuchstaben verwendet werden. Dabei ist aber zu beachten, daß der EDT zwischen Groß- und Kleinbuchstaben unterscheidet. &A und &a sind also zwei verschiedene Parameter. Die in einer Arbeitsdatei verwendeten Parameternamen gelten nur innerhalb dieser Arbeitsdatei.

EDT-Parameter werden mit

– @PARAMS in der ersten Zeile einer EDT-Prozedur definiert,
– @DO beim Aufruf der Prozedur mit Werten versorgt.

Man unterscheidet Stellungsparameter und Schlüsselwortparameter.

Stellungsparameter erhalten der Reihe nach die aktuellen Werte aus dem @DO-Aufruf (siehe @DO).

In Schlüsselwortparametern steht hinter dem Parameternamen ein Gleichheitszeichen. Dem Gleichheitszeichen folgt der Parameterwert. Schlüsselwortparameter erhalten den aktuellen Wert ebenfalls aus der Parameterliste im @DO-Aufruf. Falls er dort fehlt, wird der vordefinierte Anfangswert aus @PARAMS eingesetzt. Der Wert des Parameters ergibt sich aus allen angegebenen Zeichen, einschließlich der Leerzeichen. Wird für einen Stellungsparameter oder einen nicht vorbelegten Schlüsselwortparameter kein Wert angegeben, so erhält er den Wert leere Zeichenfolge (Leerstring).

Enthält ein Parameterwert Kommas oder schließende Klammern, muß er in Hochkommas eingeschlossen werden. Um einfache Hochkommas innerhalb von Hochkommas zu erzeugen, müssen paarige Hochkommas im Parameterwert angegeben werden (siehe auch Beispiel). Wird zuvor mit @QUOTE einem anderen Zeichen die Funktion des Hochkommas zugeordnet, gilt dies nicht für die den Parameterwert einschließenden Hochkommas.

Beispiel

Zeile in Prozedurdatei	Parametereingabe	erzeugte Zeile
@ON &F‘&SEARCH‘ @SET #S1=&STR &DATA @P RANGE	&SEARCH=A“B STR=␣‘TEXT‘ &DATA=‘A‘B &RANGE=‘3,7‘	@ON &F‘A“B‘ @SET #S1=␣‘TEXT‘ ‘A‘B @P 3,7

Operation	Operanden	@PROC
@**PAR**AMS	formal [,...]	

formal — Formaler (symbolischer) Parameter.

Der EDT ignoriert in @PARAMS Leerzeichen, die
- in, vor oder nach einem formalen Stellungsparameter,
- an beliebiger Stelle vor dem Gleichheitszeichen in einem formalen Schlüsselwortparameter auftreten.

Beispielsweise ist @PARAMS & A B C, &ZEILE= gleichwertig mit @PARAMS &ABC,&ZEILE=

Dies gilt nur bei @PARAMS. Bei @DO sind keine Leerzeichen im Parameternamen erlaubt.
Soll einem Schlüsselwortparameter kein Anfangswert zugewiesen werden, ist unmittelbar nach dem Gleichheitszeichen ein Komma anzugeben oder @PARAMS zu beenden.

,... — Deutet an, daß mehrere durch Komma voneinander getrennte formale Parameter angegeben werden dürfen. Werden sowohl Stellungs- als auch Schlüsselwortparameter angegeben, sind die Stellungsparameter zuerst anzuführen.
Die Stellungsparameter müssen bei @PARAMS und bei @DO in der gleichen Reihenfolge angegeben sein. Die Reihenfolge der Schlüsselwortparameter kann bei @PARAMS und bei @DO verschieden sein.
Die Anzahl der in @PARAMS möglichen formalen Parameter ist durch die maximale Länge einer EDT-Anweisung von 256 Zeichen beschränkt.

Werden in einer Prozedur Parameter verwendet, muß @PARAMS die erste Zeile bilden. Weitere @PARAMS werden ignoriert.

Ersetzen der Parameter durch ihre Werte

Die bei @DO angegebenen Parameterwerte werden den in der Prozedur stehenden formalen Parametern über @PARAMS zugewiesen. Fehlt @PARAMS, werden die formalen Parameter nicht mit aktuellen Werten versorgt. Das gleiche gilt für die Parameter, die bei @PARAMS nicht aufgeführt sind. Wird ein Schlüsselwortparameter bei @DO nicht angegeben, erhält er in der Prozedur den Wert, der als Anfangswert bei @PARAMS steht.

Die Parameter können innerhalb der Prozedur an beliebiger Stelle verwendet und mit anderen Zeichenfolgen und Parametern verkettet werden.

Steht zwischen dem formalen Parameter und der nachfolgenden Zeichenfolge bzw. dem nachfolgenden formalen Parameter ein Punkt, erscheint dieser Punkt nicht im Ergebnis der Verkettung. Der EDT interpretiert ihn als Information, daß eine Verkettung stattfinden soll. Soll ein Parameterwert mit einer nachfolgenden Zeichenfolge verkettet werden, die mit einem Buchstaben, einer Ziffer oder einem Punkt beginnt, muß der Punkt zwischen dem formalen Parameter und dieser Zeichenfolge unbedingt angegeben werden.

In den nachfolgenden Beispielen wird angenommen, daß der Parameter @PARAM den Wert A hat.

Darstellung in der Prozedurzeile	Generierte Zeichen
&PARAM(BC)	A(BC)
&PARAM.(BC)	A(BC)
&PARAM..(BC)	A.(BC)
&PARAM..BC	A.BC
&PARAM.2BC	A2BC
&PARAM,.2B	A,.2B
BC&PARAM	BCA
BC,&PARAM	BC,A
B2&PARAM	B2A
&PARAM.&PARAM	AA
&PARAM&PARAM	AA
&PARAM..&PARAM	A.A

Soll ein bei @PARAMS angeführter formaler Parameter ausnahmsweise in der Prozedurdatei nicht durch den aktuellen Parameterwert ersetzt werden, ist das Zeichen & doppelt anzugeben. Beim Ablauf der Prozedur wird eines der beiden & entfernt.

Beispiel

@PARAMS &DRUCKER=

&&DRUCKER=&DRUCKER

@DO...(DRUCKER=L2) erzeugt in der Prozedurdatei eine Zeile mit folgendem Inhalt:

&DRUCKER = L2

Wird in der Prozedur ein formaler Parameter durch den aktuellen Parameterwert ersetzt, kann dies zur Folge haben, daß die Zeile länger als 256 Zeichen wird. Das führt zu einer Fehlermeldung bei der Ausführung der Prozedur.

Beispiel 1

```
   7.     @PRINT
   1.0000 WAS JETZT
   2.0000 AUSGEGEBEN
   3.0000 WERDEN SOLL,
   4.0000 WIRD ERST
   5.0000 IM @DO-KOMMANDO
   6.0000 ENTSCHIEDEN
   7.     @SET #S3 = '*** SO IST ES ***'
   7.     @PROC 1
   1.     @ @PARAMS &ZEILEN ------------------------------------------- (01)
   2.     @ @PRINT &ZEILEN ------------------------------------------- (02)
   3.     @END
   7.     @DO 1(2-4) ------------------------------------------------- (03)
   2.0000 AUSGEGEBEN
   3.0000 WERDEN SOLL,
   4.0000 WIRD ERST
   7.     @DO 1(#S3),PRINT
   7.     @PRINT #S3 ------------------------------------------------- (04)
     #S03 *** SO IST ES ***
   7.     @DO 1(2,4N) ------------------------------------------------ (05)
 %  EDT4963 TOO MANY OPERANDS
   7.     @DO 1('2,4N') ---------------------------------------------- (06)
   2.0000 AUSGEGEBEN
WIRD ERST
   7.
```

(01) Innerhalb der Arbeitsdatei 1 wird in der ersten Zeile der Stellungsparameter &ZEILEN vereinbart.

(02) Dieser Parameter taucht innerhalb von @PRINT auf. Welche Zeilen nun ausgegeben werden sollen, hängt von dem in der @DO-Anweisung genannten Parameterwert ab.

(03) Die Arbeitsdatei 1 wird ausgeführt. Vor der Ausführung der einzelnen Anweisung wird jedoch dem Parameter &ZEILEN der Wertbereich 2-4 zugeordnet.

(04) Besonders deutlich kommt das Einsetzen des Parameterwertes zum Ausdruck, wenn man sich die einzelnen Prozeduranweisungen vor ihrer Ausführung auf dem Bildschirm ausgeben läßt, weil hier bereits das Einsetzen des Parameterwertes vorgenommen wurde.

(05) Versucht man beispielsweise, die Zeile 2 mit Zeilennummer und die Zeile 4 ohne Zeilennummer auszugeben, wird ein Komma, das Bestandteil des Parameterwertes ist, als Trennzeichen zweier Parameter interpretiert und dieses @DO damit abgelehnt.

(06) Ein Parameterwert läßt sich auch innerhalb von Hochkommas übergeben. Hierbei wird dem Parameter &ZEILEN der Wert zugeordnet, der zwischen den Hochkommas steht. Damit können auch Kommas als Bestandteil eines Parameterwertes übergeben werden.

Beispiel 2

```
   1.     @PROC 2
   1.     @ @PARAMS &STRVAR1,&STRVAR2,&INHALT1=**** ---------------------- (01)
   2.     @ @SET &STRVAR1 = '&INHALT1'
   3.     @ @SET #S2 = '&STRVAR1'
   4.     @ @SET #S3 = &STRVAR1
   5.     @ @SET &STRVAR2 = &STRVAR1
   6.     @ @SET #S4 = 'VON &STRVAR1 BIS &STRVAR2'
   7.     @ @PRINT &STRVAR1,&STRVAR2,#S2,#S3,#S4
   8.     @END
   1.     @DO 2(#S0,#S1) ------------------------------------------------ (02)
     #S00 ****
     #S01 ****
     #S02 #S0
     #S03 ****
     #S04 VON #S0 BIS #S1
   1.     @DO 2(#S15,#S13,INHALT1=AUWEIA) ------------------------------- (03)
     #S15 AUWEIA
     #S13 AUWEIA
     #S02 #S15
     #S03 AUWEIA
     #S04 VON #S15 BIS #S13
1.
```

(01) 2 Stellungs- und 1 Schlüsselwortparameter werden für die Arbeitsdatei 2 definiert.

(02) Die Werte für die Stellungsparameter müssen bei @DO in der Reihenfolge angegeben werden, die der Reihenfolge der Stellungsparameter in der @PARAMS-Zeile entspricht. Hierbei wird bei Durchführung der Arbeitsdatei 2 für &STRVAR1 der Wert #S0 eingesetzt, für &STRVAR2 der Wert #S1. Da kein Wert für den Schlüsselwortparameter &INHALT1 angegeben ist, wird bei Durchführung der Standardwert - also **** - angenommen.

(03) Wird für einen Schlüsselwortparameter ein Parameterwert bei @DO angegeben, wird dadurch der Standardwert ersetzt.

Beispiel 3

```
1.     @PROC 3
1.     @ @PARAMS &A,&B,&C,&X=111,&Y=222,&Z=333 ---------------------- (01)
2.     @ @CREATE #S10: '&A','&B','&C','&X','&Y','&Z'
3.     @ @PRINT #S10
4.     @END
1.     @DO 3 (AAAAA,BB,CCCCCCC) ------------------------------------- (02)
  #S10 AAAAABBCCCCCCC111222333
1.     @DO 3 (AA,BBBB,C,Y=****,X=########) --------------------------- (03)
  #S10 AABBBBC########****333
1.
```

(01) Innerhalb der Arbeitsdatei 3 werden 3 Stellungs- und 3 Schlüsselwortparameter definiert.

(02) Die Arbeitsdatei 3 wird ausgeführt. Da jedoch kein Wert für einen Schlüsselwortparameter angegeben ist, werden hierfür die Standardwerte angenommen.

(03) Nun werden auch Werte für 2 Schlüsselwortparameter angegeben. Man beachte, daß die Reihenfolge der angegebenen Werte für die Schlüsselwortparameter nicht übereinstimmt mit der Reihenfolge der Definition der Schlüsselwortparameter in der @PARAMS-Zeile.

@PREFIX Voranstellen von Zeichenfolgen

@PREFIX stellt jeder Zeile in dem angegebenen Bereich eine Zeichenfolge voran (siehe auch @SUFFIX, Anhängen von Zeichenfolgen an Zeilen).

Operation	Operanden	F-Modus / L-Modus
@**PRE**FIX	range **W**ITH string	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden. Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

string — Zeichenfolge, die vor jede Zeile des angegebenen Bereichs gestellt werden soll.
string kann angegeben werden:
- explizite Angabe in Hochkomma,
- implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Beispiel

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 EINMAL..................................................................
   5.00 EINMAL..................................................................
   6.00 .......................................................................




prefix 4-5 with ' NOCH ' .....................................0001.00:001(0)
```

Im Zeilenbereich von 4 bis 5 soll die Zeichenfolge NOCH vorangestellt werden.

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00  NOCH EINMAL...........................................................
   5.00  NOCH EINMAL...........................................................
   6.00 .....................................................................




prefix 4-5 with 1 ..............................................0001.00:001(0)
```

Im Zeilenbereich von 4 bis 5 soll der Inhalt der Zeile 1 vorangestellt werden.

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 UND NOCH EINMAL.......................................................
   5.00 UND NOCH EINMAL.......................................................
   6.00 .....................................................................




prefix 4-5 with ' '*5 ..........................................0001.00:001(0)
```

Im Zeilenbereich von 4 bis 5 soll jeweils fünf Leerzeichen vorangestellt werden.

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 UND NOCH EINMAL.......................................................
   5.00 UND NOCH EINMAL.......................................................
   6.00 .....................................................................




prefix 4-5 with 4 ..............................................0001.00:001(0)
```

Im Zeilenbereich von 4 bis 5 soll der Inhalt der Zeile 4 vorangestellt werden.

```
1.00 UND.....................................................................
2.00 NOCH....................................................................
3.00 EINMAL...................................................................
4.00      UND NOCH EINMAL     UND NOCH EINMAL...............................
5.00      UND NOCH EINMAL     UND NOCH EINMAL...............................
6.00 .....................................................................
```

@PRINT Zeilenbereiche bzw. Inhalte von Zeichenfolgevariablen ausgeben

Im L-Modus gibt @PRINT die Zeilen eines angegebenen Bereichs oder den Inhalt von Zeichenfolgevariablen aus. Im F-Modus kann nur der Inhalt von Zeichenfolgevariablen ausgegeben werden.

Im Dialog erfolgt die Ausgabe auf den Bildschirm (SYSOUT), bei Stapelverarbeitung auf den Drucker (SYSLST).

Operation	Operanden	F-Modus / L-Modus
@**P**RINT	[rng [:domain] [**X**] [**N**] [**S**] [**V** | **E**]] [,...]	

rng Zeilenbereich, bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

rng muß angegeben werden, wenn X, N, V, E oder S verwendet werden.

Wird rng nicht angegeben, wird die ganze Arbeitsdatei abschnittsweise (wie mit V) ausgegeben.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte der Rest der Zeile ausgegeben.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile ausgegeben.

X Die Ausgabe erfolgt in hexadezimaler Form. In diesem Fall können Zeilen nur bis zu maximal 128 Zeichen ausgegeben werden. Bei Verwendung von X muß rng angegeben werden.

N Unterdrückt die Zeilennummern bzw. die Bezeichnung der Zeichenfolgevariablen bei der Ausgabe. Es muß rng angegeben werden.

S Ist nur im Stapelbetrieb sinnvoll. Bei Angabe von S wird die erste Leerzeile unterdrückt, die normalerweise bei Ausgabe auf den Drucker der ersten auszugebenden Zeile vorangeht. Bei Verwendung von S muß rng angegeben werden.

V Ist nur beim Arbeiten am Bildschirm sinnvoll. V bewirkt, daß der EDT den angegebenen Zeilenbereich abschnittsweise ausgibt. Nach Ausgabe des ersten Abschnittes wird auf jeden Fall eine Blättereingabe angefordert, auch wenn die letzte Zeile des angegebenen Zeilenbereichs schon ausgegeben wurde. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet EDT wie bei Eingabe von +.

Aus wieviel physikalischen Zeilen (Bildschirmzeilen) ein Abschnitt besteht, hängt von der benutzten Datensichtstation ab. Der vom System vorgegebene Wert (Bildschirmgröße) kann mit @VDT verändert werden.

E Wie V, aber mit folgendem Unterschied:
Enthält der erste Abschnitt die letzte Zeile des angegebenen Zeilenbereichs, so wird nach Ausgabe dieser Zeile die @PRINT-Funktion beendet. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet EDT wie bei Eingabe von *.

Ausgabe des Arbeitsdateiinhalts

Wird rng angegeben, wird der komplette Bereich ausgegeben. Die Ausgabe wird nur unterbrochen, wenn die Überlaufkontrolle des Betriebssystems (MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL = USER-ACKNOWLEDGE) eingeschaltet ist.
Nach der Ausgabeunterbrechung durch %PLEASE ACKNOWLEDGE kann man durch K2 und RESUME-PROGRAM die Ausgabe des Zeilenbereichs abbrechen. Positionieren ist also nicht möglich.

Durch die Angabe von V oder E gibt der EDT, wie bei @PRINT ohne rng, den Bereich abschnittsweise aus. Es werden so viele Bildschirmzeilen ausgegeben, wie vom System oder durch @VDT eingestellt ist.
Wird bei @PRINT ohne rng weder V noch E angegeben, wird wie mit Angabe V gearbeitet. Nach der Ausgabe des Dateiabschnitts fordert der EDT eine der Eingaben *, +, – oder 0, mit der abgebrochen oder positioniert werden kann.

Das Verhalten bei Eingabe von nur DUE ist abhänig vom Operanden V oder E, bzw. von der letzten vorangehenden Blättereingabe *, + oder +int.

*	Der EDT gibt den Dateiabschnitt aus, der dem zuletzt ausgegebenen unmittelbar folgt. Enthält der auszugebende Dateiabschnitt die letzte Zeile des angegebenen Bereichs, wird nach Ausgabe dieser Zeile die @PRINT-Funktion beendet. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet der EDT wie bei Eingabe von *. Wurde im zuletzt ausgegebenen Dateiabschnitt die Ober- oder Untergrenze des angegebenen Zeilenbereichs über- bzw. unterschritten, bewirkt * die sofortige Beendigung der @PRINT-Funktion, ohne daß eine Ausgabe erfolgt.
+	Der EDT gibt den Dateiabschnitt aus, der dem zuletzt ausgegebenen unmittelbar folgt. Ein Dateiabschnitt, dessen Ausgabe durch + bewirkt wird, kann teilweise oder ganz außerhalb des angegebenen Zeilenbereichs liegen. Obergrenze ist das Dateiende. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet der EDT wie bei Eingabe von +.
+int	Der EDT gibt den Dateiabschnitt aus, der in der Datei int Zeilen hinter der letzten ausgegebenen Zeile beginnt. Ein Dateiabschnitt, dessen Ausgabe durch +int bewirkt wird, kann teilweise oder ganz außerhalb des angegebenen Zeilenbereichs liegen. Obergrenze ist das Dateiende. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet der EDT wie bei Eingabe von +.
–	Der EDT gibt den Dateiabschnitt aus, der unmittelbar vor dem zuletzt ausgegebenen steht. Ein Dateiabschnitt, dessen Ausgabe durch – bewirkt wird, kann teilweise oder ganz außerhalb des angegebenen Zeilenbereichs liegen. Untergrenze ist der Dateianfang. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet der EDT wie bei Eingabe von +.
–int	Der EDT gibt den Dateiabschnitt aus, der in der Datei int Zeilen vor der ersten Zeile des zuletzt ausgegebenen Dateiabschnitts beginnt. Ein Dateiabschnitt, dessen Ausgabe durch –int bewirkt wird, kann teilweise oder ganz außerhalb des angegebenen Zeilenbereichs liegen. Untergrenze ist der Dateianfang. Bei nachfolgenden leeren Blättereingaben (nur DUE) arbeitet der EDT wie bei Eingabe von +.
0	Beendet die Ausgabe des Zeilenbereichs.

Eine Zeile der aktuellen Arbeitsdatei, die sich bei der Ausgabe über mehrere Bildschirmzeilen erstreckt, teilt der EDT nicht auf mehrere Bildschirmausgaben auf. Das kann dazu führen, daß eine Bildschirmausgabe weniger Bildschirmzeilen enthält, als maximal möglich sind.

Wurde der EDT als selbständiges Programm mit START-PROGRAM oder LOAD-PROGRAM aufgerufen, kann der Benutzer statt *, +, +int, –, –int oder 0 auch eine Anweisung eingeben. Dies bewirkt, daß die @PRINT-Funktion beendet und anschließend die Anweisung ausgeführt wird. Selbst bei eingeschaltetem Blockmodus ist in diesem Fall aber die Eingabe einer Anweisungsfolge nicht zulässig. Wurde der EDT als Unterprogramm aufgerufen, akzeptiert der EDT bei noch nicht beendeter @PRINT-Funktion eine Anweisung nur dann, wenn im Funktionsbyte 2 der Parameterliste das Bit 2^0 nicht gesetzt ist (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]).

@PROC Umschalten von Arbeitsdateien

@PROC bietet in 2 Formaten im L-Modus folgende Funktionen:

– Umschalten in eine andere Arbeitsdatei (Format 1),

– Informationen über die freien und belegten Arbeitsdateien bzw. über die aktuelle Arbeitsdatei anzeigen (Format 2).

@PROC (Format 1) Umschalten von Arbeitsdateien

Mit diesem Format von @PROC kann man im L-Modus in eine andere Arbeitsdatei umschalten.

Operation	Operanden	L-Modus / @PROC
@**PRO**C	procnr [comment]	

procnr Nummer einer Arbeitsdatei, in der EDT-Prozeduren ablaufen können (1 bis 22) oder eine Ganzzahlvariable, die einen dieser Werte enthält.

comment Beliebiger Kommentar.
Damit ist auch bei strenger Syntaxkontrolle (@SYNTAX) ein Kommentar möglich.

Die Arbeitsdatei, in die durch dieses Anweisungsformat gewechselt wurde, bleibt solange aktuell bis

– mit @END in die zuletzt aktuelle Arbeitsdatei zurückgekehrt wird,

– mit einem weiteren @PROC oder @SETF (procnr) erneut in eine andere Arbeitsdatei gewechselt wird.

Mit @PROC wird in eine andere Arbeitsdatei gewechselt, ohne die darunter liegende(n) zu beenden (geschachtelte Arbeitsdateien). Im Gegensatz dazu werden durch @SETF (procnr) vor dem Umschalten in die neue Arbeitsdatei die darunterliegenden Arbeitsdateien alle beendet.

Wird die Nummer einer leeren Arbeitsdatei angegeben, ist die aktuelle Zeilennummer 1 und die aktuelle Schrittweite 1.

Wird die Nummer einer Arbeitsdatei angegeben, die bereits Daten oder Anweisungen enthält, ist die aktuelle Zeilennummer und die aktuelle Schrittweite gleich der, die man beim letzten Verlassen dieser Arbeitsdatei hatte.

Beispiel 1

```
5.     @PRINT
1.0000 AAAA
2.0000 BBBAADAFD
3.0000 AAA
4.0000 CCCCCCCCCCCCCCCC
5.     @PROC 6
1.     @ @SET #I6 = LENGTH !
2.     @ @SET #L6 = !
3.     @ @DO 10
4.     @ @DO 12
5.     @ @CREATE #S6: 'ZEILE ',#S12,' IST ',#S10,' ZEICHEN LANG'
6.     @ @PRINT #S6 N
7.     @END ------------------------------------------------------- (01)
5.     @PROC 10 --------------------------------------------------- (02)
1.     @ @SET #S10 = CHAR #I6
2.     @ @ON #S10:2-2: DELETE '0'
3.     @ @IF .TRUE. GOTO 2
4.     @END
5.     @PROC 12 --------------------------------------------------- (03)
1.     @ @SET #S12 = CHAR #L6
2.     @END
5.     @DO 6 !=%,$
ZEILE    1.0000 IST  4 ZEICHEN LANG
ZEILE    2.0000 IST  9 ZEICHEN LANG
ZEILE    3.0000 IST  3 ZEICHEN LANG
ZEILE    4.0000 IST  17 ZEICHEN LANG
5.
```

(01) Es wird in die Arbeitsdatei 0 umgeschaltet. Die Arbeitsdatei 6 enthält 6 EDT-Anweisungen unter anderem ein @DO 10 und ein @DO 12. Zu diesem Zeitpunkt existieren aber diese beiden Arbeitsdateien noch nicht. Demnach würde ein jetzt gegebenes @DO 6 zu einem Fehler führen.

(02) Die Arbeitsdatei 10 wird definiert und hat die Aufgabe, den in #I6 festgehaltenen Wert in #S10 abdruckbar abzubilden und führende Nullen zu löschen.

(03) Die Arbeitsdatei 12 wird definiert und hat lediglich die Aufgabe, den Inhalt der Zeilennummervariablen #L6 in #S12 abdruckbar abzubilden.

Beispiel 2

```
1.     @SET #I4 = 1
1.     @PROC #I4 ------------------------------------------------- (01)
1.     @4.00
4.00   @ @SET #I4 = #I4 + 1 ------------------------------------- (02)
4.01   @ @IF #I4 > 4 GOTO 8
4.02   @ @PROC #I4
4.03   @ @PROC ------------------------------------------------- (03)
4.04   @ @GOTO 4
4.05   @8.00
8.00   @ @SET #I4 = #I4 - 1
8.01   @ @END -------------------------------------------------- (04)
8.02   @ @IF #I4 = 2 RETURN
8.03   @ @PROC ------------------------------------------------- (05)
8.04   @ @GOTO 8
8.05   @END
1.     @DO #I4 -------------------------------------------------- (06)
<02>
<03>
<04>
<03>
<02>
1.
```

(01) In eine Arbeitsdatei kann auch über eine Ganzzahlvariable umgeschaltet werden. Die Ganzzahl muß zwischen 1 und 22 liegen.

(02) Beim Ausführen der Arbeitsdatei 1 wird in die Arbeitsdateien 2 bis 4 umgeschaltet, wobei die jeweilige Arbeitsdateinummer immer über #I4 übergeben wird.

(03) Mit @PROC wird abgefragt, in welcher Arbeitsdatei man sich befindet. Somit wird beim Ausführen der Arbeitsdatei #I4 an dieser Stelle das jeweilige #I4 quittiert.

(04) Ein einziges @END wird verwendet, um in allen Fällen wieder in die Arbeitsdatei 0 zurückzuschalten.

(05) Nach dem Zuklappen wird wieder gefragt, in welcher Arbeitsdatei man sich befindet. Somit wird sich eine absteigende Sequenz der Arbeitsdateinummern 3 bis 2 ergeben.

(06) Auch das Anstoßen einer Arbeitsdatei kann über eine Ganzzahlvariable erfolgen.

@PROC (Format 2) Ausgeben von Informationen

Mit diesem Format von @PROC kann man sich folgende Informationen ausgeben lassen:

– die Nummer der aktuellen Arbeitsdatei (@PROC),
– die Nummern aller freien Arbeitsdateien (@PROC FREE),
– die Nummern aller belegten Arbeitsdateien (@PROC USED).

Operation	Operanden	L-Modus / @PROC
@**PRO**C	[**F**REE | **U**SED]	

FREE Es werden die Nummern jener Arbeitsdateien (1-22) ausgegeben, die noch nicht belegt wurden.

USED Es werden die Nummern jener Arbeitsdateien (1-22) ausgegeben, die schon belegt sind. Zu jeder Nummer wird die niedrigste und höchste Zeilennummer ausgegeben.

Ist keine Arbeitsdatei außer der Arbeitsdatei 0 belegt, gibt der EDT eine Meldung aus.

Wird kein Operand angegeben, dann wird die Nummer der aktuellen Arbeitsdatei am Bildschirm angezeigt.

Beispiel 1

```
1.     @PROC ------------------------------------------------------ (01)
<00>
1.     @PROC 15 --------------------------------------------------- (02)
1.     @PROC ------------------------------------------------------ (03)
<15>
1.     @END -------------------------------------------------------- (04)
1.     @PROC ------------------------------------------------------ (05)
<00>
```

(01) Die Abfrage, in welcher Arbeitsdatei man gerade tätig ist, bringt die Nummer <00> also die Hauptebene.

(02) Es wird in die Arbeitsdatei 15 umgeschaltet.

(03) Natürlich ergibt sich bei Abfrage der Nummer der Arbeitsdatei ebenfalls 15.

(04) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(05) Man befindet sich jetzt wieder in der Arbeitsdatei 0.

Beispiel 2

```
   1.     @DROP ALL ------------------------------------------------- (01)
   1.     @PROC USED ------------------------------------------------ (02)
% EDT0907 NO PROCEDURE FILES DECLARED
   1.     @PROC 13
   1.     A
   2.     @END
   1.     @PROC USED ------------------------------------------------ (03)
   <13>     1.0000 TO    1.0000
```

(01) Alle Arbeitsdateien werden freigegeben.

(02) Es wird gefragt, welche Arbeitsdateien definiert sind.

(03) Nach Definition der Arbeitsdatei 13 wird wieder gefragt, welche Arbeitsdateien definiert sind.

Beispiel 3

```
   1.     @DROP ALL ------------------------------------------------- (01)
   1.     @PROC FREE ------------------------------------------------ (02)
% EDT0907 NO PROCEDURE FILES DECLARED
   1.     @PROC 13
   1.     A
   2.     @END
   1.     @PROC FREE ------------------------------------------------ (03)
   01 02 03 04 05 06 07 08 09 10 11 12 14 15 16 17 18 19 20 21 22
```

(01) Alle Arbeitsdateien werden freigegeben.

(02) Es wird gefragt, welche Arbeitsdateien noch nicht definiert sind.

(03) Nach Definition der Arbeitsdatei 13 wird wieder gefragt, welche Arbeitsdateien nicht definiert sind.

@QUOTE Begrenzersymbol für Zeichenfolgen umdefinieren

Wenn in einer Anweisung eine Zeichenfolge anzugeben ist (mit search, string, file usw.), ist diese in Hochkommas einzuschließen. @QUOTE definiert Zeichen, die diese Hochkommas ersetzen.

Operation	Operanden	F-Modus / L-Modus
@**Q**UOTE @**QE**	{ spec, char spec ,char }	

spec Sonderzeichen, ersetzt das Hochkomma (').

char Zeichen, ersetzt das Anführungszeichen (").

spec und char müssen verschieden sein.

Anführungszeichen finden ausschließlich bei @ON Anwendung, und zwar im Zusammenhang mit Textbegrenzern (siehe @DELIMIT und @ON). Hochkommas werden ebenfalls bei @ON angewandt, darüber hinaus aber auch in anderen Anweisungen.

Ist spec kein Sonderzeichen, wird @QUOTE mit der Fehlermeldung abgewiesen:
% EDT3952 INVALID SYMBOL

i Anwendungsfälle für diese Anweisung ergeben sich beispielsweise, wenn bei @ON eine Suchzeichenfolge aufgefunden werden soll, die ein Hochkomma oder ein Anführungszeichen enthält.

@RANGE Zeilenbereichssymbol definieren

@RANGE definiert ein neues Symbol für einen Zeilen- und Spaltenbereich. Das alte Zeilenbereichssymbol wird ungültig.

Operation	Operanden	F-Modus / L-Modus
@**RA**NGE	[=r =rng [:domain]]	

r Symbol für den zu vereinbarenden Bereich.

rng Zeilenbereich, bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol, durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25).

Dieser Spaltenbereich wirkt nur bei @ON.

Wird nur eine Spaltennummer angegeben, erstreckt sich der Spaltenbereich von dieser Spaltennummer bis zum Ende der Zeile. Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt. Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird domain nicht angegeben, wird der Spaltenbereich 1-256 angenommen.

Es existiert immer nur ein Bereichssymbol. Das standardmäßige Bereichssymbol ist das &-Zeichen mit dem standardmäßig vorgegebenen Zeilenbereich von 0.0001-9999.9999.

Wird kein Operand angegeben, wird das Bereichssymbol zurückgenommen. Es existiert dann kein Zeilenbereichssymbol, bis wieder ein neues durch @RANGE vereinbart wird. Der Spaltenbereich wird nicht verändert.

Ist r kein Sonderzeichen, wird @RANGE mit einer Fehlermeldung abgewiesen:
% EDT3952 INVALID SYMBOL

@READ Einlesen einer SAM-Datei

Mit @READ wird eine SAM-Datei ganz oder teilweise von Platte oder Band in die aktuelle Arbeitsdatei eingelesen.

Die Datei ist nur während der Ausführung von @READ geöffnet. Für die einzulesende SAM-Datei nimmt der EDT standardmäßig variable Satzlänge an (siehe Abschnitt „Bearbeiten von SAM-Dateien mit vom Standard abweichenden Attributen“ auf Seite 43).

Operation	Operanden	F-Modus / L-Modus
@**REA**D	['file'] [(ver)] [range*] [:col:] [{**R**ECORD**S** / **K**EY}] [**S**TRIP]	

file — Dateiname.
Besteht noch kein lokaler @FILE-Eintrag für den Dateinamen, so wird der angegebene Dateiname eingetragen. Fehlt der Operand file, so wird, falls vorhanden, der lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE). Ist weder ein lokaler noch globaler @FILE-Eintrag oder die SAM-Datei nicht vorhanden, wird @READ mit einer Fehlermeldung abgewiesen.

Wenn der Dateikettungsname EDTSAM einer Datei zugeordnet ist, genügt die Angabe /, um diese Datei einzulesen (siehe Abschnitt „Dateibearbeitung“ auf Seite 40ff.).

ver — Versionsnummer der einzulesenden Datei.
Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer.
Wird * angegeben, wird die aktuelle Versionsnummer am Bildschirm angezeigt. Wird eine falsche Versionsnummer angegeben, erscheint die aktuelle auf dem Bildschirm und diese Datei wird eingelesen.

range* — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15 bzw. 0.0004,0.0006,0.0015)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19 bzw. 0.0005-0.0010,0.0017-0.0019)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8 bzw. 0.0004,0.0007-0.0023,0.0008)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Der Wert symbolischer Zeilennummern bezieht sich auf die aktuelle Arbeitsdatei.
Zeichenfolgevariablen dürfen nicht angegeben werden.

Wird range* ohne RECORDS angegeben, interpretiert der EDT die ersten 8 Zeichen jeder Zeile als Zeilennummer und nicht als Zeileninhalt. Beispielsweise wird mit 1 die Zeile angesprochen, die mit der Zeichenfolge 00010000 beginnt.

Wird range* zusammen mit RECORDS angegeben, bezieht sich range* auf die logischen Zeilennummern. Beispielsweise werden mit 0.0001-0.0005 die ersten 5 Zeilen der Datei eingelesen.

Fehlt range*, werden alle Zeilen eingelesen.

col

Spaltenbereich bestehend aus:
- einer oder mehreren (durch Komma getrennten) Spalten
- (z.B. 10,15,8)
- einem oder mehreren (durch Komma getrennten) zusammenhängenden Spaltenbereichen (z.B. 15-25,18-23)
- einer Kombination von einzelnen Spalten und Spaltenbereichen (z.B. 10,14-29,23-50,17)

Wiederholungen und Überlappungen von Spalten und Spaltenbereichen sind erlaubt.

Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge eingelesen.

KEY

Der EDT interpretiert die ersten 8 Zeichen jedes einzulesenden Satzes als Schlüssel. Die Sätze, die in die Arbeitsdatei oder in die durch @OPEN eröffnete Datei gebracht werden, erhalten diesen Schlüssel als Zeilennummer und nicht als Teil des Zeileninhalts.
Der EDT prüft dabei, ob in den ersten 8 Zeichen jeder Zeile ein gültiger Schlüssel steht. Gültig heißt, daß der Schlüssel nur aus den Ziffern 0 bis 9 bestehen darf. Sonst wird @READ mit einer Fehlermeldung abgewiesen.

RECORDS

Gibt dem Benutzer die Möglichkeit, eine nicht mit Schlüsseln versehene SAM-Datei teilweise einzulesen. Hierbei erfolgt eine gedachte Zuordnung von Sätzen und Zeilennummern, wobei 0.0001 für den ersten Satz in der Datei steht, 0.0002 für den zweiten Satz usw. Die eingelesenen Sätze werden an den Inhalt der Arbeitsdatei oder der durch @OPEN eröffneten Datei angehängt. Sie erhalten dort die Zeilennummer, die sich aus der aktuellen Zeilennummer und der aktuellen Schrittweite ergibt. Wenn STRIP nicht angegeben wird, kann RECORDS auch durch R abgekürzt werden.

STRIP Löscht die nachfolgenden Leerzeichen jeder eingelesenen Zeile. Besteht eine Zeile nur aus Leerzeichen, werden alle Leerzeichen bis auf eines gelöscht.

Ist KEY oder range* ohne RECORDS angegeben, werden die Zeilen ignoriert, die kürzer als 8 Zeichen sind.

Bei der Angabe von range* bzw. col können sich Zeilennummern bzw. Spaltennummern wiederholen, was zu einem mehrmaligen Einlesen der entsprechenden Zeilen bzw. Spalten führt.

Vergabe der Zeilennummern

Die Zeilennummern werden abhängig von der aktuellen Zeilennummer und der aktuellen Schrittweite vergeben. Bei leerer Arbeitsdatei sind die aktuelle Zeilennummer und die aktuelle Schrittweite standardmäßig 1. Beide Werte können mit @SET ln (inc) geändert werden (siehe @SET, Format 6).

Beispiel

```
SET 0.01;READ 'file'.
```

Vor dem Einlesen werden die aktuelle Zeilennummer 0.01 und die Schrittweite 0.01 (implizit) vereinbart.

Wird versucht, mit @READ eine ISAM-Datei einzulesen, gibt der EDT eine Fehlermeldung am Bildschirm aus und setzt den Schalter für EDT-Fehler. Er liest die angegebene Datei aber trotzdem ein, da intern ein @GET auf diese Datei gegeben wird.
Wird file angegeben, kann @READ im F-Modus weiterhin mit R abgekürzt werden.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @READ der Coded Character Set Name (CCSN) der Datei berücksichtigt.

Die @READ-Anweisung wird nur dann ausgeführt, wenn entweder der CCSN der Datei gleich dem im EDT aktuell eingestelltem ist, alle Arbeitsdateien leer sind und das Coded Character Set an der Datensichtstation dargestellt werden kann.

Beispiel

```
  23.00 ..........................................................................
read 'test.sam.f'.............................................0000.00:001(1)
```

Die Datei TEST.SAM.F soll eingelesen werden.

```
  22.00 ..........................................................................
%  EDT4200 'OPEN': DMS ERROR CODE: 'ODC2'
system........................................................0000.00:001(1)
```

Das Einlesen der Datei SAM.TEST.F wird mit Fehlermeldung % EDT4200 zurückgewiesen.

Um die Katalogeinträge der Datei ausgeben zu lassen, soll in den Systemmodus gewechselt werden.

```
/show-file-attributes test.sam.f,information=all
%00000114 :1OSN:$USER.TEST.SAM.F
%  ------------------------------ HISTORY      --------------------------------
%  CRE-DATE   = 1994-08-30  ACC-DATE   = 1994-10-07  CHANG-DATE = 1994-08-30
%  CRE-TIME   =   07:22:23  ACC-TIME   =   10:34:09  CHANG-TIME =   07:22:26
%  ACC-COUNT  = 20          S-ALLO-NUM = 7
%  ------------------------------ SECURITY     --------------------------------
%  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
%  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
%  AUDIT      = NONE        DESTROY    = NO          EXPIR-DATE = 1994-08-29
%  SP-REL-LOCK= NO                                  EXPIR-TIME =   23:00:00
%  ------------------------------ BACKUP       --------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED
%  ------------------------------ ORGANIZATION ---------------------------------
%  FILE-STRUC = SAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
%  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
%  REC-FORM   = (V,N)       REC-SIZE   = 20
%  ------------------------------ ALLOCATION   ---------------------------------
%  SUPPORT    = PUB         S-ALLOC    = 18          HIGH-US-PA = 114
%  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
%       8       1OSN.4       D3480
%  NUM-OF-EXT = 8
%PLEASE ACKNOWLEDGE
```

```
  %:1OSN: PUBLIC:      1 FILE  RES=        3 FREE=        2 REL=        0 PAGES
 /set-file-link link-name=edtsam,file-name=test.sam.f, -
 /              record-format=fixed(record-size=20)
 /resume-program
```

Mit dem SHOW-FILE-ATTRIBUTES-Kommando werden die Katalogeinträge der Datei TEST.SAM.F angefordert und ausgegeben. Die Datei hat eine feste Satzlänge von 20 Bytes.

Mit dem SET-FILE-LINK-Kommando wird der Datei TEST.SAM.F der Dateikettungsname EDTSAM zugeordnet. Die Satzlänge wird auf 20 Byte feste Satzlänge eingestellt.

Mit dem RESUME-PROGRAM-Kommando wird in den Programmodus zurückgekehrt, der EDT ist noch geladen.

```
  23.00 ..........................................................................
read 'test.sam.f'.............................................0000.00:001(1)
```

Die Datei TEST.SAM.F soll eingelesen werden.

```
   1.00 DIESE SAM-DATEI.....................................................
   2.00 HAT EINE............................................................
   3.00 FESTE SATZLAENGE.....................................................
   4.00 DIE SATZLAENGE........................................................
   5.00 BETRAEGT 20 BYTES. ...................................................
   6.00 ..........................................................................









delete ; read '/'.............................................0001.00:001(1)
```

Durch das Zuweisen des Dateikettungsnamens EDTSAM ließ sich die Datei TEST.SAM.F einlesen.

Die Arbeitsdatei soll gelöscht und die Datei TEST.SAM.F erneut eingelesen werden. Statt des Dateinamens TEST.SAM.F kann auch / angegeben werden.

```
   1.00 DIESE SAM-DATEI.....................................................
   2.00 HAT EINE.......... ...................................................
   3.00 FESTE SATZLAENGE. ...................................................
   4.00 DIE SATZLAENGE........................................................
   5.00 BETRAEGT 20 BYTES. ...................................................
   6.00 ..........................................................................









write 'test.sam.neu'..........................................0001.00:001(1)
```

Die Datei TEST.SAM.F wurde wieder eingelesen.

Die aktuelle Arbeitsdatei soll in die Datei TEST.SAM.NEU geschrieben werden.

```
  22.00 ..........................................................................
%  EDT4900 A FILE COMMAND IS IN EFFECT
system '/remove-file-link edtsam' ; write 'test.sam.neu'.........0000.00:001(1)
```

Solange der Dateikettungsname EDTSAM einer Datei zugewiesen ist, kann in keine andere Datei geschrieben werden.
Die Zuordnung des Dateikettungsnamens EDTSAM zu der Datei TEST.SAM.F soll aufgehoben und die aktuelle Arbeitsdatei in die Datei TEST.SAM.NEU geschrieben werden.
Es wird eine neue SAM-Datei mit Standard-Merkmalen angelegt.

@RENUMBER Neu numerieren

Mit @RENUMBER werden die in der virtuellen Datei aufgebauten Zeilen neu durchnumeriert. Es kann sowohl die erste Zeile angegeben werden, bei der die Neunumerierung erfolgen soll, als auch die Schrittweite.

Die neue aktuelle Zeilennummer ergibt sich aus der nach der Neunumerierung höchsten Zeilennummer plus der aktuellen Schrittweite.

Operation	Operanden	F-Modus / L-Modus
@**R**ENUMBER	[ln [(inc)]]	

ln Zeilennummer (z.B. 5), bei der die Neunumerierung beginnen soll.
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

inc Schrittweite zur Berechnung der neuen Zeilennummern. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. Fehlen ln und inc, ist die Schrittweite 1.

Wird kein Operand angegeben, wird sowohl für ln als auch für inc der Standardwert 1 angenommen.

@RENUMBER kann nicht für eine durch @OPEN eröffnete Datei gegeben werden.

Die Zeilennummern, die im Kopierpuffer vorhanden sind (siehe Abschnitt „Kurzanweisungen im F-Modus“, „C Markieren zum Kopieren“ auf Seite 80ff. und „R Markieren zum Kopieren (ohne Löschen des Kopierpuffers)“ auf Seite 92ff.), werden nicht umnumeriert.

Wird die Anweisung im Dialog abgesetzt und würden Zeilen verloren gehen, weil die höchstmögliche Zeilennummer erreicht wird, wird die Meldung ausgegeben:

```
% EDT0910 '@RENUMBER': LINES WILL BE LOST
% EDT0911 CONTINUE PROCESSING? REPLY (Y=YES; N=NO)
```

N: @RENUMBER wird nicht ausgeführt,

Y: @RENUMBER wird ausgeführt und folgende Meldung ausgegeben:

```
% EDT2904 MAXIMUM LINE NUMBER WHEN PROCESSING '@RENUMBER'.
SOME LINES ARE LOST.
```

Im F-Modus-Fenster bleibt der gezeigte Ausschnitt erhalten, es sei denn die Zeilen im Arbeitsfenster gehen durch die Umnumerierung verloren. In diesem Fall wird das Arbeitsfenster auf die letzte Zeile positioniert.

Berechnung der Schrittweite

Wird ln, nicht aber inc angegeben, hängt die Schrittweite von der Anzahl der in ln angegebenen Dezimalstellen ab. Z.B. wird für ln = 2 die Schrittweite 1 genommen, für ln = 2.4 jedoch die Schrittweite 0.1, für ln = 2.40 die Schrittweite 0.01 usw.

i

– Die im 3-stufigen Keller (siehe auch @) enthaltenen Einträge werden von @RENUMBER nicht verändert. Nach einer Neunumerierung wird es jedoch im Regelfall nicht mehr sinnvoll sein, auf diese Kellerungseinträge zurückzugreifen.

– Information über die Anzahl der Zeilen in der aktuellen Arbeitsdatei kann mit @LIMITS angefordert werden.

@RESET EDT- und DVS-Fehlerschalter rücksetzen

Mit @RESET wird der EDT- und DVS-Fehlerschalter zurückgesetzt (siehe auch @IF, Format 1).

Operation	Operanden	F-Modus / L-Modus / @PROC
@**RES**ET		

@RETURN Beenden des Bildschirmdialogs und Abbrechen von Prozeduren

Mit @RETURN kann man

– den EDT beenden,
– den Bildschirmdialog nach @DIALOG beenden,
– Prozeduren (@DO- und @INPUT-Prozeduren) abbrechen,
– den Bildschirmdialog nach Aufruf des EDT als Unterprogramm beenden.

Operation	Operanden	F-Modus / L-Modus
@**RET**URN	[message]	

message Beliebiger Text. Das Ende wird durch das Anweisungsende bestimmt. Zwischen @RETURN und message muß mindestens ein Blank stehen.

message darf nur bei Aufruf des EDT über die Unterprogrammschnittstelle angegeben werden.

Existieren noch ungesicherte Arbeitsdateien, so werden nach der Meldung: % EDT0900 EDITED FILE(S) NOT SAVED! die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben.

Zusätzlich wird ausgegeben, falls vorhanden, entweder

– ein lokaler @FILE-Eintrag
  – explizit definiert durch @FILE LOCAL, oder
  – implizit definiert durch @READ, @GET, @OPEN (Format 1)
– oder der Bibliotheks- und Elementsname eines mit
  – @OPEN (Format 2) eröffneten Bibliothekselements
– oder der Dateiname einer mit
  – @OPEN (Format 2) eröffneten SAM-oder ISAM-Datei
  – @XOPEN eröffneten POSIX-Datei

Danach folgt die Anfrage an den Benutzer:

% EDT0904 TERMINATE EDT? REPLY (Y=YES, N=NO)

N: Im F-Modus erscheint das Arbeitsfenster wieder; im L-Modus die Eingabeaufforderung. Der Benutzer kann ungesicherte Dateien schließen und zurückschreiben.

Y: Die ungesicherten, virtuellen Dateien gehen verloren. Der EDT wird beendet, das angegebene Programm gestartet.

Die Sicherungsabfrage kann unterdrückt werden, indem man vor dem Aufruf des EDT den Auftragsschalter 4 einschaltet.

@RETURN nach einem Bildschirmdialog eingeleitet durch START-PROGRAM $EDT

Wurde der Bildschirmdialog durch START-PROGRAM $EDT eingeleitet, wirkt @RETURN wie @HALT. Der EDT wird beendet.

@RETURN nach einem Bildschirmdialog eingeleitet durch @DIALOG

Wurde der Bildschirmdialog durch @DIALOG eingeleitet, wird mit der durch @DIALOG unterbrochenen Abarbeitung (L-Modus-Dialog oder Lesen von SYSDTA) fortgesetzt.

Die Sicherungsabfrage unterbleibt.

@RETURN nach einem Bildschirmdialog bei Aufruf des EDT als Unterprogramm

Wird @RETURN bei Aufruf des EDT als Unterprogramm eingegeben, wird zum rufenden Benutzerprogramm zurückgekehrt.

Mit message kann man an das rufende Programm einen Text übergeben. Dieser kann nach der Rückkehr aus dem EDT-Bildschirmdialog im rufenden Programm ausgewertet werden. Die Übergabe dieses Textes erfolgt im Meldungsfeld des Kontrollblocks EDTGLCB (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]).

@RETURN in EDT-Prozeduren

Wird @RETURN in @DO- oder @INPUT-Prozeduren verwendet, wird die Abarbeitung der Prozedur abgebrochen und dort weitergearbeitet, wo der Prozeduraufruf erfolgte.

Beispiel 1

```
1.     @PROC 6 ------------------------------------------------------ (01)
1.     @ @SET #S1 = 'ICH BIN #S1'
2.     @ @SET #S2 = 'ICH BIN #S2'
3.     @ @PRINT #S1
4.     @ @RETURN ---------------------------------------------------- (02)
5.     @ @PRINT #S2
6.     @END --------------------------------------------------------- (03)
1.     @DO 6 -------------------------------------------------------- (04)
  #S01 ICH BIN #S1
```

(01) Die Arbeitsdatei 6 wird zur Bearbeitung geöffnet.

(02) Kommt es beim Ausführen der Prozedur in Arbeitsdatei 6 zu dieser Anweisung, werden die nachfolgenden Anweisungen nicht mehr ausgeführt.

(03) Die Bearbeitung der Arbeitsdatei 6 wird beendet.

(04) Die Prozedur wird ausgeführt.

Beispiel 2

```
1.     AAAA
2.     BBBB
3.     CCCC
4.     @PROC 7 ------------------------------------------------------ (01)
1.     @ @PRINT ! --------------------------------------------------- (02)
2.     @ @RETURN ---------------------------------------------------- (03)
3.     @END
4.     @DO 7,!=%,$ -------------------------------------------------- (04)
1.0000 AAAA
4.
```

(01) Die Prozedur wird in der Arbeitsdatei 7 erstellt.

(02) Beim Ausführen der Anweisungen der Prozedur soll die über das Schleifensymbol ! angesprochene Zeile ausgegeben werden.

(03) Die Ausführung der Anweisungen der Prozedur wird an dieser Stelle unterbrochen, egal ob das Schleifensymbol ! bereits die obere Grenze erreicht hat oder nicht.

(04) Die Prozedur wird aufgerufen. Dabei soll das Schleifensymbol ! die Werte 1 bis 3 (%=1, $=3) annehmen. Wegen des in der Prozedur stehenden @RETURN kommt es jedoch nicht zu einem Hochzählen des Schleifensymbolwertes.

@RUN Aufruf eines Benutzerprogramms als Unterprogramm

Mit @RUN wird ein Benutzerprogramm als Unterprogramm geladen und gestartet. Das Programm muß in Form eines Bindemoduls in einer Bibliothek vorliegen (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]). Anders als bei Programmaufrufen mit @LOAD oder @EXEC bleibt der EDT geladen und die virtuellen Arbeitsdateien bleiben erhalten.

Soll ein Benutzerprogramm ablaufen, das nur im 24-Bit-Adressierungsmodus ablauffähig ist, so ist der EDT über das Treibermodul EDTC zu laden (siehe auch Abschnitt „Starten des EDT“ auf Seite 25ff.).

Das aufgerufene Benutzerprogramm kann die Zeilen der aktuellen Arbeitsdatei, in der @RUN eingegeben wurde, bearbeiten.

Operation	Operanden	F-Modus / L-Modus
@RUN	(entry [,modlib]) [:string] [[,]**U**NLOAD]	

entry — Name
- eines Programmabschnittes (CSECT),
- einer Einsprungstelle (ENTRY) eines Programms.

Das entsprechende Bindemodul oder die Ladeeinheit wird geladen. entry kann auch über eine Zeichenfolgevariable angegeben werden.

modlib — Name der Bibliothek, in der das Bindemodul oder die Ladeeinheit gespeichert ist. modlib kann auch über eine Zeichenfolgevariable angegeben werden.
Fehlt diese Angabe, wird das Bindemodul oder die Ladeeinheit in der Modulbibliothek mit dem Standardnamen EDTRUNLB gesucht.

string — Zeichenfolge, die an das aufgerufene Programm übergeben wird.

string kann angegeben werden:
- explizite Angabe in Hochkomma,
- implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Wird string angegeben, übergibt der EDT in
- Register 2 die Adresse des 1. Byte der Zeichenfolge,
- Register 3 ihre um 1 reduzierte Länge.

UNLOAD — Gibt an, daß das Modul nach der Rückkehr in den EDT entladen werden soll. UNLOAD kann verwendet werden, wenn der Modulname mit dem Namen der Einsprungstelle (ENTRY) übereinstimmt.

Wenn eine Datei durch @OPEN real eröffnet ist, ist @RUN nicht erlaubt.

@SAVE Schreiben als ISAM-Datei

Mit @SAVE wird der Inhalt der aktuellen Arbeitsdatei ganz oder teilweise als ISAM-Datei auf Platte geschrieben.

Die ISAM-Datei ist nur während der Ausführung von @SAVE geöffnet.

Operation	Operanden	F-Modus / L-Modus
@**SA**VE	['file'] [(ver)] [range*] [:col:] [{ **U**PDATE / [**R**ENUMBER [ln [(inc)]]] [**O**VERWRITE] }]	

file Dateiname.
Fehlt der Operand file, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag, dann der globale @FILE-Eintrag und zuletzt der implizite lokale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE); Ansonsten wird @SAVE mit einer Fehlermeldung abgewiesen.

Wenn der Dateikettungsname EDTISAM einer Datei zugeordnet ist, genügt die Angabe /, um diese Datei zurückzuschreiben (siehe Abschnitt „Dateibearbeitung“ auf Seite 40ff.).

ver Versionsnummer der Datei.
Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer.
Wird eine falsche Versionsnummer angegeben, wird die richtige Versionsnummer am Bildschirm ausgegeben.

Versionsnummern sind vorgesehen, um Dateien vor unbeabsichtigtem Überschreiben zu schützen. Eine neue Versionsnummer entsteht, wenn eine Datei erstmalig angelegt oder eine bereits existierende Datei verändert wird. Hierbei erhöht sich die Versionsnummer um 1. Eine neu angelegte Datei erhält die Versionsnummer 1. Die maximale Versionsnummer ist 255. Die darauffolgende Versionsnummer ist 0.

range* Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereiche (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

Fehlt range*, werden alle Zeilen in die ISAM-Datei geschrieben.

col — Spaltenbereich bestehend aus:
- einer oder mehreren (durch Komma getrennten) Spalten (z.B. 10,15,8)
- einem oder mehreren (durch Komma getrennten) zusammenhängenden Spaltenbereichen (z.B. 15-25,18-23)
- einer Kombination von einzelnen Spalten und Spaltenbereichen (z.B. 10,14-29,23-50,17)

Wiederholungen und Überlappungen von Spalten und Spaltenbereichen sind erlaubt.

Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge gespeichert.

UPDATE — Ist nur sinnvoll, wenn bereits eine ISAM-Datei mit dem angegebenen Namen existiert.
UPDATE bewirkt, daß die abzuspeichernden Zeilen in die ISAM-Datei eingefügt werden. Der EDT überschreibt in der ISAM-Datei nur die Zeilen, deren Nummern auch in der abzuspeichernden Datei (Arbeitsdatei, virtuellen bzw. mit @OPEN eröffneten Datei) existieren, und im Bereich range* liegen. Die restlichen Zeilen der ISAM-Datei bleiben erhalten.

RENUMBER — Die abzuspeichernden Zeilen werden neu numeriert.
Die Zeilennumerierung in der Arbeitsdatei, virtuellen bzw. der durch @OPEN eröffneten Datei bleibt unverändert.
Wird RENUMBER nicht angegeben, entstehen die ISAM-Schlüssel aus den Zeilennummern der abzuspeichernden Zeilen. Beim Wiedereinlesen in die Arbeitsdatei, virtuelle bzw. die durch @OPEN eröffnete Datei bleiben diese Zeilennummern erhalten, wenn bei @GET der Operand NORESEQ angegeben wird.

ln — Startnummer des ISAM-Satzschlüssels für die zu schreibende Datei. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. Fehlt inc, wird mit ln implizit die Schrittweite des ISAM-Satzschlüssels festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden. Wird ln nicht angegeben, ist die Startnummer des ISAM-Satzschlüssels 0001.0000.

inc Schrittweite des ISAM-Satzschlüssels.

Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. Fehlen ln und inc, ist die Schrittweite des ISAM-Satzschlüssels 1.

OVERWRITE Unterdrückt die Abfrage OVERWRITE FILE? (Y/N).

Eine vorhandene Datei gleichen Namens wird überschrieben. Existiert die Datei file noch nicht, ist OVERWRITE wirkungslos.

Wird UPDATE oder OVERWRITE nicht angegeben und existiert bereits eine Datei mit dem gleichen Namen, reagiert der EDT mit der Frage:

% EDT0903 FILE 'file' IS IN THE CATALOG, FCBTYPE = fcbtyp
% EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)

Antwortet der Benutzer mit

N wird @SAVE nicht ausgeführt,

Y wird @SAVE ausgeführt und die bestehende Datei als ISAM-Datei mit dem Inhalt der aktuellen Arbeitsdatei überschrieben.

Bei variabler Satzlänge (RECORD-FORMAT=VARIABLE; siehe auch Abschnitt „Bearbeiten von ISAM-Dateien mit vom Standard abweichenden Attributen“ auf Seite 42) gehen beim Zurückschreiben ab Position 257 die Sätze verloren.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @SAVE nach dem Zurückschreiben der Datei ein Coded Character Set Name (CCSN) als Codemerkmal mitgegeben.

Unabhängig davon, ob die Datei bereits existiert und welchen CCSN sie besitzt, wird mit @SAVE der aktuell im EDT eingestellte CCSN vergeben.

@SDFTEST Syntaxprüfung von Datenzeilen durch SDF

Mit @SDFTEST

- wird der Inhalt einer Zeile oder eines Zeilenbereiches an SDF zur Syntaxkontrolle übergeben
- wird eine von SDF zurückgegebene Zeichenfolge in den Datenbereich aufgenommen
- kann ein Programmname zur Prüfung von Anweisungen eingestellt werden

Im Stapelbetrieb wird diese Anweisung nicht ausgeführt, sondern ignoriert.

Operation	Operanden	F-Modus / L-Modus
@**SDF**TEST	[range*] [**PROG**RAM [=name]]	

range* Zeilenbereich bestehend aus ein oder mehreren Zeilen und Zeilenbereichen, kann auch symbolisch angegeben werden.
Zeichenfolgevariablen können nicht angegeben werden.
Alle Zeilen des Zeilenbereiches, die mit einem / beginnen, werden zusammen mit ihren Folgezeilen an SDF zur Syntaxkontrolle übergeben.
Eine Folgezeile wird erwartet, wenn die Zeile abgesehen von Leerzeichen und Nilzeichen mit – endet.
Es sind maximal 255 Folgezeilen erlaubt.
Ist range* nicht angegeben, werden alle Zeilen der Arbeitsdatei berücksichtigt.

PROGRAM Zusätzlich werden Datenzeilen, die mit // beginnen an SDF als Anweisungen übergeben.

=name Interner Name des Programms, dessen Anweisungen geprüft werden sollen. maximal 8-stellig (z.B. $LMSSDF, BINDER, $SDAEDXT)
SDF verwendet zur Prüfung die aktuelle Syntaxhierarchie, in der die Anweisungen des Programms beschrieben sein muß.
Ist dieser Operand nicht angegeben, wird der durch ein vorhergegangenes @SDFTEST mit PROGRAM=name oder durch @PAR SDF-PROGRAM=name eingestellte interner Programmname verwendet oder, falls kein Name voreingestellt ist, eine Fehlermeldung ausgegeben.

Ist der Operand PROGRAM nicht angegeben, wird im angegebenen Zeilenbereich nur auf Kommandosyntax geprüft, d.h. Zeilen, die nicht mit einem einzelnen / beginnen, werden nicht berücksichtigt.

Nachdem SDF die Syntax geprüft hat, wird je nach Einstellung der SDF-Optionen das von SDF zurückgegebene Format des Kommandos oder der Anweisung an die gleiche Stelle im Datenbereich zurückgeschrieben. Falls nötig, werden Fortsetzungszeilen eingefügt oder gelöscht.

Fortsetzungszeilen bei der Ausgabe

Ist die Ausgabe länger als 71 Zeichen wird sie in Teilstücke zerlegt. Das Fortsetzungszeichen wird an die 72. Spalte gesetzt. Wenn nötig, werden die nachfolgenden Zeilen umnumeriert.

Im F-Modus wird die Meldung ausgegeben:
% EDT0285 SDF: SYNTAX TESTED. '0' ERROR(S) IN RANGE.

Verhalten, wenn SDF einen Syntaxfehler bemerkt:

Tritt bei der Prüfung durch SDF ein Fehler auf, so wird bei eingestelltem GUIDANCE-Mode MIN/MED/MAX in den geführten SDF-Fehlerdialog gewechselt, und der Anwender kann das Kommando oder die Anweisung korrigieren.

Wird der Fehlerdialog mit F1 abgebrochen oder ist er durch die Einstellung GUIDANCE=NO/EXPERT nicht möglich, gibt EDT eine Fehlermeldung aus:
% EDT4310 SDF: SYNTAX ERROR AT LINE (&00)

Anschließend wird, wenn noch Datenzeilen zu bearbeiten sind, der Anwender gefragt, ob die Bearbeitung fortgesetzt werden soll:
% EDT0911 CONTINUE PROCESSING? REPLY(Y=YES; N=NO)

Y, es wird mit der Bearbeitung weiterer Zeilen fortgefahren.
N, es wird die Abarbeitung unterbrochen.

Im F-Modus wird die fehlerhafte Zeile an oberster Bildschirm-Position gezeigt.

Die Namen der aktuell eingestellten Syntaxdateien und des vordefinierten internen Programmnamens können mit der Anweisung @STATUS=SDF erfragt werden.

Der Programmname kann mit der Anweisung @PAR SDF-PROGRAM=... voreingestellt werden. Die explizite Angabe eines Programmnamens in @SDFTEST überschreibt die Voreinstellung.

Der interne Programmname kann mit SDF-A ermittelt werden, falls er nicht dem Namen des Programms entspricht wie z.B. bei C, BINDER, LMS, SORT:

ASSEMBH	INTERNAL-NAME=ASS$XT
PASCAL-XT	INTERNAL-NAME=PASXT21A
SDF-A	INTERNAL-NAME=$SDAEDXT
SDF-U	INTERNAL-NAME=CMDEDIT

Kennwörter und andere Operanden, die mit OUTPUT=SECRET-PROMPT definiert wurden, werden bei GUIDANCE-Einstellung MIN,MED oder MAX durch P ersetzt.

&Ersetzung wird nur in Operandenwerten akteptiert, nicht jedoch in Kommando-, Anweisungs- und Operandennamen oder Teile von ihnen.

@SEQUENCE Zeilennummern prüfen bzw. übernehmen

@SEQUENCE bewirkt:

– Der EDT schreibt in jede Zeile eines Zeilenbereichs eine Zahl. Die Zahlen bilden eine aufsteigende Folge (Format 1).
– Der EDT schreibt in jede Zeile eines Zeilenbereichs die zugehörige Zeilennummer (Format 2).
– In jeder Zeile eines Zeilenbereichs betrachtet der EDT den Inhalt einer oder mehrerer aufeinanderfolgender Spalten. Er interpretiert die dort stehende Zeichenfolge gemäß ihrer EBCDI-Codierung als Dualzahl. Der EDT prüft, ob diese Dualzahlen eine aufsteigende Folge bilden (Format 3).

@SEQUENCE (Format 1) Zeilen numerieren

@SEQUENCE, Format 1 bewirkt, daß der EDT in jede Zeile eines zusammenhängenden Zeilenbereichs eine Zahl schreibt. Diese Zahlen bilden eine aufsteigende Folge.

Der EDT überschreibt dabei den etwaigen Inhalt der Spalten, in die er die Zahlen schreibt.

Operation	Operanden	F-Modus / L-Modus
@**SEQ**UENCE	[rng] [:[cl] [:[[n1] (n2)]]]	

rng — Zeilenbereich, bestehend aus:
– einer einzelnen Zeile (z.B. 6)
– mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.
Fehlt rng, schreibt der EDT in jede Zeile der virtuellen bzw. der durch @OPEN eröffneten Datei eine Zahl.

cl — Spalte, in der die erste Ziffer der zu schreibenden Zahl stehen soll. Fehlt cl, schreibt der EDT die erste Ziffer in Spalte 73.

n1 — Ganze Dezimalzahl, die der EDT in die erste Zeile des betrachteten Zeilenbereichs schreibt. n1 hat maximal 8 Stellen. Die Zahlen, die in die folgenden Zeilen geschrieben werden, haben die gleiche Stellenanzahl. Der Wert von n1 ist beliebig. Fehlt n1, schreibt der EDT in die erste betrachtete Zeile die Zahl 00000100.

n2 Ganzzahlige Schrittweite zur Bildung der folgenden Zahlen. Diese sind jeweils die Summe aus der vorhergehenden Zahl und der Schrittweite. Fehlt n2, nimmt der EDT als Schrittweite den Wert 100.

@SEQUENCE (Format 2) Zeilennummern übernehmen

@SEQUENCE, Format 2 bewirkt, daß der EDT in jede Zeile eines zusammenhängenden Zeilenbereichs die zugehörige Zeilennummer schreibt. Die Zeilennummer wird als 8-stellige Zahl ohne Dezimalpunkt geschrieben. Falls erforderlich, wird sie rechts- und linksbündig mit Nullen aufgefüllt.

Beispielsweise wird in die Zeile 12.345 die Zahl 00123450 geschrieben. Der EDT überschreibt dabei den etwaigen Inhalt der 8 Spalten, in die er die Zeilennummer schreibt.

Operation	Operanden	F-Modus / L-Modus
@**SEQ**UENCE	[rng*] : [cl] : **L**INE	

rng* Zeilenbereich, bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

Fehlt rng*, schreibt der EDT in jede Zeile der virtuellen bzw. der durch @OPEN eröffneten Datei die zugehörige Zeilennummer.

cl Spalte, ab der der EDT die Zeilennummern schreiben soll. Fehlt cl, schreibt der EDT die Zeilennummern ab Spalte 73.

Beispiel

```
   0.00 AUF.....................................................................
   1.11 DIE.....................................................................
  88.76 REIHENFOLGE.............................................................
  88.76 KOMMT...................................................................
5555.00 ES......................................................................
9876.54 AN......................................................................
9877.54 .....................................................................




sequence :20: line ..............................................0000.00:001(0)
```

In jede Zeile der Arbeitsdatei soll ab Spalte 20 die zugehörige Zeilennummer geschrieben werden.

```
   0.00 AUF                00000035.....................................
   1.11 DIE                00011110.....................................
  88.76 REIHENFOLGE        00887610.....................................
  88.76 KOMMT              00887620.....................................
5555.00 ES                 55550000.....................................
9876.54 AN                 98765432.....................................
9877.54 .....................................................................
```

Die Zeilennummern wurden als 8-stellige Zahl ohne Dezimalpunkt ab Spalte 20 geschrieben. Die Zeilennummern wurden ggf. mit Nullen links und rechts aufgefüllt.

@SEQUENCE (Format 3) Zeilennummern überprüfen

@SEQUENCE, Format 3 bewirkt, daß der EDT in jeder Zeile eines zusammenhängenden Zeilenbereichs den Inhalt einer Spalte oder mehrerer zusammenhängender Spalten untersucht. Er interpretiert die dort stehende Zeichenfolge gemäß ihrer EBCDI-Codierung als Dualzahl. Liegt die zu untersuchende Spalte rechts vom Zeilenende, nimmt der EDT als Spalteninhalt den Wert X’40’ an.

Der EDT prüft, ob die Dualzahlen eine aufsteigende Folge bilden. Er gibt alle Zeilen aus, in denen er eine Dualzahl findet, die gleich oder kleiner ist, als die aus der vorhergehenden Zeile.

Operation	Operanden	F-Modus / L-Modus
@**SEQ**UENCE	[rng] : [cl] : **C**HECK [int]	

rng Zeilenbereich, bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

Fehlt rng, überprüft der EDT alle Zeilen der Hauptdatei.

cl Spalte, in der das erste zu überprüfende Zeichen steht. Fehlt cl, beginnt der EDT die Überprüfung in Spalte 73.

int Gibt an, wieviele Spalten (1-8) betrachtet werden sollen. Fehlt int, betrachtet der EDT 8 Spalten.

Beispiel

```
   1.00 15 ZEILE 1.............................................................
   2.00 20 ZEILE 2.............................................................
   3.00 21 ZEILE 3.............................................................
   4.00 16 ZEILE 4.............................................................
   5.00 18 ZEILE 5.............................................................
   6.00 01 ZEILE 6.............................................................
   7.00 99 ZEILE 7.............................................................
   8.00 97 ZEILE 8.............................................................









sequence & :1: check 2 .....................................0001.00:001(0)
```

Der EDT soll alle Zeilen darauf überprüfen, ob der Inhalt der Spalten 1 bis 2 eine aufsteigende Folge bildet.

```
   4.0000 16 ZEILE 4
   6.0000 01 ZEILE 6
   8.0000 97 ZEILE 8
PLEASE ACKNOWLEDGE
```

Der EDT gibt alle Zeilen aus, die von der aufsteigenden Folge abweichen.

```
   1.00 15 ZEILE 1.............................................................
   2.00 20 ZEILE 2.............................................................
   3.00 21 ZEILE 3.............................................................
   4.00 16 ZEILE 4.............................................................
   5.00 18 ZEILE 5.............................................................
   6.00 01 ZEILE 6.............................................................
   7.00 99 ZEILE 7.............................................................
   8.00 97 ZEILE 8.............................................................




sequence 1-4 :1: check 1 .....................................0001.00:001(0)
```

Nun soll in den ersten 4 Zeilen nur der Inhalt der 1. Spalte zur Überprüfung verwendet werden.

```
   3.0000 21 ZEILE 3
   4.0000 16 ZEILE 4
PLEASE ACKNOWLEDGE
```

Die Folge lautet 1 (Zeile 1), 2 (Zeile 2), 2 (Zeile 3) und 1 (Zeile 4). Man beachte insbesondere, daß bei Gleichheit ein Verstoß gegen eine aufsteigende Folge vorliegt. Das ist hier bei Zeile 3 der Fall.

@SET EDT-Variable mit Werten versorgen

@SET bietet in 6 Formaten folgende Funktionen:

Versorgen von Ganzzahlvariablen mit Werten (Format 1)

– weist einer Ganzzahlvariablen einen ganzzahligen Ausdruck zu
– weist einer Ganzzahlvariablen ein abdruckbare Zahl als Ganzzahl zu
– weist einer Ganzzahlvariablen den Inhalt einer Zeilennummervariablen als Ganzzahl zu
– weist einer Ganzzahlvariablen die Länge einer Zeile zu
– weist einer Ganzzahlvariablen den EBCDI-Code einer Zeichenfolge zu

Versorgen von Zeichenfolgevariablen mit Werten (Format 2)

– weist einer Zeichenfolgevariablen eine Zeichenfolge zu
– weist einer Zeichenfolgevariablen den Inhalt einer Ganzzahlvariablen, eine Zeilennummer oder den Namen einer Zeichenfolgevariablen zu
– legt den in eine abdruckbare Zahl konvertierten Inhalt einer Ganzzahlvariablen in einer Zeichenfolgevariablen ab
– legt das abdruckbare Bild einer Zeilennummer in einer Zeichenfolgevariablen ab
– legt den Namen einer Zeichenfolgevariablen in einer Zeichenfolgevariablen ab

Versorgen von Zeilennummervariablen mit Werten (Format 3)

– weist einer Zeilennummervariablen eine Zeilennummer zu
– weist einer Zeilennummervariablen den Inhalt einer Ganzzahlvariablen umgewandelt in eine Zeilennummer zu
– weist einer Zeilennummervariablen eine abdruckbare Zahl als Zeilennummer zu
– weist einer Zeilennummervariablen die interne Darstellung einer Zeichenfolge zu

Werte in Zeilen ablegen (Format 4)

– legt den Inhalt einer Ganzzahlvariablen in einer abdruckbaren Form in einer Zeile ab
– schreibt den Namen einer Zeichenfolgevariablen in eine Zeile
– legt den abdruckbar gemachten Inhalt einer Zeilennummervariablen in einer Zeile ab

Datum und Uhrzeit (Format 5)

– legt Datum oder Uhrzeit in einer Zeichenfolgevariablen ab
– legt Datum oder Uhrzeit in einer Zeile ab

Bestimmen der neuen aktuellen Zeilennummer und Schrittweite (Format 6)

– bestimmt eine neue aktuelle Zeilennummer und die Schrittweite

@SET (Format 1) Versorgen von Ganzzahlvariablen mit Werten

Mit diesem Format von @SET wird:

– einer Ganzzahlvariablen ein ganzzahliger Ausdruck zugewiesen
– eine abdruckbare Zahl einer Ganzahlvariablen als Ganzzahl zugewiesen
– der Inhalt einer Zeilennummervariablen in eine Ganzzahl umgewandelt und diese der angegebenen Ganzzahlvariablen als Wert zugewiesen
– die Länge einer Zeile ermittelt und einer Ganzzahlvariablen als Wert zugewiesen. Existiert die angegebene Zeile nicht, wird der Ganzzahlvariablen der Wert 0 zugewiesen
– einer Ganzzahlvariablen der EBCDI-Code einer angegebenen Zeichenfolge zugewiesen

Operation	Operanden	F-Modus / L-Modus / @PROC
@SET	int-var = { [+|–] int [+|– int] [...] **S**UBSTR string ln-var **L**ENGTH line **ST**RING string }	

int-var — Ganzzahlvariable (#I0 bis #I20), der ein Wert zugewiesen werden soll.

+/– — Führt eine arithmetische Verknüpfung der angegebenen int durch.

int — Ganze vorzeichenlose Zahl (z.B. 5, 0 oder 17) oder eine der Ganzzahlvariablen #I0 bis #I20, die (evtl. nach arithmetischer Verknüpfung mit anderen int) der Ganzzahlvariablen als Wert zugewiesen wird.
Wird bei einer Rechenoperation der positive oder negative Maximalwert ($2^{31}-1$,-2^{31}) überschritten, gibt der EDT die Fehlermeldung OVERFLOW ERROR aus.

... — Deutet an, daß mehrere int durch + oder – miteinander verknüpft werden können. Die einzige Einschränkung ist die maximale Zeilenlänge von 256 Zeichen.

string — Zeichenfolge.
string kann angegeben werden:
– direkte Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

SUBSTR string

Der EDT weist der angegebenen Ganzzahlvariablen eine abdruckbare Zahl als Wert zu. Beispielsweise wird die abdruckbare Zahl 17 als Ganzzahl 17 zugewiesen.
Enthält string keine Zahl, wird @SET mit einer Fehlermeldung abgewiesen. Ein in string enthaltenes Plus- oder Minuszeichen wird bei der Konvertierung berücksichtigt. Fehlt das Vorzeichen, wird + angenommen.
Enthält string Leerzeichen, werden diese bei der Umwandlung unterdrückt.

ln-var — Gibt eine der 21 Zeilennummervariablen an (#L0 bis #L20).
Der größte Wert einer Zeilennummervariablen ist 9999.9999, der kleinste 0.0001. Bei der Konvertierung in eine Ganzzahl wird der Inhalt der Zeilennummervariablen mit 10000 multipliziert und linksbündig mit Nullen aufgefüllt, um 10 Ziffern zu erhalten. Dies ist deshalb notwendig, damit die hinter dem Dezimalpunkt stehenden Ziffern einer Zeilennummer Berücksichtigung finden. Die Zeilennummer 1.23 wird in die Zahl 12300 konvertiert. Nach dem linksbündigen Auffüllen mit Nullen wird der Ganzzahlvariablen der Wert 0000012300 zugewiesen.

LENGTH — Zeigt dem EDT an, daß er die Länge einer Zeile in einer Ganzzahlvariablen ablegen soll.

line — Zeilennummer der Zeile, deren Länge in die angegebene Ganzzahlvariable gebracht werden soll (z.B. 15.34 oder #S11 oder #S11 + #I1).

STRING string

Der EDT soll den EBCDI-Code einer Zeichenfolge als Ganzzahl einer Ganzzahlvariablen als Wert zuweisen, z.B. string = '1', interne Darstellung X'F1', der zuzuweisende Wert beträgt also 241.
Der EBCDI-Code von string wird der Ganzzahlvariablen zugewiesen. Besteht string aus weniger als 4 Zeichen, wird linksbündig mit Nullen aufgefüllt. Besteht string aus mehr als 4 Zeichen, werden lediglich die ersten 4 Zeichen berücksichtigt.

Beispiel 1

Einer Ganzzahlvariablen einen ganzzahligen Ausdruck zuweisen.

```
   1.     @SET #I0 = -1 ------------------------------------------------ (01)
   1.     @SET #I1 = #I0 + 1001 ---------------------------------------- (02)
   1.     @SET #I2 = #I1 + #I1 -#I0 + 3 - #I0 -------------------------- (03)
   1.     @SET #I3 = #I2 + #I2 + #I2 + #I2 + #I2 ----------------------- (04)
   1.     @STATUS = I -------------------------------------------------- (05)
#I00=-0000000001 #I01= 0000001000 #I02= 0000002005
#I03= 0000010025 #I04= 0000000000 #I05= 0000000000
#I06= 0000000000 #I07= 0000000000 #I08= 0000000000
#I09= 0000000000 #I10= 0000000000 #I11= 0000000000
#I12= 0000000000 #I13= 0000000000 #I14= 0000000000
#I15= 0000000000 #I16= 0000000000 #I17= 0000000000
#I18= 0000000000 #I19= 0000000000 #I20= 0000000000
```

(01) Der Ganzzahlvariablen #I0 wird der Wert –1 zugewiesen.

(02) Der Ausdruck #I0 + 1001 wird #I1 zugewiesen. Da #I0 den Inhalt –1 hat, erhält #I1 den Inhalt –1 + 1001, also 1000.

(03) Ein mehrmaliges Auftreten derselben Ganzzahlvariablen in einem Ausdruck ist erlaubt.

(04) Eine Multiplikation kann auf ein n-maliges Addieren zurückgeführt werden.

(05) Der Inhalt der Ganzzahlvariablen soll ausgegeben werden.

Beispiel 2

Eine abdruckbare Zahl einer Ganzzahlvariablen als Ganzzahl zuweisen.

```
   1.     @CREATE 2:'AA1234567890BB'
   1.     @CREATE #S2: 'UM 16.05 UHR IST FEIERABEND'   ----------------- (01)
   1.     @SET #I6 = 3
   1.     @SET #L1 = 2
   1.     @SET #I7 = SUBSTR X'F0F1F2F3' -------------------------------- (02)
   1.     @SET #I8 = SUBSTR B'11110001'*6 ------------------------------ (03)
   1.     @SET #I9 = SUBSTR 2:3: --------------------------------------- (04)
   1.     @SET #I10 = SUBSTR 2:5-8,4,3,3,3: ---------------------------- (05)
   1.     @SET #I11 = SUBSTR #S2: 4-5, 7-8: ---------------------------- (06)
   1.     @SET #I12 = SUBSTR '123'*3 ----------------------------------- (07)
   1.     @SET #I13 = SUBSTR #S5-#I6:8,8,8,8,8,8,8: -------------------- (08)
   1.     @SET #I14 = SUBSTR #L1:3-6,3-6: ------------------------------ (09)
   1.     @STATUS = I -------------------------------------------------- (10)
#I00= 0000000000 #I01= 0000000000 #I02= 0000000000
#I03= 0000000000 #I04= 0000000000 #I05= 0000000000
#I06= 0000000003 #I07= 0000000123 #I08= 0000111111
#I09= 0000000001 #I10= 0034562111 #I11= 0000001605
```

```
#I12= 0123123123 #I13= 0005555555 #I14= 0012341234
#I15= 0000000000 #I16= 0000000000 #I17= 0000000000
#I18= 0000000000 #I19= 0000000000 #I20= 0000000000
 1.
```

(01) Zeile 2, Zeichenfolgevariable #S2, Ganzzahlvariable #I6 und Zeilennummervariable #L1 werden mit Werten versehen.

(02) X'F0F1F2F3' entspricht der abdruckbaren Zahl 0123. Diese Zahl wird der Ganzzahlvariablen #I7 zugewiesen.

(03) B'11110001'*6 ist sowohl gleichwertig mit X'F1'*6 als auch mit '1'*6 als auch mit der abdruckbaren Zahl 111111. Diesen Wert erhält #I8.

(04) Die Spalte 3 der Zeile 2 enthält die abdruckbare Ziffer 1. Somit erhält #I9 den Wert 1.

(05) Setzt man die abdruckbaren Ziffern der Spalten 5 bis 8 sowie der Spalte 4 und dreimal der Spalte 3 von Zeile 2 hintereinander, erhält man die abdruckbare Zahl 34562111 und somit den Wert von #I10.

(06) Die Spalten 4 bis 5 und 7 bis 8 der Zeichenfolgevariablen #S2 ergeben die abdruckbare Zahl 1605, die der Ganzzahlvariablen #I11 als Wert zugewiesen wird.

(07) '123'*3 ist gleichwertig mit '123123123'. Diese Zahl wird #I12 zugewiesen.

(08) #I6 hat den Wert 3. Damit wird mit #S5-#I6 die Zeichenfolgevariable #S2 angesprochen. Schreibt man siebenmal den Inhalt der Spalte 8 der Zeichenfolgevariablen #S2 hintereinander, ergibt dies die Zahl 55555555 und damit den Wert #I13.

(09) Über Zeilennummervariable #L1 wird die Zeile 2 angesprochen. Die Spalten 3 bis 6 von Zeile 2 ergeben bei zweimaliger Verwendung die abdruckbare Zahl 12341234 und damit den Wert von #I14.

(10) Die Inhalte der Ganzzahlvariablen #I0 bis #I20 werden ausgegeben.

Beispiel 3

Den Inhalt einer Zeilennummervariablen in eine Ganzzahl umwandeln und diese einer Ganzzahlvariablen als Wert zuweisen.

```
   1.     @SET #L0  = 0.0001 ------------------------------------------ (01)
   1.     @SET #I15 = #L0 --------------------------------------------- (02)
   1.     @STATUS   = #I15 -------------------------------------------- (03)
#I15= 0000000001
   1.     @SET #L1  = 55.5555 ----------------------------------------- (04)
   1.     @SET #I16 = #L1 --------------------------------------------- (05)
   1.     @STATUS   = #I16 -------------------------------------------- (06)
#I16= 0000555555
   1.     @SET #L2  = 9999.9999 --------------------------------------- (07)
   1.     @SET #I17 = #L2 --------------------------------------------- (08)
   1.     @STATUS   = #I17 -------------------------------------------- (09)
#I17= 0099999999
1.
```

(01) Die Zeilennummervariable #L0 erhält den Wert 0.0001

(02) Die Ganzzahlvariable #I15 erhält den 10000-fachen Wert von #L0, also den Wert 1.

(03) Der Inhalt von #I15 wird ausgegeben.

(04) #L1 wird der Wert 55.5555 zugewiesen.

(05) #I16 wird der 10000-fache Wert von #L1 zugewiesen.

(06) Der Wert von #I16, nämlich 555555, wird ausgegeben.

(07) #L2 erhält den Wert 9999.9999.

(08) #I17 wird der Wert 99999999 zugewiesen.

(09) Der Inhalt von #I17 wird ausgegeben.

Beispiel 4

Länge einer Zeile einer Ganzzahlvariablen zuweisen.

```
   1.     @CREATE 15.34: 'AB'*23
   1.     @PRINT 15.34                                          ------------ (01)
  15.3400 ABABABABABABABABABABABABABABABABABABABABABABABAB
   1.     @SET #I10 = LENGTH 15.34
   1.     @STATUS = #I10              ------------------------------------ (02)
#I10= 0000000046
   1.     @DELETE 15.34
   1.     @SET #I10 = LENGTH 15.34    ------------------------------------ (03)
   1.     @STATUS = #I10
#I10= 0000000000
```

(01) Die Zeile 15.34 wird erzeugt und ausgegeben.

(02) #I10 wird die Länge der Zeile 15.34 zugewiesen und der Wert von #I10 wird ausgegeben.

(03) Die Zeile 15.34 wird gelöscht.
Wieder wird #I10 die Länge der (inzwischen gelöschten) Zeile 15.34 zugewiesen und der Wert von #I10 ausgegeben.

Beispiel 5

Einer Ganzzahlvariablen den EBCDI-Code einer Zeichenfolge zuweisen.

```
   1.     @CREATE 1: '1234567'   ---------------------------------------- (01)
   1.     @CREATE #S5: 'CBA'
   1.     @SET #I0 = STRING X'34'
   1.     @SET #I1 = STRING 'A'
   1.     @SET #I2 = STRING X'7D5'
   1.     @SET #I3 = STRING B'00001111'   ------------------------------ (02)
   1.     @SET #I4 = STRING 1:3,2:
   1.     @SET #I5 = STRING #S5:3,3:
   1.     @SET #I6 = STRING 1:2-3,1:
   1.     @STATUS = I ------------------------------------------------- (03)
#I00= 0000000052 #I01= 0000000193 #I02= 0000002005
#I03= 0000000015 #I04= 0000062450 #I05= 0000049601
#I06= 0015922161 #I07= 0000000000 #I08= 0000000000
#I09= 0000000000 #I10= 0000000000 #I11= 0000000000
#I12= 0000000000 #I13= 0000000000 #I14= 0000000000
#I15= 0000000000 #I16= 0000000000 #I17= 0000000000
#I18= 0000000000 #I19= 0000000000 #I20= 0000000000
```

(01) Zeile 1 und Zeichenfolgevariable #S5 werden mit einem Inhalt versehen.

(02) Auf mehrere Arten werden den Ganzzahlvariablen #I0 bis #I6 Werte zugewiesen.

(03) Der Inhalt aller Ganzzahlvariablen wird ausgegeben.

@SET (Format 2) Versorgen von Zeichenfolgevariablen mit Werten

Mit diesem Format von @SET wird:

- einer Zeichenfolgevariablen eine Zeichenfolge zugewiesen. Der bisherige Wert der Zeichenfolgevariablen wird überschrieben, d.h. er wird gelöscht
- der Inhalt einer Ganzzahlvariablen, eine Zeilennummer oder der Name einer Zeichenfolgevariablen zugewiesen. Der EDT schreibt den Wert so in die Zeichenfolgevariable, wie er intern dargestellt ist
- der Inhalt einer Ganzzahlvariablen in die entsprechende abdruckbare Zahl konvertiert und diese ab einer anzugebenden Spalte in einer Zeichenfolgevariablen abgelegt
- das abdruckbare Bild der in einer Zeilennummervariablen festgehaltenen Zeilennummer ab einer anzugebenden Spalte in einer Zeichenfolgevariablen abgelegt
- der Name einer Zeichenfolgevariablen ab einer bestimmten Spalte in einer Zeichenfolgevariablen abgelegt

Operation	Operanden	F-Modus / L-Modus / @PROC
@SET	str-ln { = string | = **I**NTERNAL { int-var | ln | str-var } | [,cl] = **C**HAR { int-var | ln-var | str-var } }	

str-ln Gibt direkt oder indirekt eine der 21 Zeichenfolgevariablen #S0, #S1,..., #S20 an. Dies geschieht entweder durch die direkte Angabe dieser Zeichenfolgevariablen oder dadurch, daß man zu einer Zeichenfolgevariablen eine Distanz addiert oder von ihr subtrahiert. Beispielsweise ist #S0 + 3L gleichwertig mit #S3 oder #S16-2L gleichwertig mit #S14. Dasselbe gelingt auch im Zusammenspiel mit Ganzzahlvariablen. Es habe #I12 den Inhalt 10. Dann ist #S3 + #I12 gleichwertig mit #S3 + 10L, also gleichwertig mit #S13.

cl Spalte, ab der in die Zeichenfolgevariable geschrieben werden soll. Der Standardwert für cl ist 1.
Wird einer Zeichenfolgevariablen der Inhalt einer Ganzzahlvariablen zugewiesen, darf cl den Wert 246 nicht überschreiten.
Wird einer Zeichenfolgevariablen der Inhalt einer Zeilennummervariablen

zugewiesen, darf cl den Wert 248 nicht überschreiten.
Wird einer Zeichenfolgevariablen der Inhalt einer Zeichenfolgevariablen zugewiesen, darf cl den Wert 253 nicht überschreiten.

string — Zeichenfolge.
string kann angegeben werden:
– direkte Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich). string wird der Zeichenfolgevariablen zugewiesen.

int-var — Ganzzahlvariable (#I0 bis #I20), deren Inhalt der EDT str-ln als Wert zuweisen soll.

ln — Zeilennummer (z.B. 5). Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

ln-var — Zeilennummervariable (#L0 bis #L20), deren Wert str-ln zugewiesen wird. Der größte Wert von ln-var ist 9999.9999, der kleinste 0.0001.

str-var — Zeichenfolgevariable (#S0 bis #S20), deren Namen str-ln als Wert zugewiesen werden soll.

INTERNAL

– Der Inhalt einer Ganzzahlvariablen ist intern als 4 Byte lange Dualzahl abgespeichert. Diese Dualzahl wird unverändert in die ersten 4 Byte der Zeichenfolgevariablen geschrieben. Hat beispielsweise die Ganzzahlvariable den Wert 359, ist die entsprechende Dualzahl B'0000 0000 0000 0000 0000 0001 0110 0111', d.h. X'00 00 01 67'. Genau dieser Wert wird in die ersten 4 Byte der Zeichenfolgevariablen geschrieben.
– Eine Zeilennummer steht intern in einem 4 Byte (8 Halbbyte) langen Feld. Jedes Halbbyte enthält eine der 8 Dezimalziffern, aus denen eine Zeilennummer besteht. Der Inhalt des 4 Byte langen Feldes wird unverändert in die ersten 4 Byte der Zeichenfolgevariablen geschrieben. Beispielsweise wird die Zeilennummer 47.11 intern als X'00471100' gespeichert. Genau dieser Wert wird in die ersten 4 Byte der Zeichenfolgevariablen geschrieben.
– Der Name einer Zeichenfolgevariablen ist intern durch 4 abdruckbare Zeichen dargestellt. Diese 4 Zeichen schreibt der EDT unverändert in die ersten 4 Byte von str-ln. Ist beispielsweise als Name der Zeichenfolgevariablen #S1 angegeben, schreibt der EDT in die ersten 4 Byte von str-ln den Wert #S01, d.h. X'7BE2F0F1'.

CHAR Weist den EDT an,

- den Inhalt einer Ganzzahlvariablen in die entsprechende abdruckbare Zahl zu konvertieren und in einer Zeichenfolgevariablen abzulegen. Die Konvertierung führt zu einer 11 Zeichen langen abdruckbaren Zahl, wobei das erste Zeichen entweder ein Leerzeichen oder ein Minuszeichen ist, je nachdem ob die Ganzzahl positiv oder negativ ist.
- die Zeilennummer, die in einer Zeilennummervariablen steht, zu konvertieren und in der Form IIII.IIII ab der Spalte cl in der Zeichenfolge- variablen str-ln abzulegen. Um in jedem Fall 9 abdruckbare Zeichen zu erhalten, wird gegebenenfalls linksbündig mit Leerzeichen (nicht abdruckbaren Nullen) aufgefüllt.
- den Namen einer Zeichenfolgevariablen (str-var) in die durch str-ln angegebene Zeichenfolgevariable zu schreiben. Dabei wird der Name der Zeichenfolgevariablen in die Form #Sdd umgewandelt.
  dd kann die Werte 00,01,... bis 20 annehmen.

Beispiel 1

Einer Zeichenfolgevariablen eine Zeichenfolge zuweisen.

```
1.     @CREATE 1: '-'*40 ------------------------------------------ (01)
1.     @SET #S1 = 1
1.     @SET #S2 = '2'
1.     @SET #S3 = '3'*15
1.     @SET #S4 = B'11110000'*20   ---------------------------------- (02)
1.     @SET #S5 = X'F4'*16
1.     @SET #S6 = 'HA'*7
1.     @SET #S7 = #S6:2
1.     @SET #S9-1L = #S1+4L
1.     @SET #I7 = 10 ----------------------------------------------- (03)
1.     @SET #S19-#I7 = #S18-#I7 ------------------------------------ (04)
1.     @PRINT #S1.-#S9 --------------------------------------------- (05)
  #S01 ----------------------------------------
  #S02 2
  #S03 333333333333333
  #S04 00000000000000000000
  #S05 4444444444444444
  #S06 HAHAHAHAHAHAHA
  #S07 A
  #S08 4444444444444444
  #S09 4444444444444444
```

(01) Die Zeile 1 wird angelegt und erhält als Inhalt 40mal das Zeichen –.

(02) Den Zeichenfolgevariablen #S1 bis #S7 werden Werte zugewiesen:
#S1 wird der Inhalt von Zeile 1 zugewiesen.
#S2 wird der Inhalt ’2’ zugewiesen.
#S3 wird als Inhalt 15mal das Zeichen ’3’ zugewiesen.
#S4 wird als Inhalt 20mal das Zeichen B’11110000’, d.h. X’F0’, also das Zeichen ’0’ zugewiesen.
#S5 wird als Inhalt 16mal das Zeichen X’F4’, also das Zeichen ’4’ zugewiesen.
#S6 wird als Inhalt die 7malige Verkettung der Zeichenfolge ’HA’ zugewiesen.
#S7 wird der auf Spalte 2 stehende Inhalt von #S6 zugewiesen.
#S9 - 1L, also #S8, wird der Inhalt von #S1 + 4L, also #S5, zugewiesen.

(03) Der Ganzzahlvariablen #I7 wird der Wert 10 zugewiesen.

(04) #S19 - #I7, also #S19 - 10L, also #S9 wird der Inhalt von #S18 - #I7, also #S8 zugewiesen.

(05) Die Inhalte der Zeichenfolgevariablen #S1 bis #S9 werden ausgegeben.

Beispiel 2

Einer Zeichenfolgevariablen den Inhalt einer Ganzzahlvariablen, eine Zeilennummer und den Namen einer Zeichenfolgevariablen zuweisen (internes Format).

```
  1.     @SET #I0 = 19 ------------------------------------------------ (01)
  1.     @SET #S0 = INTERNAL #I0 -------------------------------------- (02)
  1.     @SET #I1 = -1 ------------------------------------------------ (03)
  1.     @SET #S1 = INTERNAL #I1 -------------------------------------- (04)
  1.     @SET #S3 + #I1 = INTERNAL 25.4356 ---------------------------- (05)
  1.     @SET #L3 = 2222.2222 ----------------------------------------- (06)
  1.     @SET #S10 - 7L = INTERNAL #L3 + 11.11 ------------------------ (07)
  1.     @78.7878 ----------------------------------------------------- (08)
78.7878 @SET #S4 = INTERNAL * -------------------------------------- (09)
78.7878 @SET #S5 = INTERNAL #S0 ------------------------------------ (10)
78.7878 @SET #S6 = INTERNAL #S19 ----------------------------------- (11)
78.7878 @PRINT #S0.-#S6 X,#S5,#S6 ---------------------------------- (12)
   #S00 00000013
   #S01 FFFFFFFF
   #S02 00254356
   #S03 22333322
   #S04 00787878
   #S05 7BE2F0F0
   #S06 7BE2F1F9
   #S05 #S00
   #S06 #S19
78.7878
```

(01) Der Ganzzahlvariablen #I0 wird der Wert 19, also X’13’, zugewiesen.

(02) #S0 soll die interne Darstellung von #I0, also X'00000013', erhalten.

(03) #I1 wird der Wert –1 zugewiesen. Hexadezimal X'FFFFFFFF'.

(04) #S1 wird die Interndarstellung von #I1 zugewiesen.

(05) #S3 + #I1 ist gleichwertig mit #S3 – 1L und dieses wiederum mit #S2. Somit soll #S2 die Interndarstellung der Zeilennummer 25.4356 zugewiesen werden.

(06) Der Zeilennummervariable #L3 wird der Wert 2222.2222 zugewiesen.

(07) #S10 – 7L ist gleichwertig mit #S3. Dieser Zeichenfolgevariablen wird die Interndarstellung der Zeilennummer #L3 + 11.11, also von 2222.2222 + 11.11 = 2233.3322, zugewiesen.

(08) 78.7878 wird zur aktuellen Zeilennummer.

(09) #S4 wird die Interndarstellung der aktuellen Zeilennummer zugewiesen.

(10) #S5 wird die Interndarstellung des Namens der Zeichenfolgevariablen #S0 zugewiesen.

(11) #S6 wird die Interndarstellung des Namens der Zeichenfolgevariablen #S19 zugewiesen.

(12) Die Inhalte der Zeichenfolgevariablen #S0, ..., #S6 sollen hexadezimal ausgegeben werden. Zusätzlich sollen die Inhalte von #S5 und #S6 in Normaldarstellung ausgegeben werden.

Beispiel 3

Einer Zeichenfolgevariablen den Inhalt einer Ganzzahlvariablen (abdruckbare Zahl) zuweisen.

```
1.     @SET #I0 = 11223344
1.     @SET #I1 = 55667788    ---------------------------------------- (01)
1.     @SET #I2 = -99999999
1.     @SET #I3 = 5
1.     @CREATE #S16 : '@'*40 ---------------------------------------- (02)
1.     @SET #S16 = CHAR #I0 ----------------------------------------- (03)
1.     @SET #S17-1L,14 = CHAR #I1 ----------------------------------- (04)
1.     @SET #S11 + #I3,27 = CHAR #I2 -------------------------------- (05)
1.     @PRINT #S16
  #S16  0011223344@@ 0055667788@@-0099999999@@@ -------------------- (06)
```

(01) Die Ganzzahlvariablen #I0, #I1, #I2 und #I3 werden mit Werten versehen.

(02) Die Zeichenfolgevariable #S16 soll als Inhalt 40 mal das Zeichen '@' erhalten.

(03) Die in #I0 enthaltene Ganzzahl soll abdruckbar gemacht und ab Spalte 1 in #S16 abgelegt werden.

(04) Die in #I1 enthaltene Ganzzahl wird konvertiert und ab Spalte 14 in #S17 - 1L, also in #S16, abgelegt.

(05) Wegen #I3 = 5 ist #S11 + #I3 identisch mit #S11 + 5L, also mit #S16. Dort soll ab Spalte 27 beginnend die Zahl in abdruckbarer Form abgelegt werden, die in #I2 festgehalten ist.

(06) Der Inhalt der Zeichenfolgevariablen #S16 wird ausgegeben.

Beispiel 4

Einer Zeichenfolgevariablen den Inhalt einer Ganzzahlvariablen (abdruckbare Zahl) über einen Stellungsparameter zuweisen.

```
   1.     @SET #I5 = 18
   1.     @SET #I6 = -27      ------------------------------------------ (01)
   1.     @SET #I7 = 333333
   1.     @PROC 7 ------------------------------------------------------ (02)
   1.     @ @PARAMS &INTVAR -------------------------------------------(03)
   2.     @ @SET #S0 = CHAR &INTVAR ------------------------------------ (04)
   3.     @ @ON #S0:2-2: DELETE '0' ------------------------------------ (05)
   4.     @ @IF .TRUE. GOTO 3 -----------------------------------------(06)
   5.     @ @CREATE #S1: 'DER INHALT VON &INTVAR IST ',#S0 ------------- (07)
   6.     @ @PRINT #S1 N ----------------------------------------------- (08)
   7.     @END --------------------------------------------------------- (09)
   1.     @DO 7(#I5) --------------------------------------------------- (10)
DER INHALT VON #I5 IST  18
   1.     @DO 7(#I6) --------------------------------------------------- (11)
DER INHALT VON #I6 IST -27
   1.     @DO 7(#I7) --------------------------------------------------- (12)
DER INHALT VON #I7 IST  333333
```

(01) Die Ganzzahlvariablen #I5, #I6 und #I7 werden mit Werten versehen.

(02) Es wird in die Arbeitsdatei 7 umgeschaltet.

(03) Der Stellungsparameter &INTVAR wird definiert.

(04) #S0 wird der abdruckbare Wert einer Ganzzahlvariablen zugewiesen, welche mit @DO anzugeben ist.

(05) Da in Spalte 1 von #S0 das Vorzeichen liegt, liegen ab Spalte 2 die führenden (abdruckbaren) Nullen. Die erste davon wird gelöscht.

(06) Damit werden nacheinander alle führenden Nullen gelöscht.

(07) Können keine führenden abdruckbaren Nullen mehr gefunden werden, wird die Zeichenfolgevariable #S1 erzeugt.

(08) #S1 wird ohne Zeilennummer ausgegeben.

(09) Es wird in Arbeitsdatei 0 zurückgekehrt.

(10) Die Prozedur in Arbeitsdatei 7 wird ausgeführt und als Parameter die Ganzzahlvariable #I5 übergeben.

(11) Nun wird #I6 übergeben.

(12) Zuletzt wird #I7 übergeben.

Beispiel 5

Einer Zeichenfolgevariablen die in einer Zeilennummervariablen festgehaltene Zeilennummer (abdruckbare Zahl) zuweisen.

```
1.     @SET #I0 = 6
1.     @SET #I1 = 28
1.     @SET #L0 = 0.0045       ---------------------------------------- (01)
1.     @SET #L1 = 9999.9999
1.     @SET #L2 = 55.55
1.     @CREATE #S5 : '*'*40 ---------------------------------------- (02)
1.     @SET #S5,3 = CHAR #L0 ---------------------------------------- (03)
1.     @SET #S3+2L,15 = CHAR #L1 ---------------------------------- (04)
1.     @SET #S11-#I0,#I1 = CHAR #L2 ------------------------------- (05)
1.     @PRINT #S5 ------------------------------------------------- (06)
  #S05 **   0.0045***9999.9999****  55.5500****
```

(01) Den Ganzzahlvariablen #I0 und #I1 und den Zeilennummervariablen #L0, #L1 und #L2 werden Werte zugewiesen.

(02) #S5 wird angelegt und besteht aus 40 Zeichen *.

(03) Ab Spalte 3 von #S5 wird das abdruckbare Bild der in #L0 enthaltenen Zeilennummer abgelegt.

(04) Mit #S3 + 2L wird wieder #S5 angesprochen. Somit wird dort ab Spalte 15 die abdruckbare in #L1 enthaltene Zeilennummer abgelegt.

(05) #S11 – #I0 ist gleichwertig mit #S11 – 6L und dieses wiederum mit #S5. Dort wird ab Spalte #I1, also ab Spalte 28, das abdruckbare Bild der in #L2 enthaltenen Zeilennummer abgelegt.

(06) Der Inhalt der Zeichenfolgevariablen #S5 wird ausgegeben.

Beispiel 6

Einer Zeichenfolgevariablen den Namen einer Zeichenfolgevariablen zuweisen.

```
1.     @SET #I0 = 19
1.     @SET #S0 = INTERNAL #I0
1.     @PRINT #S0 X
  #S00 00000013                                    ------------------- (01)
1.     @SET #S3 = '*'*40
1.     @PRINT #S3
  #S03 ****************************************
1.     @SET #S3 , 10 = CHAR #S0
1.     @PRINT #S3                                   ------------------- (02)
  #S03 *********#S00**************************
```

(01) Die Ganzzahlvariable #I0 und die Zeichenfolgevariablen #S0 und #S3 werden mit Werten versehen und ausgegeben.

(02) Mit @SET Format 2 wird in #S3 ab Spalte 10 der Name von #S0 in der Form #S00 gebracht und ausgegeben.

Beispiel 7

Einer Zeichenfolgevariablen den Namen einer Zeichenfolgevariablen über Stellungs- und Schlüsselwortparameter zuweisen.

```
1.     @CREATE #S18 : '-'*40 ---------------------------------------- (01)
1.     @PRINT #S18
  #S18 ---------------------------------------- ------------------- (02)
1.     @PROC 7 ------------------------------------------------------ (03)
1.     @ @PARAMS &STRVAR1,&STRVAR2,&COLUMN=1 ------------------------ (04)
2.     @ @SET &STRVAR1,&COLUMN = CHAR &STRVAR2 ---------------------- (05)
3.     @ @PRINT &STRVAR1 -------------------------------------------- (06)
4.     @END --------------------------------------------------------- (07)
1.     @DO 7(#S18,#S16) --------------------------------------------- (08)
  #S18 #S16-----------------------------------
1.     @DO 7(#S17+1L,#S20,COLUMN=20) -------------------------------- (09)
  #S18 #S16---------------#S20-----------------
1.     @DO 7(#S18,#S18,COLUMN=14) ----------------------------------- (10)
  #S18 #S16---------#S18--#S20-----------------
1.     @SET #I12 = 26 ----------------------------------------------- (11)
1.     @SET #I13 = 12
1.     @DO 7(#S6+#I13,#S0,COLUMN=#I12) ------------------------------ (12)
  #S18 #S16---------#S18--#S20--#S00-----------
```

(01) Die Zeichenfolgevariable #S18 erhält den Inhalt 40 mal das Zeichens –.

(02) Der Inhalt von #S18 wird ausgegeben.

(03) Es wird in Arbeitsdatei 7 umgeschaltet.

(04) Zwei Stellungsparameter (&STRVAR1 und &STRVAR2) und ein Schlüsselwortparameter (&COLUMN) werden definiert.

(05) Beim Durchlaufen dieser Prozedur wird an dieser Stelle von @SET, Format 4 Gebrauch gemacht. Die Werte für die Parameter müssen beim entsprechenden @D0 7 mitgegeben werden.

(06) An dieser Stelle wird der Inhalt einer Zeichenfolgevariablen ausgegeben werden.

(07) Es wird in Arbeitsdatei 0 zurückgeschaltet.

(08) Die Anweisungen der Prozedur in Arbeitsdatei 7 werden durchgeführt. Dabei wird &STRVAR1 durch #S18 und &STRVAR2 durch #S16 ersetzt. Für &COLUMN wird der Standardwert (vorbelegt mit 1) angenommen.

(09) Mit #S17 + 1L wird die Zeichenfolgevariable #S18 angesprochen. Für den Schlüsselwortparameter &COLUMN wird der Wert 20 verwendet.

(10) Auffällig ist hierbei, daß die Werte der beiden Stellungsparameter identisch sind. Damit wird also der Name der Zeichenfolgevariablen selbst abgelegt.

(11) #I12 und #I13 werden mit Werten versehen.

(12) #S6 + #I13 ist gleichwertig mit #S6 + 12L, also mit #S18. Für den Schlüsselwortparameter &COLUMN wird die Ganzzahlvariable #I12 eingesetzt.

@SET (Format 3) Versorgen von Zeilennummervariablen mit Werten

Mit diesem Format wird:

– einer Zeilennummervariablen eine Zeilennummer zugewiesen
– die in einer Ganzzahlvariablen enthaltene Ganzzahl in eine Zeilennummer umgewandelt und einer Zeilennummervariablen zugewiesen
– eine abdruckbare Zahl einer Zeilennummervariablen als Zeilennummer zugewiesen
– die interne Darstellung einer Zeichenfolge einer Zeilennummervariablen zugewiesen. Diese interne Darstellung besteht aus 8 hexadezimalen Ziffern. Zwischen der 4. und der 5. Ziffer wird der Dezimalpunkt der Zeilennummer angenommen

Operation	Operanden	F-Modus / L-Modus / @PROC
@SET	ln-var = { ln | int-var | **S**UBSTR string | **ST**RING string }	

ln-var Zeilennummervariable (#L0 bis #L20), der ein Wert zugewiesen werden soll. Der größte Wert von ln-var ist 9999.9999, der kleinste 0.0001.

ln Zeilennummer (z.B. 5). Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

int-var Ganzzahlvariable (#I0 bis #I20), deren Inhalt ln-var in umgewandelter Form zugewiesen werden soll. Der Wertebereich für int-var beträgt 1 bis 99999999, dies führt zu den Zeilennummern 0.0001 bis 9999.9999.
Bei der Konvertierung wird die Ganzzahl durch 10000 dividiert und der entsprechende Wert der Zeilennummernvariablen zugewiesen. Dies ist notwendig, um jede mögliche Zeilennummer erhalten zu können.

string Zeichenfolge.
string kann angegeben werden:
– direkte Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

SUBSTR string
Der EDT weist der angegebenen Zeilennummervariablen eine abdruckbare Zahl als Wert zu.
string muß ein abdruckbare Zahl größer 0 sein.

STRING string

Der EDT interpretiert die interne Darstellung einer Zeichenfolge als Zeilennummer.

Der EBCDI-Code von string wird der Zeilennummervariablen zugeordnet. Es dürfen nur Zeichen angegeben werden, deren EBCDI-Code aus Ziffern besteht.
Besteht die Zeichenfolge aus weniger als 4 Zeichen, wird linksbündig mit Nullen aufgefüllt. Bei mehr als 4 Zeichen werden lediglich die ersten 4 Zeichen berücksichtigt.

Bei der Verwendung von Ganzzahlvariablen innerhalb des Operanden ln ist zu beachten, daß beispielsweise für den Ausdruck #L5 = #L6 + #I7 nicht die Summe der Werte von #L6 und #I7 gebildet wird.

Der Inhalt von #I7 sei n.

Der Zeilennummervariablen #L5 wird dann die Nummer der n-ten Zeile hinter der durch #L6 angegebenen Zeilen zugewiesen. Hat beispielsweise #I7 den Wert 7, ist #L5 = #L6 + #I7 gleichwertig mit #L5 = #L6 + 7L.

Beispiel *1*

Einer Zeilennummervariablen eine Zeilennummer zuweisen.

```
  1.     @DELETE
  1.     @1:AAAAA
  2.     @C1-10 TO 2.5 (0.5)
 11.5    @PRINT
  1.0000 AAAAA
  2.5000 AAAAA
  3.0000 AAAAA
  3.5000 AAAAA
  4.0000 AAAAA
  4.5000 AAAAA
  5.0000 AAAAA
  5.5000 AAAAA                  ------------------------------------- (01)
  6.0000 AAAAA
  6.5000 AAAAA
  7.0000 AAAAA
  7.5000 AAAAA
  8.0000 AAAAA
  8.5000 AAAAA
  9.0000 AAAAA
  9.5000 AAAAA
 10.0000 AAAAA
 10.5000 AAAAA
 11.5    @SET #I10 = 4
```

```
  11.5    @SET #I11 = 7
  11.5    @SET #L0 = 1 ------------------------------------------------ (02)
  11.5    @SET #L1 = #L0 + #L0 ---------------------------------------- (03)
  11.5    @SET #L2 = % ------------------------------------------------ (04)
  11.5    @SET #L3 = #L2 + 1 ------------------------------------------ (05)
  11.5    @SET #L4 = #L2 + 1L ----------------------------------------- (06)
  11.5    @SET #L5 = #L3 + #L4 ---------------------------------------- (07)
  11.5    @SET #L6 = #L4 + #I11 --------------------------------------- (08)
  11.5    @SET #L7 = #L6 - #I10 --------------------------------------- (09)
  11.5    @SET #L8 = $-2L --------------------------------------------- (10)
  11.5    @SET #L9 = $+$ ---------------------------------------------- (11)
  11.5    @SET #L10 = $-% --------------------------------------------- (12)
  11.5    @STATUS = L ------------------------------------------------- (13)
```

```
#L00=   1.0000    #L01=   2.0000    #L02=   1.0000
#L03=   2.0000    #L04=   2.5000    #L05=   4.5000
#L06=   6.0000    #L07=   4.0000    #L08=   9.5000
#L09=  21.0000    #L10=   9.5000    #L11=   0.0000
#L12=   0.0000    #L13=   0.0000    #L14=   0.0000
#L15=   0.0000    #L16=   0.0000    #L17=   0.0000
#L18=   0.0000    #L19=   0.0000    #L20=   0.0000
```

(01) Mehrere Zeilen werden in die virtuelle Datei gebracht. Die Ganzzahlvariablen #I10 und #I11 werden mit Werten versehen.

(02) #L0 wird der Wert 1 zugewiesen.

(03) #L1 wird der Wert #L0 + #L0, nämlich 1 + 1 = 2 zugewiesen.

(04) #L2 erhält den Wert, der gleich der ersten vorhandenen Zeilennummer ist, also 1.

(05) #L3 erhält den Wert #L2 + 1, also 1 + 1, also 2.

(06) #L4 erhält den Wert #L2 + 1L, womit die auf die Zeilennummer #L2 folgende Zeilennummer angesprochen wird. Da #L2 den Wert 1 hat und die auf Zeilennummer 1 folgende Zeilennummer gleich 2.5 ist, wird dies der Wert von #L4.

(07) #L5 wird die Summe von #L3 und #L4 zugewiesen und erhält damit den Wert 2 + 2.5 = 4.5.

(08) #L6 erhält den Wert #L4 + #I11.

Es wird nicht die Summe von 2.5 und 7 gebildet, sondern diese Angabe entspricht der Angabe #L4 + 7L, was der 7. Zeilennummer entspricht, die auf #L4 folgt. Da #L4 = 2.5 ist, wird #L6 der Wert 6 zugewiesen.

(09) Wie unter (08) wird #L7 der Wert #L6 - #I10 zugewiesen, was mit #L6 - 4L identisch ist, also 4.

(10) #L8 wird die Nummer der drittletzten Zeile zugewiesen. Denn mit $ spricht man die letzte, mit $-1L die vorletzte und mit $-2L die drittletzte Zeile an. Damit erhält #L8 den Wert 9.5.

(11) #L9 wird der Wert $ + $ - also 10.5 + 10.5 = 21 zugewiesen.

(12) #L10 wird der Wert $ - % = 10.5 - 1 = 9.5 zugewiesen.

(13) Die Werte der Zeilennummervariablen werden ausgegeben.

Beispiel 2

Einer Zeilennummervariablen den Inhalt einer Ganzzahlvariablen als Zeilennummer zuweisen.

```
   1.     @SET #I0 = 1
   1.     @SET #L0 = #I0   ------------------------------------------- (01)
   1.     @STATUS  = #L0
#L00=   0.0001
   1.     @SET #I1 = 20000
   1.     @SET #L1 = #I1    ------------------------------------------- (02)
   1.     @STATUS  = #L1
#L01=   2.0000
   1.     @SET #I2 = 12345678
   1.     @SET #L2 = #I2       ---------------------------------------- (03)
   1.     @STATUS  = #L2
#L02=1234.5678
   1.
```

(01) #I0 wird der Wert 1 zugewiesen.

#L0 wird der Wert #I0/10000, also 0.0001 zugewiesen und der Wert von #L0 wird ausgegeben.

(02) #I1 wird der Wert 20000 zugewiesen.

#L1 wird der Wert #I1/10000, also 2.0000 zugewiesen und der Wert von #L1 wird ausgegeben.

(03) #I2 wird der Wert 12345678 zugewiesen.

#L2 wird der Wert #I2/10000, also 1234.5678 zugewiesen und der Wert von #L2 wird ausgegeben.

Beispiel 3

Einer Zeilennummervariablen eine abdruckbare Zahl als Zeilennummer zuweisen.

```
   1.     @CREATE 1: 'ABC1.23.4.5.67' ---------------------------------- (01)
   1.     @CREATE #S0: '     .5'
   1.     @SET #L0 = SUBSTR X'F1F2F3' ---------------------------------- (02)
   1.     @SET #L1 = SUBSTR '123.456' ---------------------------------- (03)
   1.     @SET #L2 = SUBSTR B'11110111'*4 ------------------------------ (04)
   1.     @SET #L3 = SUBSTR 1:11-13,6-7: ------------------------------- (05)
   1.     @SET #L4 = SUBSTR #S0 ---------------------------------------- (06)
   1.     @SET #L5 = SUBSTR '9'*3 -------------------------------------- (07)
   1.     @STATUS = L -------------------------------------------------- (08)
#L00= 123.0000    #L01= 123.4560    #L02=7777.0000
#L03=   5.6230    #L04=   0.5000    #L05= 999.0000
#L06=   0.0000    #L07=   0.0000    #L08=   0.0000
#L09=   0.0000    #L10=   0.0000    #L11=   0.0000
#L12=   0.0000    #L13=   0.0000    #L14=   0.0000
#L15=   0.0000    #L16=   0.0000    #L17=   0.0000
#L18=   0.0000    #L19=   0.0000    #L20=   0.0000
   1.
```

(01) Zeile 1 und Zeichenfolgevariable #S0 werden angelegt.

(02) X’F1F2F3’ ist identisch mit der abdruckbaren Zeichenfolge ’123’. Diese Zahl wird #L0 zugeordnet.

(03) #L1 wird die Zahl 123.4560 zugeordnet.

(04) B’11110111’*4 ist gleichwertig mit X’F7’*4, also mit ’7777’. Damit erhält #L2 den Wert 7777.0000.

(05) Verkettet man die Inhalte der Spalten 11, 12, 13, 6, 7 von Zeile 1 in dieser Reihenfolge, ergibt sich der Wert ’5.623’. Somit erhält #L3 den Wert 5.6230.

(06) #S0 hat den Inhalt ’.5’. Da Leerzeichen bei der Konvertierung nicht berücksichtigt werden, wird #L4 der Wert 0.5000 zugeordnet.

(07) ’9’*3 ist gleichwertig mit ’999’. Damit erhält #L5 den Wert 999.0000.

(08) Die Werte aller Zeilennummervariablen werden ausgegeben.

Beispiel 4

Einer Zeilennummervariablen die interne Darstellung einer Zeichenfolge als Zeilennummer zuweisen.

```
   1.     @CREATE 1: X'01020304050607' ------------------------------------ (01)
   1.     @CREATE #S0: B'01111001'*20 ------------------------------------- (02)
   1.     @SET #L0 = STRING X'34' ----------------------------------------- (03)
   1.     @SET #L1 = STRING B'00000110' ----------------------------------- (04)
   1.     @SET #L2 = STRING ' '*2 ----------------------------------------- (05)
   1.     @SET #L3 = STRING 1:5,4: ---------------------------------------- (06)
   1.     @SET #L4 = STRING 1:1-2,2-4: ------------------------------------ (07)
   1.     @SET #L5 = STRING #S0:6: ---------------------------------------- (08)
   1.     @STATUS = L ----------------------------------------------------- (09)
#L00=   0.0034    #L01=   0.0006    #L02=   0.4040
#L03=   0.0504    #L04= 102.0203    #L05=   0.0079
#L06=   0.0000    #L07=   0.0000    #L08=   0.0000
#L09=   0.0000    #L10=   0.0000    #L11=   0.0000
#L12=   0.0000    #L13=   0.0000    #L14=   0.0000
#L15=   0.0000    #L16=   0.0000    #L17=   0.0000
#L18=   0.0000    #L19=   0.0000    #L20=   0.0000
   1.
```

(01) Die Zeile 1 wird mit einem 7 Zeichen langen (nicht abdruckbaren) Inhalt versehen.

(02) Die Zeichenfolgevariable #S0 besteht aus 20mal demselben, nicht abdruckbaren Zeichen X'79'.

(03) #L0 wird die Zeilennummer 0.0034 zugewiesen.

(04) #L1 wird die Zeilennummer 0.0006 zugewiesen, da B'00000110' = X'06' ist.

(05) #L2 wird die Zeilennummer 0.4040 zugewiesen, da ' '*2 = X'40'*2 = X'4040' ist.

(06) Da in den Spalten 5 und 4 der Zeile 1 die Werte X'05' bzw. X'04' enthalten sind, wird #L3 der Wert 0.0504 zugewiesen.

(07) Betrachtet man in dieser Reihenfolge die Inhalte der Spalten 1, 2, 2, 3, 4 von Zeile 1, ergibt dies den Wert X'0102020304'. Da maximal nur 4 Spalten für den Inhalt von #L4 genommen werden können, wird die letzte genannte Spalte nicht berücksichtigt. Somit erhält #L4 den Wert 102.0203.

(08) In #S0 Spalte 6 steht der Wert B'01111001', also X'79'. Damit wird #L5 der Wert 0.0079 zugewiesen.

(09) Die Werte aller Zeilennummervariablen werden ausgegeben.

@SET (Format 4) Werte in Zeilen ablegen

Mit diesem Format von @SET wird:

– der Inhalt einer Ganzzahlvariablen in eine abdruckbare Form umgewandelt und in der durch die Zeilennummervariable angegebenen Zeile abgelegt
– der Name einer Zeichenfolgevariablen ab einer bestimmten Spalte in eine Zeile geschrieben. Die Nummer ist über eine Zeilennummervariable anzugeben
– der Inhalt einer Zeilennummervariablen abdruckbar gemacht und in einer Zeile abgelegt, deren Nummer in einer 2. Zeilennummervariablen angegeben sein muß

Operation	Operanden	F-Modus / L-Modus / @PROC
@SET	ln-var [,cl] = **C**HAR { int-var / str-var / ln-var1 }	

ln-var Zeilennummervariable (#L0 bis #L20). Der EDT soll in die Zeile, deren Nummer in der Zeilennummervariablen steht, den Inhalt von int-var, str-var oder ln-var1 in abdruckbarer Form schreiben. Der größtmögliche Wert von ln-var ist 9999.9999, der kleinste ist 0.0001.

cl Spalte, ab der in die Zeile geschrieben werden soll. Der Standardwert von cl ist 1. Wird ein cl > 1 angegeben, wird die Zeile bis zur Spalte cl mit Leerzeichen aufgefüllt.
– Wird eine Ganzzahl in eine Zeile geschrieben, darf cl den Wert 246 nicht überschreiten.
– Wird der Name einer Zeichenfolgevariablen in eine Zeile geschrieben, darf cl den Wert 253 nicht überschreiten.
– Wird der Wert einer Zeilennummervariablen in eine Zeile geschrieben, darf cl den Wert 248 nicht überschreiten.

CHAR int-var Weist den EDT an, die in int-var enthaltene Ganzzahl abdruckbar zu machen und ab der Spalte cl in der Zeile abzulegen, deren Nummer in ln-var enthalten ist.
int-var ist eine der Ganzzahlvariablen (#I0 bis #I20).
Die Konvertierung der Ganzzahlvariablen führt zu einer 11 Zeichen langen abdruckbaren Zahl, wobei das 1. Zeichen das Vorzeichen enthält und die nachfolgenden Zeichen die Ziffern der Zahl darstellen. Ist die Zahl nicht negativ, wird dies durch ein vorangestelltes Leerzeichen zum Ausdruck gebracht, im anderen Fall durch ein Minuszeichen –. Da in jedem Fall 11 Zeichen erzeugt werden, werden die Stellen nach dem Vorzeichen gegebenenfalls mit führenden abdruckbaren Nullen aufgefüllt.

CHAR str-var Weist den EDT an, den Namen einer Zeichenfolgevariablen in eine anzugebende Zeile zu schreiben.
str-var ist eine der 21 Zeichenfolgevariablen (#S0 bis #S20). Die für str-var gemachte Angabe schreibt der EDT in die über ln-var angegebene Zeile. Der Name einer Zeichenfolgevariablen wird umgewandelt in die Form #Sdd. Dabei kann dd die Werte 00,01... bis 20 annehmen.

CHAR ln-var1 Weist den EDT an, den Wert der Zeilennummervariablen ln-var1 abdruckbar zu machen und in einer ln-var1 ist eine der Zeilennummervariablen (#L0 bis #L20), deren Wert in eine Zeile geschrieben werden soll.
Die Konvertierung des Wertes einer Zeilennummervariablen führt immer zu 9 abdruckbaren Zeichen der Form IIII.IIII wobei jedes I eine abdruckbare Ziffer darstellt. Hierbei werden führende Nullen durch Leerzeichen ersetzt.

Beispiel 1

Den Inhalt einer Ganzzahlvariablen in der durch eine Zeilennummervariablen angegebenen Zeile ablegen.

```
1.     @SET #I18 = 1234 ------------------------------------------- (01)
1.     @SET #I19 = 5678
1.     @SET #I20 = #I18 + #I19
1.     @CREATE 5: 'XXXXXXXXXXX PLUS YYYYYYYYYYY GIBT ZZZZZZZZZZZ'
1.     @SET #L18 = 5
1.     @SET #L18 = CHAR #I18 -------------------------------------- (02)
1.     @PRINT 5
5.0000  0000001234 PLUS YYYYYYYYYYY GIBT ZZZZZZZZZZZ
1.     @SET #L18,17 = CHAR #I18 ----------------------------------- (03)
1.     @PRINT 5
5.0000  0000001234 PLUS 0000001234Y GIBT ZZZZZZZZZZZ
1.     @SET #L18,18 = CHAR #I19
1.     @PRINT 5
5.0000  0000001234 PLUS  0000005678 GIBT ZZZZZZZZZZZ
1.     @SET #L18,35 = CHAR #I20 ----------------------------------- (04)
1.     @PRINT 5
5.0000  0000001234 PLUS  0000005678 GIBT  0000006912
1.
```

(01) Die Ganzzahlvariablen #I18, #I19 und #I20 sowie die Zeile 5 und die Zeilennummervariable #L18 werden mit Werten versehen.

(02) Bei Spalte 1 beginnend wird in der über #L18 angesprochenen Zeile die in #I18 festgehaltene Zahl abdruckbar abgelegt.

(03) Nun wird in der über #L18 angesprochenen Zeile ab Spalte 17 beginnend das abdruckbare Bild von #I18 abgelegt.

(04) Zuletzt wird in der über #L18 angesprochenen Zeile ab Spalte 35 beginnend das abdruckbare Bild von #I20 abgelegt.

Beispiel 2

Den Namen einer Zeichenfolgevariablen in der durch eine Zeilennummervariablen angegebenen Zeile ablegen.

```
  1.     @SET #L1 = 666.66 ------------------------------------------ (01)
  1.     @SET #L1 = CHAR #S20 --------------------------------------- (02)
  1.     @PRINT #L1
666.6600 #S20
  1.     @SET #L1,30 = CHAR #S13 ------------------------------------ (03)
  1.     @SET #L1,15 = CHAR #S7 ------------------------------------- (04)
  1.     @PRINT #L1
666.6600 #S20      #S07           #S13
  1.
```

(01) #L1 wird der Wert 666.6600 zugewiesen.

(02) Die nicht existierende Zeile 666.6600 wird erzeugt und erhält (ab Spalte 1 beginnend) den Inhalt '#S20'.

(03) Ab Spalte 30 sollen in der über #L1 angesprochenen Zeile die Zeichen '#S13' abgelegt werden.

(04) Die interne Darstellung der Zeilennummer von #S7 ist '#S07'. Diese Zeichenfolge wird in Zeile 666.66 ab Spalte 15 gebracht.

Beispiel 3

Den Inhalt einer Zeilennummervariablen (ln-var2) in der durch eine Zeilennummervariablen (ln-var1) angegebenen Zeile ablegen.

```
  1.     @DELETE
  1.     @SET #L3 = 57.45   ----------------------------------------- (01)
  1.     @SET #L4 =99.99
  1.     @SET #L3 = CHAR #L4 ---------------------------------------- (02)
  1.     @PRINT
 57.4500   99.9900
  1.     @SET #L3,20 = CHAR #L3 ------------------------------------- (03)
  1.     @PRINT
 57.4500   99.9900            57.4500
  1.
```

(01) Den Zeilennummervariablen #L3 und #L4 werden Werte zugewiesen.

(02) Der EDT schreibt in eine bislang noch nicht existierende Zeile ab Spalte 1 den Inhalt von #L4 in abdruckbarer Form.

(03) Der EDT schreibt in eine bereits existierende Zeile ab Spalte 20 den Inhalt von #L3 in abdruckbarer Form. Zu beachten ist, daß ln-var1 und ln-var2 - wie hier gezeigt - gleich sein können.

@SET (Format 5) Datum und Uhrzeit

Mit diesem Format von @SET wird:

– entweder das Datum oder die Uhrzeit ab einer gewünschten Spalte in einer Zeichenfolgevariablen abgelegt
– entweder das Datum oder die Uhrzeit in der Zeile abgelegt, deren Zeilennummer über eine Zeilennummervariable angegeben ist

Operation	Operanden	F-Modus / L-Modus / @PROC
@SET	{ str-ln / ln-var } [,cl] = { **D**ATE [**I**SO[**4**]] / **T**IME }	

str-ln — Gibt direkt oder indirekt eine der 21 Zeichenfolgevariablen an (#S0 bis #S20). Dies erfolgt entweder durch die explizite Angabe dieser Zeichenfolgevariablen oder dadurch, daß man zu einer Zeichenfolgevariable eine Distanz addiert oder von ihr subtrahiert. Beispielsweise ist #S0 + 3L gleichwertig mit #S3 oder #S16 - 2L gleichwertig mit #S14. Dasselbe gelingt auch im Zusammenspiel mit Ganzzahlvariablen. Es habe #I12 den Inhalt 10. Dann ist #S3 + #I12 gleichwertig mit #S3 + 10L, also gleichwertig mit #S13.

ln-var — Zeilennummervariable (#L0 bis #L20). In die über ln-var angegebene Zeile soll der EDT das Datum bzw. die Uhrzeit schreiben. Der größte Wert von ln-var ist 9999.9999, der kleinste 0.0001.

cl — Spalte, ab der das Datum bzw. die Uhrzeit abgelegt werden soll. Standardwert von cl ist 1. cl darf den Wert 246 bzw. 251 nicht überschreiten.

DATE — Weist den EDT an, das aktuelle Datum in der angegebenen Zeichenfolgevariablen bzw. Zeile abzulegen. Dabei benutzt der EDT die Form mm/dd/yyjjj, wenn ISO nicht angegeben wird. Durch mm wird der Monat, durch dd der Tag, durch yy das Jahr und durch jjj der Jahrestag angegeben.

ISO — Der EDT gibt das Datum im Format yy-mm-ddjjj aus.

ISO4 — Der EDT gibt das Datum im Format yyyy-mm-ddjjj aus.

TIME — Der EDT legt die Uhrzeit in der angegebenen Zeichenfolgevariablen bzw. Zeile ab. Dabei benutzt der EDT die Form hhmmss. Durch hh werden die Stunden, durch mm die Minuten und durch ss die Sekunden angegeben.

Beispiel 1

Datum in der Form mm/dd/yyjjj und Uhrzeit in der Form hhmmss in einer Zeichenfolgevariablen ablegen.

```
   1.     @PROC 3 ------------------------------------------------------ (01)
   1.     @ @DELETE #S0.-.#S4 ------------------------------------------ (02)
   2.     @ @SET #S0 = DATE -------------------------------------------- (03)
   3.     @ @SET #S1 = TIME -------------------------------------------- (04)
   4.     @ @CREATE #S2: '*'*10, ' HEUTE IST DER ',#S0:4-5: ------------ (05)
   5.     @ @CREATE #S2: #S2,'.',#S0:1-2:,'.',#S0:7-8:,' ','*'*11
   6.     @ @CREATE #S3: '*'*15,' ES IST JETZT ','*'*16
   7.     @ @CREATE #S4: '***** ',#S1:1-2:,' UHR ',#S1:3-4:,' MINUTEN UND '
   8.     @ @CREATE #S4: #S4,#S1:5-6:,' SEKUNDEN *****'
   9.     @ @PRINT #S2.-.#S4N ------------------------------------------ (06)
  10.     @END --------------------------------------------------------- (07)
   1.     @DO 3
********** HEUTE IST DER 29.09.90 ********** ---------------------------- (08)
*************** ES IST JETZT ****************
***** 14 UHR 25 MINUTEN UND 14 SEKUNDEN *****
```

(01) Es wird in Arbeitsdatei 3 umgeschaltet.

(02) Die Inhalte der Zeichenfolgevariablen #S0, #S1, #S2, #S3 und #S4 werden bei Ausführung der Prozedurdatei 3 an dieser Stelle gelöscht, d.h. sie werden genau ein Leerzeichen zum Inhalt haben.

(03) In #S0 soll ab Spalte 1 das Datum stehen.

(04) In #S1 soll ab Spalte 1 die Uhrzeit stehen.

(05) Die Zeichenfolgevariablen #S2, #S3 und #S4 werden benutzt, um das aktuelle Datum nebst Uhrzeit mit entsprechendem Text festzuhalten.

(06) Bei Ausführung der Prozedurdatei 3 werden an dieser Stelle die Inhalte der Zeichenfolgevariablen #S2, #S3 und #S4 ausgegeben.

(07) Es wird in Arbeitsdatei 0 zurückgekehrt.

(08) Bei Ausführung der Prozedur in Arbeitsdatei 3 werden Datum und Uhrzeit bis auf die Sekunde genau in aufbereiteter Form ausgegeben.

Beispiel 2

Datum in der Form yyyy-mm-ddjjj in einer über eine Zeilennummervariable angegebenen Zeile ablegen.

```
1.     @CREATE 2.5: '-'*40 ------------------------------------------ (01)
1.     @SET #L13 = 2.5 ---------------------------------------------- (02)
1.     @SET #L13,15 = DATE ISO4 ------------------------------------- (03)
1.     @PRINT 2.5
2.5000 --------------1992-05-05126-------------   ------------------- (04)
1.
```

(01) Die Zeile 2.5 wird neu angelegt.

(02) Der Zeilennummervariablen #L13 wird der Wert 2.5 zugewiesen.

(03) In der über #L13 angesprochenen Zeile soll ab Spalte 15 das Datum abgelegt werden.

(04) Die Zeile 2.5 wird ausgegeben.

@SET Bestimmen der neuen aktuellen Zeilennummer und Schrittweite

@SET Format 6

Mit diesem Format von @SET bestimmt man eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite. Darüber hinaus kann eine Eingabe für den EDT gemacht werden.

Operation	Operanden	F-Modus / L-Modus
@SET	ln [(inc)] [:text]	

Im F-Modus muß das Schlüsselwort SET angegeben werden. Nur @ln ohne SET wird mit einer Fehlermeldung abgewiesen.

ln — Neue aktuelle Zeilennummer, z.B. 5.
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. Fehlt inc, wird mit ln implizit auch die neue aktuelle Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.
ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

inc — Neue aktuelle Schrittweite.
Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. Fehlen ln und inc, ist die Schrittweite 1.

text — Beliebige Zeichenfolge. Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein EDT-Anweisungssymbol und kein Benutzerfluchtsymbol, werden die dem: folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   - text steht am Anfang der durch ln angegebenen Zeile;
   - vorhandene Tabulatorzeichen werden berücksichtigt;
   - die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht.
2. das EDT-Anweisungssymbol, werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   - kein EDT-Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   - das EDT-Anweisungssymbol, wird text als Textzeile wie bei 1. behandelt.
3. das Benutzerfluchtsymbol, wird die externe Anweisungsroutine ausgeführt (siehe @USE).

@SET erzeugt gegenüber @ keinen neuen Kellereintrag (siehe auch @).

Beispiel

```
  23.00 ..........................................................................
set 105(0.3):read 'bsp.text'.....................................0000.00:001(1)
```

Die Zeile 105 soll angelegt werden. Da der Text hinter dem Doppelpunkt nicht mit @ beginnt, wird er als Inhalt der Zeile 105 abgelegt.

```
 105.00 READ 'BSP.TEXT'..........................................................
 106.00 .........................................................................
```

```
delete ; lower on ; set 105(0.3):@read 'bsp.text'.................0105.00:001(1)
```

Die Arbeitsdatei soll gelöscht und ab Zeile 105 mit der Schrittweite 0.3 die Datei 'BSP.TEXT' eingelesen werden. Mit @LOWER ON werden die in der Datei enthaltenen Kleinbuchstaben ausgegeben.

```
 105.00 Mit @SET bestimmt man eine neue aktuelle Zeilennummer und eine neue.....
 105.30 aktuelle Schrittweite. Darueber hinaus kann eine Eingabe fuer den.......
 105.60 EDT gemacht werden. Diese Eingabe kann eine EDT-Anweisung sein, ........
 105.90 die sofort ausgefuehrt wird. Die Eingabe kann auch eine beliebige.......
 106.20 andere Zeichenfolge sein, die dann Inhalt der neuen Zeile wird. ........
 106.50 Anschliessend wird die aktuelle Zeilennummer um die aktuelle............
 106.80 Schrittweite erhoeht. ..................................................
 107.80 .........................................................................
```

@SETF Sichtfenster positionieren

Mit @SETF kann man

– die Arbeitsdatei wechseln

– in einer Arbeitsdatei vertikal oder horizontal positionieren:
  – in der aktuellen Arbeitsdatei
  – in einer beliebigen Arbeitsdatei (0 bis 9) ohne Verlassen der aktuellen Arbeitsdatei
  – in einer beliebigen Arbeitsdatei mit Wechseln in diese Arbeitsdatei
  – in allen Arbeitsdateien (0 bis 9) gleichzeitig

Operation	Operanden	F-Modus / L-Modus
@SETF #	[{fwkfv / **GL**OBAL / (fwkfnr)}] [{ln / vpos}] [{:cl: / hpos}]	

darf nur im F-Modus eingegeben werden. Es muß mindestens ein Operand angegeben werden.

fwkfv — Arbeitsdatei ($0..$9), in der positioniert wird. Die aktuelle Arbeitsdatei bleibt unverändert.

fwkfnr — Arbeitsdatei (0..9), in der positioniert wird. Es wird vor der Positionierung in die Arbeitsdatei verzweigt.
Wird nur (fwkfnr) angegeben, wird nur die aktuelle Arbeitsdatei gewechselt.

GLOBAL — Es wird in allen 10 Arbeitsdateien (0 bis 9) gleichzeitig positioniert.
In @DO- und @INPUT-Prozeduren wird @SETF GLOBAL mit einer Fehlermeldung abgewiesen.

vpos — Relative vertikale Positionieranweisung. Es können +[n], ++, –[n], – – sowie +([m,..]), ++([m,..]), –([m,..]) und – –([m,..]) angegeben werden.
m ist eine der 9 möglichen Satzmarkierungen, zu der positioniert werden soll. Es können mehrere Satzmarkierungen angegeben werden.
Markierungen mit Sonderfunktionen (z.B. Markierung 15 für Schreibschutz) werden hier nicht ausgewertet.

cl — Absolutangabe der horizontalen Position. Spaltennummer, die die erste Spalte in der angegebenen Arbeitsdatei bilden soll. Werte zwischen 1 und 256 sind zulässig.

hpos — Relative horizontale Positionieranweisung. Es können >[n], <[n] und << angegeben werden.

Wird nur der erste Operand angegeben, wird bei

– @SETF (fwkfnr) nur die aktuelle Arbeitsdatei gewechselt. Darunterliegende Arbeitsdateien werden zuvor beendet.

– @SETF fwkfv in der angegebenen Arbeitsdatei auf die erste Zeile und die erste Spalte positioniert.

– @SETF in der aktuellen Arbeitsdatei auf die erste Zeile und die erste Spalte positioniert.

– @SETF GLOBAL in allen Arbeitsdateien auf die erste Zeile und erste Spalte positioniert.

In Prozeduren (@DO- und @INPUT-Prozeduren) kann nach @ON das Bildschirmfenster mit @SETF ? positioniert werden.

Die früheren Operanden FIRST und LAST müssen durch die symbolischen Zeilennummern % und $ ersetzt werden.

Beispiel

```
  23.00 ..............................................................................
setf (1) 7 :3: ..................................................0000.00:001(2)
```

Der EDT soll das Arbeitsfenster in die Arbeitsdatei 1 auf die Zeilennummer 7, Spalte 3 positionieren.

```
   7.00 Mit @SETF kann man.....................................................
   8.00  .....................................................................
   9.00 1. die Arbeitsdatei (0 bis 9) wechseln,.................................
  10.00  .....................................................................
  11.00 2. in Arbeitsdateien vertikal oder horizontal positionieren:.............
  12.00  .....................................................................
  13.00     -  in der aktuellen Arbeitsdatei,.................................
  14.00     -  in einer beliebigen Arbeitsdatei (0 bis 9) ohne.................
  15.00        Verlassen der aktuellen Arbeitsdatei,..........................
  16.00     -  in einer beliebigen Arbeitsdatei mit Wechseln in.................
  17.00        diese Arbeitsdatei,...........................................
  18.00     -  in allen Arbeitsdateien (0 bis 9) gleichzeitig. .................
  19.00  .....................................................................
  20.00 ..............................................................................
```

@SETJV Jobvariable katalogisieren und Wert zuweisen

Mit @SETJV kann:

– eine Jobvariable in den Katalog eingetragen werden,
– einer Jobvariablen ein Wert zugewiesen werden.

Ist das Subsystem „Jobvariablen-Support“ nicht installiert, wird diese Anweisung mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@SETJV	{ [string1] = string2 [,...] / string1 }	

string1 Zeichenfolge, die einen vollqualifizierten Jobvariablen-Namen angibt.
Ist die Jobvariable noch nicht im Katalog vorhanden, wird sie mit den Standardfunktionen des DCLJV-Makros katalogisiert. string1 kann angegeben werden:
– explizit als Zeichenfolge in Hochkomma,
– implizit über eine Zeilennummer, eine Zeilennummervariable (#L0-#L20) oder eine Zeichenfolgevariable (#S0-#S20) (jeweils mit Spaltenbereich möglich).
Der Jobvariablen wird der Kettungsname *EDTLINK zugeordnet, sie kann über ihn angesprochen werden.
Wird string1 nicht angegeben, wird die Jobvariable über den Kettungsnamen *EDTLINK angesprochen. In diesem Fall muß string2 angegeben werden.

string2[,...] Zeichenfolge, die der Jobvariablen als Wert zugewiesen werden soll.
Die Länge des Jobvariablen-Wertes wird durch die Länge der aufbereiteten Zeichenfolge bestimmt. Ist die aufbereitete Zeichenfolge länger als 256 Zeichen, werden nur die ersten 256 Zeichen als Wert zugewiesen. Der EDT gibt dann eine Fehlermeldung aus.
Wird string2 mehrmals angegeben, wird in der angegebenen Reihenfolge verkettet. string2 kann angegeben werden:
– explizit als Zeichenfolge in Hochkomma,
– implizit über eine Zeilennummer, eine Zeilennummervariable (#L0-#L20) oder eine Zeichenfolgevariable (#S0-#S20), jeweils mit Spaltenbereich.

Tabulatorzeichen werden von @SETJV nicht verarbeitet.

Beispiel: siehe Beispiel zu @STAJV

@SETLIST Erweitern einer Listenvariablen

Mit @SETLIST kann eine Liste

- gelöscht werden (/FREE-VAR)
- um ein einzelnes Element erweitert werden
- um alle Datenzeilen in einem Zeilenbereich erweitert werden
- um die markierten Datenzeilen in einem Zeilenbereich erweitert werden

Operation	Operanden	F-Modus / L-Modus
@**SETL**IST	string { [range*] [**MA**RK [m]] | str-var } [:col:][,] [**M**ODE=**A**PPEND | **P**REFIX | **O**VERWRITE]	

string — Zeichenkette, die den Namen der Listenvariablen angibt.
string kann angegeben werden:
- explizit als Zeichenfolge in Hochkomma
- implizit über eine Zeilennummer, eine Zeilennummernvariable oder eine Zeichenfolgevariable, jeweils mit Spaltenbereich.

range* — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereiche (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.

Ist kein Zeilenbereich angegeben, werden alle Zeilen der aktuellen Arbeitsdatei berücksichtigt.

MARK — Es sollen nur die markierten Zeilen des angegebenen Zeilenbereiches übernommen werden.

m m = 1,...,9
Nummer der Satzmarkierung, die berücksichtigt werden soll.
Ist m nicht angegeben, so werden nur Datenzeilen mit Markierung 1 berücksichtigt.

str-var Name der Zeichenfolgevariablen, deren Inhalt als Element aufgenommen werden soll.

col Eine oder mehrere Spalten oder Spaltenbereiche
Wiederholungen und Überlappungen von Spalten und Spaltenbereichen sind erlaubt.
Ist dieser Operand nicht angegeben, wird die ganze Zeile berücksichtigt.

MODE legt fest, auf welche Art die Liste erweitert werden soll.

=PREFIX Die Liste wird am Anfang erweitert, d.h. die Zeichenfolgen werden der Reihe nach als Elemente vor dem ersten Element eingefügt.
Ist die Datei leer oder im angegebenen Zeilenbereich keine Zeile vorhanden, wird die Meldung % EDT2903 FILE IS EMPTY ausgegeben.

=APPEND Die Liste wird am Ende erweitert, d.h. die Zeichenfolgen werden als Elemente nach dem letzten Element angefügt.
Ist die Datei leer oder im angegebenen Zeilenbereich keine Zeile vorhanden, wird die Meldung % EDT2903 FILE IS EMPTY ausgegeben.

=OVERWRITE
Vor dem Schreiben wird ein /FREE-VARIABLE ausgeführt. Dann werden die Elemente der Reihe nach wie bei APPEND angefügt.
Ist die Datei leer oder im angegebenen Zeilenbereich keine Zeile vorhanden, wird nur /FREE-VARIABLE ausgeführt und im F-Modus die Meldung % EDT0211 ausgegeben.

Ist der MODE-Operand nicht angegeben, wird MODE=APPEND angenommen.

Das Komma vor dem Operanden MODE muß zur Unterscheidung von MARK angegeben werden, wenn außer dem Listennamen oder einer etwaigen Bereichs-Angabe kein anderer Operand angegeben wird, z.B. @SETLIST 'LISTE',M=O

Die Listenvariable muß vorher deklariert werden mit
/DECLARE-VARIABLE chars, MULTIPLE-ELEMENTS=LIST und TYPE=STRING oder TYPE=ANY.

Die Länge eines erzeugten Listenelementes ergibt sich aus der Anzahl der im Spaltenbereich angegebenen Zeichen der Zeile oder der Zeichenfolgevariablen.
Ist die Zeile oder die Zeichenfolgevariable kürzer, wird mit Leerzeichen aufgefüllt.
Ist kein Splatenbereich angegeben, ergibt sich die Länge aus der Länge der Zeile oder der Zeichenfolgevariablen.

Bei der Angabe von range* bzw. col können sich Zeilennummern bzw. Spaltennummern wiederholen, was zu einem mehrmaligen Einlesen der entsprechenden Zeilen bzw. Spalten führt.

Vergabe der Zeilennummern

Die Zeilennummern werden abhängig von der aktuellen Zeilennummer und der aktuellen Schrittweite vergeben. Bei leerer Arbeitsdatei sind die aktuelle Zeilennummer und die aktuelle Schrittweite standardmäßig 1. Beide Werte können mit @SET ln (inc) geändert werden (siehe @SET, Format 6).

@SETSW Schalter setzen

Mit @SETSW werden Benutzer- und Auftragsschalter gesetzt oder zurückgesetzt.

Operation	Operanden	F-Modus / L-Modus
@SETSW	{ [**ON**=] / **OF**F= } [**U**] int1 [-int2] [,...]	

ON = Die angegebenen Schalter werden gesetzt.

OFF = Die angegebenen Schalter werden zurückgesetzt.

int1 Schalter, der gesetzt oder zurückgesetzt werden soll. Für int1 ist eine Ganzzahl zwischen 0 und 31 oder eine Ganzzahlvariable (#I0-#I20) anzugeben.

Falls vor int1 der Parameter U angegeben wird, bezieht sich die Angabe auf einen Benutzerschalter der eigenen Benutzerkennung, sonst auf einen Auftragsschalter.

int2 Alle Schalter, die zwischen int1 und int2 liegen, werden gesetzt bzw. zurückgesetzt. Für int2 ist eine Ganzzahl zwischen 0 und 31 oder eine Ganzzahlvariable (#I0-#I20) anzugeben. Die Schalterart ergibt sich aus der Angabe für int1.

Die Hauptanwendung von @SETSW liegt in EDT-Prozeduren. Durch @IF, Format 4 kann überprüft werden, ob ein Auftragsschalter oder ein Benutzerschalter der eigenen Kennung gesetzt ist oder nicht.

Es ist möglich, innerhalb einer @SETSW-Anweisung sowohl Benutzerschalter als auch Auftragsschalter zu setzen bzw. zurückzusetzen.

Beispiel 1

@SETSW ON = U1-6,12-20,U31 (Die Benutzerschalter 1 bis 6 und 31 und die Auftragsschalter 12 bis 20 werden gesetzt).

Beispiel 2

```
   1.     @SET #S2 = 'SCHALTER 15 IST AUS'
   1.     @SET #S3 = 'SCHALTER 15 IST AN'
   1.     @PROC 9
   1.     @ @IF ON = 15 GOTO 4 ------------------------------------- (01)
   2.     @ @PRINT #S2 N
   3.     @ @RETURN
   4.     @ @PRINT #S3 N
   5.     @END
   1.     @SETSW OFF = 15 ------------------------------------------ (02)
   1.     @DO 9 ---------------------------------------------------- (03)
SCHALTER 15 IST AUS
   1.     @SETSW ON = 15 ------------------------------------------- (04)
   1.     @DO 9 ---------------------------------------------------- (05)
SCHALTER 15 IST AN
   1.
```

(01) In der Arbeitsdatei 9 wird eine Prozedur abgelegt, die bei gesetztem Auftragsschalter 15 die Zeichenfolgevariable #S3 ausgibt, andernfalls aber die Zeichenfolgevariable #S2.

(02) Der Auftragsschalter 15 wird zurückgesetzt.

(03) Die Prozedur von Arbeitsdatei 9 wird ausgeführt.

(04) Auftragsschalter 15 wird gesetzt.

(05) Die Prozedur von Arbeitsdatei 9 wird ausgeführt.

@SETVAR Deklarieren einer S-Variablen und Wertzuweisung

Mit @SETVAR kann man:

– eine S-Variable deklarieren (TYPE=ANY)
– einer deklarierten S-Variablen einen Wert zuweisen (TYPE=STRING, TYPE=INTEGER)

In Systemen, in denen das Subsystem SDF-P nicht installiert ist, wird @SETVAR mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@**SETV**AR	{ string [=string1 | =int-var] / **SYSEDT** } [,**M**ODE=**A**NY | **N**EW | **U**PDATE]	

string Zeichenfolge, die den Namen einer einfachen S-Variablen angibt.

string1 Zeichenfolge, die der mit string angegebenen S-Variablen vom Typ STRING als Wert zugewiesen wird.
string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

int-var Ganzzahlvariable (#I0-#I20), deren Inhalt der über string angegebenen S-Variablen vom Typ INTEGER als Wert zugewiesen wird.

Wird weder string1 noch int-var angegeben, so wird eine S-Variable vom Typ ANY deklariert.

SYSEDT Den S-Variablen SYSEDT-S00 bis SYSEDT-S20 werden die Inhalte der Zeichenfolgevariablen #S00 bis #S20 zugewiesen. Wird der Operand SYSEDT angegeben, werden keine Meldungen über Erfolg oder Mißerfolg bei der Wertzuweisung ausgegeben. Es werden keine Fehlerschalter gesetzt.

MODE legt fest, ob die S-Variable schon deklariert sein soll.

= ANY Einer existierenden oder einer neuen S-Variable wird ein Wert zugewiesen.

= NEW Die S-Variable darf noch nicht deklariert sein. Wenn int-var angegeben ist, wird die S-Variable mit mit TYPE=INTEGER definiert, sonst mit TYPE=STRING.

= UPDATE Die S-Variable muß schon deklariert sein.

Soll eine S-Variable mit SCOPE=TASK erzeugt werden, so muß dies mit dem SDF-P-Kommando DECLARE-VARIABLE erfolgen.

@SHOW Ausgeben eines Inhaltsverzeichnisses

Mit @SHOW Format 1 wird das Inhaltsverzeichnis einer Programm-Bibliothek oder eines Benutzerkataloges ausgegeben.

Mit @SHOW Format 2 wird eines Liste der im System möglichen Coded Character Set Namen (CCSN) ausgegeben.

@SHOW (Format 1) Ausgeben eines Inhaltsverzeichnisses

Die Auflistung aller Elemente der Bibliothek oder des Benutzerkataloges (Inhaltsverzeichnis) erfolgt wahlweise:
- in die Arbeitsdatei 9,
- in die aktuelle Arbeitsdatei,
- im L-Modus am Bildschirm,
- im Stapelbetrieb auf SYSOUT.

Operation	Operanden	F-Modus / L-Modus
@SHOW	{ **L**IBRARY=path1 [,[**TYP**E=]elemtyp]] **TYP**E=elemtyp **F**ILES[=path2] } [**T**O] [ln [(inc)]] [**S**HORT | **LONG**]	

path1 Name der Programm-Bibliothek. path1 kann auch als Zeichenfolgevariable angegeben werden.

elemtyp Typ des Elements. elemtyp kann auch als Zeichenfolgevariable angegeben werden.
Entsprechend der Angabe wird das Inhaltsverzeichnis dieses Elementtyps ausgegeben. Wird kein Elementtyp angegeben, wird das gesamte Inhaltsverzeichnis der Bibliothek ausgegeben.

Zulässige Typangaben: S, M, P, J, D, X, R, C, H, L, U, F, *STD und freie Typnamen mit entsprechendem Basistyp.

Wird ein freier Typnamen verwendet, so liegt es in der Verantwortung des Benutzers, daß der zugehörige Basistyp einem zulässigen Typ S, M, P, J, D, X, R, C, H, L, U oder F entspricht.

Typ	Elementinhalt
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats
R	Bindemodule
C	Lademodule
H	von H-Assembler erzeugt
L	vom BINDER erzeugt
U	von IFG erzeugt
F	von IFG erzeugt

*STD
Typ S ist die Voreinstellung nach Aufruf des EDT. Der mit @PAR voreingestellte Elementtyp hat hier keine Wirkung.

ln

Nummer der Zeile, ab der das Ergebnis in die aktuelle Arbeitsdatei geschrieben werden soll. ln kann auch als symbolische Zeilennummer oder als Zeilennummervariable angegeben werden.
Die Nummern der folgenden Zeilen des Empfangsbereichs errechnet der EDT, indem er die jeweilige Zeilennummer um die für den Empfangsbereich geltende Schrittweite erhöht. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999.
Mit ln wird implizit die Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.

Ist ln nicht angegeben, wird das Ergebnis
- im L-Modus am Bildschirm ausgegeben,
- im Stapelbetrieb auf SYSOUT ausgegeben,
- im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht.

Falls die Ausgabe einer Informationszeile (@PAR INFORMATION = ON) eingeschaltet ist, wird die Zeile mit den Bezeichnungen der Einträge nicht als erste Zeile in die Arbeitsdatei geschrieben, sondern als nicht überschreibbare Überschriftenzeile ausgegeben.

inc

Inkrement zur Berechnung der Zeilennummern. Ist inc nicht angegeben, wird die mit ln implizit gegebene Schrittweite verwendet.

path2 Namen der Dateien, die ausgegeben werden sollen. path2 kann auch als Zeichenfolgevariable angegeben werden.

SHORT Standardwert.
Für Bibliothekselemente:
Die Zeilenlänge beträgt 72. Bei Typnamen > 4, bei Elementnamen > 32 bzw. Versionsbezeichnungen > 12 Zeichen besteht ein Eintrag aus 2 Zeilen.

Für Dateien:
Entspricht der Ausgabe bei @FSTAT mit Operand SHORT.

LONG Für Bibliothekselemente:
Jeder Eintrag besteht nur aus 1 Zeile. Die Zeilenlänge beträgt 120. LONG eignet sich besonders zur Verarbeitung in EDT-Prozeduren.

Spalte	Überschrift	Bedeutung
1-8	TYP	Elementtyp
9-72	ELEMENT	Elementname
73-96	VERSION	Versionsnummer
97-106	VAR	Variantennummer
107-120	DATE	Datum der letzten Änderung

Der Name der Bibliothek wird in der Überschrift mitausgegeben.

Für Dateien:
Entspricht der Ausgabe bei @FSTAT mit Operand LONG.

Angezeigt wird nur die höchste Variante jedes Elements. Die Variante erfüllt die Funktion eines Schreibzugriffzählers. Bei jedem schreibenden Zugriff auf ein Bibliothekselement wird die Variantennummer dieses Elements um 1 erhöht.

Hat ein Element verschiedene Versionsnummern, werden alle Versionen angezeigt.

@SHOW wird abgewiesen, wenn

- keine Elemente eines angegebenen Elementtyps existieren,
- die angegebene Bibliothek nicht existiert oder keine Programm-Bibliothek ist.
- in der Arbeitsdatei 9 ein Bibliothekselement mit @OPEN eröffnet wurde und noch geöffnet ist.

i

- Die aktuelle Zeilennummer wird verändert, wenn eine Zeile angelegt wurde, deren Nummer größer als die bisherige höchste Zeilennummer ist.
- Ein Element mit der höchsten Versionsnummer erhält im Inhaltsverzeichnis für die Version ein @ eingetragen.

Beispiel

```
   1.00 MIT @SHOW WIRD DAS INHALTSVERZEICHNIS EINER BIBLIOTHEK AUSGEGEBEN. .....
   2.00 DER NAME DER BIBLIOTHEK IST IM OPERANDEN DER ANWEISUNG ANZUGEBEN. ......
   3.00 ..........................................................................









show library=edt.lib.bsp 100(10) ..............................0001.00:001(0)
```

Das gesamte Inhaltsverzeichnis der Bibliothek EDT.LIB.BSP soll in der aktuellen Arbeitsdatei ab Zeilennummer 100 mit der Schrittweite 10 ausgegeben werden.

```
   1.00 MIT @SHOW WIRD DAS INHALTSVERZEICHNIS EINER BIBLIOTHEK AUSGEGEBEN. .....
   2.00 DER NAME DER BIBLIOTHEK IST IM OPERANDEN DER ANWEISUNG ANZUGEBEN. ......
 100.00 TYP   E L E M E N T                      VERSION       VAR    DATE
 110.00  C    PROG1                             @             0004   1989-01-11
 120.00  D    E.TEXT1                           @             0006   1988-12-05
 130.00  D    E.TEXT2                           @             0013   1988-12-05
 140.00  D    E.TEXT3                           100           0011   1988-12-13
 150.00  D    E.TEXT3                           101           0121   1988-12-20
 160.00  D    E.TEXT3                           102           0007   1989-01-11
 170.00  J    ASSEMB                            @             0099   1989-01-11
 180.00  J    FILE-TRANSFER                     @             0045   1989-01-11
 190.00  J    PROC1                             1             0001   1988-12-05
 200.00  D    DAS-IST-EIN-ELEMENT-MIT-EINEM-SE  @             0000   1994-08-17
 210.00       HR-LANGEN-NAMEN
 220.00  D    DAS-IST-EIN-ELEMENT-MIT-EINER-LA  URALT.VERSIO  0000   1994-08-17
 230.00       NGEN-VERSIONSANGABE               N
 240.00  FREE DAS-IST-EIN-ELEMENT-MIT-EINEM-FR  @             0000   1994-08-17
 250.00  TYPX EIEN-TYPNAMEN
 251.00 ........................................................................
 252.00 ........................................................................
 253.00 ........................................................................
 254.00 ........................................................................
 255.00 ........................................................................
show library=edt.lib.bsp,d 1000(10) ; #1000 .....................0100.00:001(0)
```

Das Inhaltsverzeichnis für den Elementtyp D der Bibliothek EDT.LIB.BSP soll in der aktuellen Arbeitsdatei ab Zeilennummer 1000 mit der Schrittweite 10 ausgegeben werden.

```
1000.00 TYP   E L E M E N T                          VERSION       VAR    DATE
1010.00  D    E.TEXT1                                 @             0006   1988-12-05
1020.00  D    E.TEXT2                                 @             0013   1988-12-05
1030.00  D    E.TEXT3                                 100           0011   1988-12-13
1040.00  D    E.TEXT3                                 101           0121   1988-12-20
1050.00  D    E.TEXT3                                 102           0007   1989-01-11
1060.00  D    DAS-IST-EIN-ELEMENT-MIT-EINEM-SE        @             0000   1994-08-17
1070.00       HR-LANGEM-NAMEN
1080.00  D    DAS-IST-EIN-ELEMENT-MIT-EINER-LA        URALT.VERSIO  0000   1994-08-17
1090.00       NGEN-VERSIONSANGEABE                    N
1191.00 ......................................................................
```

SHOW LIBRARY = EDT.LIB, R

In der Arbeitsdatei 9 werden alle Elemente der Bibliothek EDT.LIB aufgelistet, die zum Elementtyp R gehören.

@SHOW (Format 2) Ausgeben der Coded Character Set Namen

Mit @SHOW Format 2 wird eines Liste der im System möglichen Coded Character Set Namen ausgegeben. Zusätzlich wird ein partieller Code durch die Angabe von P gekennzeichnet. E bedeutet EBCDI-Code, I bedeutet ISO-Code.

In Systemen, in denen das Subsystem XHCS nicht installiert ist, wird diese Anweisung mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@**SHOW**	**C**CS [[**T**O] ln [(inc)]]	

ln Zeilennummer, ab der das Ergebnis in die aktuelle Arbeitsdatei geschrieben werden soll. ln kann auch durch Zeilennummervariablen (#L0-#L20) oder durch symbolische Zeilennummern (z.B. %,$) angegeben werden.

Ist ln nicht angegeben, wird das Ergebnis
- im L-Modus am Bildschirm ausgegeben,
- im Stapelbetrieb auf SYSOUT ausgegeben,
- im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht.

Falls ln nicht angegeben wurde und @PAR INFORMATION=ON eingeschaltet ist, wird im F-Modus eine Überschriftzeile zur Beschreibung der ausgegebenen Information angezeigt.

inc Inkrement zur Berechnung der Zeilennummern. Ist inc nicht angegeben, wird die mit ln implizit gegebene Schrittweite verwendet.

Spalte	**Bedeutung**
1-8	CCSN
10	P, wenn partieller Code
12	E (EBCDIC) oder I (ISO)
14	*, wenn der Coded Character Set an der Datensichtstation dargestellt werden kann.

@SORT Sortieren von Zeilen in Zeilenbereichen

Mit @SORT kann man zusammenhängende Zeilenbereiche in der aktuellen Arbeitsdatei byteweise aufsteigend oder absteigend sortieren.

@SORT verwendet eine Kombination von Quicksort und Bubblesort. Die Datenbestände werden durch Vertauschen der Satzverkettungen sortiert.

Operation	Operanden	F-Modus / L-Modus
@SORT	[rng*] { [:domain] / :**R** (clrng) } { [**A**] / **D** }	

rng* Zeilenbereich, dessen Daten sortiert werden sollen. Bestehend aus:
- einer einzelnen Zeile (z.B. 6)
- mehreren aufeinanderfolgenden Zeilen (z.B. 8-20)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.
Wird kein Zeilenbereich angegeben, werden alle Datensätze sortiert, bzw. der durch @RANGE angegebene Zeilenbereich.

domain Spaltenbereich, dessen Zeichen zur Sortierung berücksichtigt werden sollen. Bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, werden die Zeichen ab dieser Spalte bis zum Ende der Zeile zur Sortierung berücksichtigt.
Ist die erste Spaltenangabe größer als die Zeilenlänge, wird diese Zeile nicht behandelt und als „kleiner“ eingestuft.
Es wird byteweise von links nach rechts sortiert. Wird dabei das Ende einer Zeile erreicht, so ist sie „kleiner“.
Die zweite Spaltenangabe
- darf nicht kleiner als die erste sein,
- kann größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile zur Sortierung berücksichtigt.

R Die Spaltenangabe in clrng wird vom Satzende in Richtung Satzanfang interpretiert.

clrng — Spaltenbereich, wird vom Satzende an gezählt. Es gelten die gleichen Konventionen wie für domain.

A — Standardwert. Es wird aufsteigend sortiert (ascending).

D — Es wird absteigend sortiert (descending).

Beispiel

@SORT &:1-15

sortiert alle Sätze der aktuellen Arbeitsdatei aufsteigend nach dem Inhalt der Spalten 1 bis 15.

@SORT %-.%+19L :R(1-8) D

sortiert die ersten 20 Sätze der aktuellen Arbeitsdatei absteigend nach dem Inhalt der letzten 8 Spalten.

@SORT 20-.$:#I1-#I2

sortiert die Sätze von Zeilennummer 20 bis zum Ende der aktuellen Arbeitsdatei aufsteigend. Der die Sortierung bestimmende Spaltenbereich wird durch die Ganzzahlvariablen #I1 und #I2 angegeben.

@STAJV Information über Jobvariable ausgeben

Mit @STAJV kann der Benutzer abfragen, welche Jobvariablen unter einer bestimmten Benutzerkennung vorhanden sind und welche Eigenschaften diese Jobvariablen haben.

Die Informationen können:

– am Bildschirm ausgegeben werden,
– in eine Arbeitsdatei geschrieben werden.

Ist das Subsystem „Jobvariablen-Support“ nicht installiert, wird diese Anweisung mit einer Fehlermeldung abgewiesen.

Operation	Operanden	F-Modus / L-Modus
@STAJV	[string] [**T**O ln [(inc)]] [**S**HORT] | **LONG**	

string Auswahl derJobvariablen, die ausgegeben werden sollen.

string kann angegeben werden:
– explizit als Zeichenfolge in Hochkomma,
– implizit über eine Zeilennummer, eine Zeilennummervariable (#L0-#L20) oder eine Zeichenfolgevariable (#S0-#S20) (jeweils mit Spaltenbereich möglich).

Es sind alle Angaben erlaubt, die auch im BS2000-Kommando SHOW-JV-ATTRIBUTES gegeben werden dürfen, solange die Länge von 54 Zeichen nicht überschritten wird.
Es kann auch “*...*“ angegeben werden. Der EDT nimmt dann die Auswahl aus den Jobvariablen der eigenen Kennung nach der wildcard-Syntax (analog zum BS2000-Kommando SHOW-FILE-ATTRIBUTES) selbst vor.
Der Operand wird vom EDT nicht geprüft, sondern unverändert an das System weitergegeben.

Wird string nicht angegeben, werden alle Jobvariablen der eigenen Kennung ausgegeben.

Wird keine Jobvariable mit entsprechendem Namen gefunden, meldet der EDT einen Fehler und setzt den EDT-Schalter für DVS-Fehler. DVS-Fehler können in EDT-Prozeduren mit @IF, Format 1 abgefragt werden.

Falls string nicht vollqualifiziert oder nicht als Kettungsname angegeben wurde, braucht der EDT zur Durchführung zusätzlich einen Puffer von 8 PAM-Seiten, den er mit dem Makro REQM anfordert. Falls diese Seiten vom System nicht bereitgestellt werden können, wird @STAJV mit einem Fehler abgewiesen.

Ist string vollqualifiziert, wird die CATID nur mitausgegeben, wenn sie in string schon vorhanden war.

ln Zeilennummer, ab der die Informationen der Jobvariablen in die aktuelle Arbeitsdatei geschrieben werden. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. ln kann auch durch Zeilennummervariablen (#L0-#L20) oder durch symbolische Zeilennummern (z.B. %,$) angegeben werden.

Ist ln nicht angegeben, wird das Ergebnis
- im L-Modus am Bildschirm ausgegeben,
- im Stapelbetrieb auf SYSOUT ausgegeben,
- im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht.

inc Schrittweite, aus der die auf ln folgende Zeilennummern gebildet werden. Wird ln nicht angegeben, wird die implizit gegebene Schrittweite verwendet.

SHORT Es werden nur die Jobvariablennamen (einschließlich CATID und USERID) ausgegeben (Standardwert).

LONG Zusätzlich zu den Jobvariablennamen (inklusive CATID und USERID) werden weitere Kataloginformationen ausgegeben.

Falls ln nicht angegeben wurde und @PAR INFORMATION=ON eingeschaltet ist, wird im F-Modus eine Überschriftenzeile zur Beschreibung der Kataloginformationen ausgegeben.

Überschrift	**Bedeutung**
SIZE	Länge des aktuellen Wertes in Zeichen
M	ob auftragsüberwachende MONJV ('*')
JOBVARIABLE NAME	Jobvariablen-Name
CR-DATE	Erstellungsdatum (YY-MM-DD)
S	SHARE-Attribut (Y/N)
A	ACCESS-Atribut (W/R)
R	READ-PASS-Attribut (Y/N)
W	WRITE-PASS-Attribut (Y/N)

Bei der Angabe von LONG im F-Modus beträgt die Ausgabelänge für jeden Jobvariablen-Namen 80 Zeichen.

i

- Wenn eine Zeile angelegt wurde, deren Nummer größer als die bisherige höchste Zeilennummer ist, wird die aktuelle Zeilennummer verändert,
- Die Statusabfrage auf Systemjobvariablen ($SYSJV.) wird abgewiesen. Es gibt es jedoch folgende Möglichkeit: @SYSTEM 'SHOW-JV-ATTR JV-NAME($SYSJV.)' TO 1

Beispiel

```
   1.00 90-06-27178.....................................................
   2.00 GUTEN MORGEN, HEUTE IST DER 27.06.1990...........................
   3.00 .....................................................................









setjv 'heute'=2.............................................0001.00:001(0)
```

Der Jobvariablen HEUTE wird die Zeichenfolge, die in Zeile 2 steht zugeordnet.

```
   1.00 90-06-27178.....................................................
   2.00 GUTEN MORGEN, HEUTE IST DER 27.06.1990...........................
   3.00 .....................................................................









par global,information=on,index=off;stajv 'heute' long.....0001.00:001(0)
```

Einschalten der Informationszeile und Ausschalten der Zeilennummernanzeige. Mit @STAJV werden Informationen über die Jobvariable HEUTE in die Arbeitsdatei 9 geschrieben.

```
SIZE   M    JOBVARIABLE NAME                                  CR-DATE  S A R W
0000038 $EW.HEUTE                                             90-06-27 Y W N N
   .....................................................................
   .....................................................................









.....................................................0001.00:001(9)
```

Informationen über die Jobvariable werden angezeigt.

@STATUS Aktuelle Voreinstellungen und Variableninhalte anzeigen

Mit @STATUS können die vereinbarten Modi des EDT sowie die Inhalte von verschiedenen EDT-Konstanten und -Variablen ausgegeben werden.

Operation	Operanden	F-Modus / L-Modus
@**ST**ATUS	[=**A**LL] | [= **T**IME | **B**UFFER | **S**IZE | **SY**MBOLS | **D**ELIM | **V**DT | **M**ODES | **F**ILE | **P**AR[(procnr)] | **L**INEV | **I**NTV | ln-var | int-var | **SDF** | **C**CS | **LOG**] [,...] [**T**O ln [(inc)]]	

ALL Es werden alle Informationen der Parameter TIME, BUFFER, SIZE, SYMBOLS, DELIM, VDT, MODES und FILE ausgegeben. Zusätzlich werden die Benutzerkennung (USERID) und die Prozeßfolgenummer (TSN) ausgegeben.

Wird ALL angegeben, sind alle weiteren Operanden wirkungslos.

TIME Ausgegeben werden
- die momentane Uhrzeit,
- die bisherige Dauer des EDT-Laufs,
- die bisher benötigte CPU-Zeit,
- der CPU-Zeitunterschied zwischen den beiden letzten @STATUS.

BUFFER Ausgegeben werden
- die aktuelle Puffergröße,
- die Spaltennummer, ab der die Eingaben auf zulässige Länge geprüft werden sollen (@CHECK),
- bei Datenschreibstationen die rechte Schreibgrenze.

Im Stapelbetrieb wird nur die Spaltennummer ausgegeben.

SIZE Gibt die Gesamtzahl der virtuellen Seiten aus, die für die Arbeitsdateien gerade benötigt werden.

SYMBOLS Ausgegeben werden
- das aktuelle Anweisungssymbol, das mit @: verändert werden kann (siehe @:),
- die gültigen Textbegrenzer, die man mit @QUOTE verändern kann (siehe @QUOTE),
- die gültigen Jokerzeichen, die man mit @SYMBOLS verändern kann (siehe @SYMBOLS),
- das aktuelle Bereichssymbol mit dem vereinbarten Bereich (siehe @RANGE).
- das aktuelle Füllzeichen in hexadezimaler Form (siehe @SYMBOLS).
- das aktuelle Symbol für das Satztrennzeichen (siehe @PAR SEPARATOR).

DELIM Gibt die vereinbarte Menge der Textbegrenzerzeichen aus (siehe @DELIMIT).

VDT Gibt die Anzahl der Bildschirmzeilen und -spalten im L-Modus aus (siehe @VDT).
Bei der Datensichtstation 9763 wird zusätzlich das aktuelle Bildschirmformat ausgegeben (siehe @VDT).

MODES Gibt die Voreinstellungen aus, die mit @BLOCK, @CHECK, @LOWER, @INPUT, @TABS, @EDIT und @VTCSET vereinbart wurden.
Zusätzlich wird die Einstellung der Syntaxkontrolle im L-Modus, der Ausführungsmodus (siehe @SYNTAX) und die mit @AUTOSAVE einstellbaren Werte ausgegeben.

FILE Gibt den globalen Dateinamen aus, der durch das letzte @FILE vereinbart wurde. Wurde dort auch eine Versionsnummer angegeben, wird diese mit ausgegeben.

Falls eine POSIX-Datei mit @XOPEN eröffnet wurde, wird der Name der Datei ausgegeben.

Falls ein lokaler @FILE-Eintrag implizit oder explizit definiert wurde, bzw. ein Bibliothekselement oder eine Datei mit @OPEN (Format 2) eröffnet wurde, wird der Name der Datei bzw. Bibliotheks- und Elementname ausgegeben.

FILE wird ignoriert, wenn weder ein globaler noch ein lokaler @FILE Eintrag vereinbart wurde.

PAR(procnr) Ausgegeben werden
- der lokale @FILE-Eintrag, bzw. das eröffnete Bibliotheks-Element mit Bibliotheksname und Typ bzw. der Name der eröffneten POSIX-Datei analog @STATUS = FILE (siehe @FILE LOCAL).
- der Name einer eventuell voreingestellten Standard-Bibliothek bzw. Standard -Typ (siehe @PAR LIBRARY bzw. @PAR ELEMENT-TYPE).
- die mit @PAR LIMIT und INC eingestellten Werte.

für die Arbeitsdateien 0 bis 9 zusätzlich
- die Art der Darstellung im F-Modus Bildschirm: @PAR LOWER, @PAR HEX, @PAR EDIT LONG, @PAR CODE (siehe @PAR).
- das aktuelle Struktursymbol (siehe @PAR STRUCTURE) und die mit @PAR LIMIT und @PAR INCREMENT eingestellten Werte.
- Information über Bildschirm-Dienste (siehe @PAR SCALE, @PAR INFORMATION, @PAR PROTECTION).
- Fensterspezifische Voreinstellungen für Position und Darstellung (siehe @PAR INDEX, @PAR EDIT FULL, @SETF).
  Die Position ist 0, falls nicht definiert.

procnr Nummer der Arbeitsdatei (0-22).

LINEV Zeigt die Inhalte aller Zeilennummervariablen #L0 bis #L20.

INTV Zeigt die Inhalte aller Ganzzahlvariablen #I0 bis #I20.

ln-var Der Inhalt der genannten Zeilennummervariablen wird ausgegeben.

int-var Der Inhalt der genannten Ganzzahlvariablen wird ausgegeben.

ln Zeilennummer, ab der die Informationen in die aktuelle Arbeitsdatei geschrieben werden.
Wenn eine Zeile angelegt wurde, deren Nummer größer ist als die bisherige höchste Zeilennummer, wird die aktuelle Zeilennummer verändert.

Ist ln nicht angegeben, wird das Ergebnis
- im L-Modus am Bildschirm ausgegeben,
- im Stapelbetrieb auf SYSOUT ausgegeben,
- im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht.

inc	Schrittweite, aus der die auf ln folgenden Zeilennummern gebildet werden sollen. Wird inc nicht angebenen, wird die implizit gegebene Schrittweite verwendet.
SDF	Es wird der Name der aktuellen SDF-Syntaxdateien (System-/Gruppen-/Benutzer-Syntaxdatei) und eines mit @PAR SDF-PROGRAM oder @SDFTEST eingestellten internen Programmnamens ausgegeben. Ist kein interner Programmname definiert, wird *NONE ausgegeben.
CCS	Ausgegeben wird – den Namen des voreingestellten CCS. – der Name des aktuell eingestellten Coded Character Sets (CCSN). Wenn das Subsystem XHCS im System nicht vorhanden ist, wird CCS ignoriert.
LOG	Es werden die eingestellten Werte für die Protokollierung ausgegeben (siehe @LOG)

Wird kein Operand angegeben, wird ALL angenommen.

Wird ein Operand, außer ALL, angegeben, muß auch das Gleichheitszeichen angegeben werden. Zwischen dem Gleichheitszeichen und dem ersten Operanden darf kein Komma stehen.

Wurde zuvor mit @SYNTAX TEST=ON der Testmodus für die L-Modus-Eingabe eingeschaltet, und wird @STATUS im L-Modus eingegeben, so wird eine etwaige Angabe von TO ln(inc) ignoriert, d.h. die Ausgabe erfolgt stattdessen nach SYSOUT.

Es ist möglich, ln-var und int-var gleichzeitig, bzw. öfter in einer @STATUS-Anweisung anzugeben.

@SUFFIX Anhängen von Zeichenfolgen an Zeilen

Mit @SUFFIX wird eine Zeichenfolge an das Ende einer oder mehrerer bestehender Zeilen angehängt (siehe auch @PREFIX, Voranstellen von Zeichenfolgen).

Operation	Operanden	F-Modus / L-Modus
@**SU**FFIX	range **W**ITH string	

range — Zeilenbereich, bestehend aus:
- einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
- einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
- einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden. Auch die Angabe von Zeichenfolgevariablen (#S0 bis #S20) ist zulässig.

string — Zeichenfolge, die angehängt werden soll.

string kann angegeben werden:
- explizite Angabe in Hochkomma,
- implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Beispiel

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 UND.....................................................................
   5.00 UND.....................................................................
   6.00 ........................................................................
suffix 4-5 with ' noch ' .....................................0001.00:001(0)
```

Den Zeilen 4 und 5 wird die Zeichenfolge NOCH angehängt.

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 UND NOCH................................................................
   5.00 UND NOCH................................................................
   6.00 .....................................................................




suffix 4-5 with 3 ..............................................0001.00:001(0)
```

Den Zeilen 4 und 5 wird der Inhalt der Zeile 3 angehängt.

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 UND NOCH EINMAL..........................................................
   5.00 UND NOCH EINMAL..........................................................
   6.00 .....................................................................




suffix 4-5 with ' '*5 ; suffix 4-5 with 4 .......................0001.00:001(0)
```

Zunächst werden den Zeilen 4 und 5 je 5 Leerzeichen und anschließend den Zeilen 4 und 5 der Inhalt der Zeile 4 angehängt.

```
   1.00 UND.....................................................................
   2.00 NOCH....................................................................
   3.00 EINMAL..................................................................
   4.00 UND NOCH EINMAL     UND NOCH EINMAL.....................................
   5.00 UND NOCH EINMAL     UND NOCH EINMAL.....................................
   6.00 .....................................................................
```

@SYMBOLS Symbole definieren

Mit @SYMBOLS können:

– die Jokersymbole Asterisk (='*') und Slash ('/') zur Angabe des Suchbegriffs für die @ON-Anweisung auf andere Zeichen umdefiniert werden (z.B. um nach den Zeichen * und / zu suchen).

– das aktuelle Füllzeichen für einen Bereich im Datenfenster zwischen Satzende und Bildschirmzeilenende umdefiniert werden.

Operation	Operanden	F-Modus / L-Modus
@**SYM**BOLS	[,] [**A**STERISK [= '*' | ='spec1'] [,] [**S**LASH [= '/' | ='spec2'] [,] [**FILLER**] [= X'00' | =X'hex' | ='char']	

ASTERISK Definiert das Jokerzeichen für eine beliebig lange, auch leere Zeichenfolge.

='*' Standardwert.

='spec1' Sonderzeichen, das das Jokerzeichen für eine beliebig lange, auch leere Zeichenfolge bestimmt.

SLASH Definiert das Jokerzeichen

='/' Standardwert.

='spec2' Sonderzeichen, das das Jokerzeichen bestimmt.

FILLER Definiert das Füllzeichen, das im F-Modus zwischen Satzende und Bildschirmzeilenende eingesetzt wird.

=X'00' Standardwert.

=X'hex' Ein beliebiges Zeichen in hexadezimaler Darstellung. Nichtabdruckbare Zeichen werden als Schmierzeichen dargestellt.

='char' Beliebiges Zeichen, das das Füllzeichen bestimmt.

spec1 und spec2 müssen verschieden voneinander und den in @QUOTE definierten Zeichen sein.

Ist spec1 oder spec2 kein Sonderzeichen, wird @SYMBOLS mit einer Fehlermeldung abgewiesen: % EDT3952 INVALID SYMBOL

Füllzeichen, die am Ende einer Bildschirmdatenzeile stehen, werden nicht in die Datei aufgenommen. Füllzeichen innerhalb eines Datensatzes werden bei Neuaufnahme, bzw. Änderung einer Bildschirmzeile in Leerzeichen umgesetzt.

Im F-Modus-Bildschirm wird das Füllzeichen zwischen dem Satzende und dem Bildschirm-Zeilenende standardmäßig mit X'00' (NIL-Zeichen) angenommen. Dadurch wird das Satzende erkennbar und es wird das ungewollte Abschneiden von Leerzeichen am Satzende verhindert.

Zum Löschen eines ganzen Datensatzes können LZE und LZF nur bedingt verwendet werden.

– LZE löscht alle Zeichen des Datensatzes ab der eingegebenen Position.

– LZF löscht nur den Zeilenrest, etwaige Zeichen im Datensatz dahinter werden nachgezogen.

Ein ganzer Datensatz bei Spaltenposition ungleich 1 muß explizit mit @DELETE oder mit der Kurzanweisung D gelöscht werden.

Durch @SYMBOLS FILLER = ' ' wird die bis zu EDT V16.2 entsprechende Darstellungsform eingestellt.

Bildschirmzeilen, die nur aus Füllzeichen ungleich ' ' bestehen, werden nicht in die Datei aufgenommen.
Bildschirmzeilen, die nur aus Füllzeichen = ' ' bestehen, werden als Datensätze bestehend aus zwei Leerzeichen angelegt.

@SYNTAX Einstellen der Syntaxkontrolle und des Ausführungsmodus

Mit @SYNTAX von kann für die Linemodus-Eingabe (Lesen mit @EDIT, @EDIT ONLY oder von SYSDTA)

– die Art der Syntaxkontrolle eingestellt werden
– ein Testmodus ein- und ausgeschaltet werden

Operation	Operanden	F-Modus / L-Modus
@**SYN**TAX	[**SEC**URITY { [=**HIGH**] / =**LOW** }] [[,] **TEST**MODE { [**=ON**] / **=OF**F }]	

SECURITY

= HIGH Im L-Modus gilt die selbe Syntaxkontrolle wie im F-Modus.
Wesentliche Unterschiede zu LOW:
– Leerzeichen in Schlüsselwörtern werden nicht mehr überlesen
– Kommentare am Ende einer Anweisung sind nur erlaubt, wenn in der Syntaxbeschreibung angegeben (z.B. @NOTE,@PROC, @END)

= LOW Kompatible Syntaxkontrolle des L-Modus.
Sie toleriert bei Anweisungen, die aus dem Anweisungsvorrat des EDT V15 stammen, eingestreute Kommentare. Diese Schreibweisen garantieren keine Eindeutigkeit, Fehlersituationen werden nicht immer erkannt.

Wird der Operand SECURITY nicht angegeben, bleibt die momentane Einstellung erhalten.

Mit der Anweisung @STATUS kann die aktuelle Syntaxeinstellung ausgegeben werden.

TESTMODE

= ON Die Anweisungen werden syntaktisch geprüft, aber nicht ausgeführt.
Im Line-Modus-Dialog eingegebene Datenzeilen werden nicht in die Arbeitsdatei übernommen.
Eine Filter-Routine oder Benutzeranweisungen werden nicht ausgeführt

Ausnahmen:
Die Anweisungen @LOG, @SYNTAX und das Umdefinieren des aktuellen EDT-Fluchtsymbols (@:) werden auch im Testmodus ausgeführt.
Die Anweisung @STATUS wird ausgeführt, gibt aber nur auf SYSOUT aus.
@HALT wird ausgeführt. Mit @RETURN kann zum aufrufenden Programm zurückgekehrt werden. Dabei wird implizit der Test-Modus zurückgesetzt.
Bei Anweisungen mit variablen Operanden (z.B. @OPEN &#S01) wird der Inhalt der Stringvariable nicht überprüft.

Der Operand <text> bei der Anweisung @SET Format 6, @IF Format 1 und @+, @– wird nicht syntaktisch überprüft.
Externe Anweisungen (Anweisungen mit Benutzer-Fluchtsymbol) werden nicht geprüft.
In diesen Fällen wird im Dialog folgende Meldung ausgegeben:
% EDT0110 TESTMODE: SYNTAX CANNOT BE TESTED
Eingabe in einer EDT-Prozedur, oder L-Modus-Dialog:
Beginnt die Zeichenfolge mit mehr als einem Anweisungssymbol (z.B. @@...), so wird sie syntaktisch geprüft.
Bei Fehlerfreiheit wird im Dialog eine Quittung ausgegeben:
% EDT0100 TESTMODE: NO SYNTAX ERROR

= OFF Der Test-Modus wird ausgeschaltet.
Beendet @HALT oder @RETURN den DIALOG-Modus, wird der Test-Modus ebenfalls ausgeschaltet.

Wird der Operand TESTMODE nicht angegeben, bleibt der Ausführungsmodus erhalten.

Beispiel: @SYNTAX SEC,TEST

Diese Anweisung gibt dem Anwender die Möglichkeit, alte EDT-Prozeduren zu prüfen, ob sie der im Handbuch beschriebenen Syntax entsprechen.

Anweisungen, die nicht dieser Syntax entsprechen, sollten korrigiert werden.

Datensätze und Anweisungen, die über die LU15-Schnittstelle eingegeben werden, sind von der Einstellung des Testmodes nicht betroffen.

Diese Anweisung wird nicht unterstützt, wenn der EDT allein über die alte Line-Modus-Unterprogrammschnittstelle aufgerufen wird.

Protokollierung im Test-Modus

Im L-Modus wird zusätzlich zur fehlerhaften Anweisung die Stelle mit : markiert, an der der Fehler erkannt wurde.
Ist im L-Modus SECURITY=LOW eingestellt und werden Zeichen bei der syntaktischen Prüfung überlesen, so wird zur Warnung die Meldung % EDT0120 ausgegeben und die überlesenen Zeichen mit : markiert.

Voreinstellung

SECURITY=LOW
– im Stapelbetrieb
– im L-Modus-Dialog bei gesetztem Systemschalter 5
– und an der LU15-Schnittstelle.

SECURITY=HIGH in allen anderen Fällen. Bei Beginn des EDT-Laufs ist der Testmodus ausgeschaltet.

@SYSTEM Systemkommandos absetzen

Mit @SYSTEM kann man

– den EDT-Lauf unterbrechen und in das Betriebssystem verzweigen (wie mit K2).

– ein Betriebssystemkommando zur Ausführung bringen, ohne daß der EDT-Lauf unterbrochen wird.

Operation	Operanden	F-Modus / L-Modus
@**SY**STEM	[string [**T**O ln [(inc)]]]	

string Zeichenfolge, die ein Kommando angibt.
string kann angegeben werden:
– explizite Angabe in Hochkomma,
– implizite Angabe über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils mit Spaltenbereich möglich).

Das Kommando wird sofort ausgeführt und anschließend wird der EDT-Lauf fortgesetzt.
Bei den System-Kommandos EXIT-JOB, LOGOFF, HELP-SDF, CALL-PROCEDURE, START-PROGRAM und LOAD-PROGRAM und mittels SDF-A definierten und durch Kommandoprozeduren implementierten Anwender-Kommandos wird der EDT-Lauf abgebrochen und der EDT entladen. Das Kommando kann mit und ohne Schrägstrich am Anfang geschrieben werden. Kleinbuchstaben werden in Großbuchstaben umgewandelt.

Es dürfen nur Kommandos angegeben werden, die mit dem CMD-Makro abgesetzt werden dürfen. Ist ein Kommando nicht erlaubt, wird es mit einer Fehlermeldung abgewiesen. CMD-Makro und die zugelassenen Kommandos siehe Handbuch „Makroaufrufe an den Ablaufteil“ [5].

Wird string nicht angegeben, wird in das Betriebssystem verzweigt. Durch das Kommando RESUME-PROGRAM wird der EDT-Lauf an der Stelle fortgesetzt, an der er durch @SYSTEM unterbrochen wurde.

ln Bei Angabe von ln, wird eine eventuelle Ausgabe des Systemkommandos, die nicht mit mode=phys oder mode=form erfolgt, in die aktuelle Datei übernommen, beginnend ab Zeilennummer ln.
In diesem Fall braucht der EDT zusätzlich einen Puffer, den er mit dem Makro REQM anfordert.
ln kann auch durch Zeilennummervariablen (#L0-#L20) oder symbolisch (z.B. %,$) angegeben werden.

inc Schrittweite, aus der die auf ln folgenden Zeilennummern gebildet werden sollen. Wird inc nicht angegeben, wird die implizit gegebene Schrittweite verwendet.

Die aktuelle Zeilennummer wird verändert, wenn eine Zeile angelegt wurde, deren Nummer größer ist als die bisherige höchste Zeilennummmer.

Ist eine Datei durch @OPEN real eröffnet, dann sollte man den EDT-Lauf keinesfalls mit @SYSTEM beenden, da sonst die Datei nicht geschlossen wird.

Statt @SYSTEM 'START-PROGRAM...' oder @SYSTEM 'LOAD-PROGRAM...' sollte man @EXEC bzw. @LOAD verwenden, da bei diesen vor Ausführung der Anweisungen offene Dateien geschlossen werden und eventuell vorhandene Sicherungsdateien (siehe @AUTOSAVE) gelöscht werden.

Ab BS2000/OSD-BC V1.0 wird bei einer Fehlermeldung auch der Meldungsschlüssel aus dem Kommando-Returncode des aufgerufenen Kommandos ausgegeben.

@TABS Tabulator setzen

Mit @TABS kann man

- ein Tabulatorzeichen mit bis zu 8 Positionen (Spalten) definieren,
- das aktuelle Tabulatorzeichen mit den zugehörigen Tabulatorspalten am Bildschirm ausgeben lassen,
- bis zu 8 Positionen für den Hardware-Tabulator definieren,
- die Funktion des Hardware-Tabulators ein- und ausschalten.

Im Stapelbetrieb und in EDT-Prozeduren (@DO, @INPUT) ist das Tabulatorzeichen ungültig.

Operation	Operanden	F-Modus / L-Modus
@**T**ABS	:: [tab [[:]cl1 [,cl2,...]] [**C**HECK | **F**ORWARD] [cl]]] [cl1[,cl2,...]] [**ON**] | **O**FF [::] **V**ALUES	

tab — Zeichen für den Sofware-Tabulator, das der EDT künftig als Tabulatorzeichen interpretieren soll.
Im F-Modus kann das Zeichen ; nicht verwendet werden, da es als Trennzeichen für Anweisungen interpretiert wird.
Das Zeichen : hinter tab ist dann notwendig, wenn tab eines der Zeichen C, F oder V oder eine Ziffer ist.

cl1,cl2,... — Gibt, durch Komma voneinander getrennt, bis zu 8 Spalten an, auf die mit dem Tabulatorzeichen positioniert werden kann. Der EDT positioniert auf die Spalten in genau der Reihenfolge, in der sie angegeben werden. Deshalb müssen die angegebenen Spaltennummern aufsteigend geordnet sein. Alle Werte zwischen 1 und 255 können angegeben werden.

Wird der Stellungsoperand :: angegeben, so beziehen sich die Werte auf den Software-Tabulator.
Wird :: nicht angegeben, so überprüft der EDT, ob die Werte in aufsteigender Reihenfolge angegeben sind und weist @TABS mit einer Fehlermeldung zurück, falls die Werte nicht aufsteigend sind.

CHECK Wenn der EDT auf eine Spalte positioniert, berücksichtigt er normalerweise nicht, ob diese Spalte durch den Text, der vor dem Tabulatorzeichen steht, bereits beschrieben wurde. Die vorher angelegten Textzeilen werden überschrieben. CHECK bewirkt daß der EDT eine Warnung am Bildschirm ausgibt.

FORWARD Verhindert eine Rückwärtspositionierung. Der EDT hält sich zwar auch hier an die angegebene Reihenfolge der Tabulatorpositionen. Falls die Tabulatorposition, die gerade an der Reihe ist, aber vor oder auf der zuletzt beschriebenen Spalte liegt, übergeht der EDT diese Tabulatorposition und nimmt die nächste.

Bei Aufruf des EDT ist CHECK und FORWARD ausgeschaltet.

cl (Nur im L-Modus)
Verändert den Wert für die Prüfung der Zeilenlänge ($1 \leq cl \leq 256$). Der EDT prüft jede Texteingabe, ob sie länger ist als cl Zeichen. Der Standardwert für cl ist 256. Besteht eine Texteingabe aus mehr als cl Zeichen, wird die Zeile (mit einer maximalen Länge von 256 Zeichen) zwar angelegt, der EDT gibt aber eine Warnung am Bildschirm aus.

Der Wert für die Überprüfung der Zeilenlänge kann auch mit @CHECK (L-Modus) verändert werden.

Wird in den F-Modus umgeschaltet, wird der Wert von cl wieder auf 256 gesetzt. Im F-Modus gibt es eine entsprechende Funktion (siehe @PAR LIMIT).

ON Falls in der gleichen oder einer vorherigen Anweisung schon Positionen für den Hardware-Tabulator definiert sind, wird die Funktion eingeschaltet, d.h. durch ⇥ positioniert sich die Schreibmarke auf die nächste definierte Spalte. Der Standardwert ist ON.

OFF Die Funktion des Hardware-Tabulators wird ausgeschaltet. Die definierten Positionen bleiben erhalten und können durch @TABS ON wieder aktiviert werden.

VALUES Es werden das gültige Software-Tabulatorzeichen und die zugehörigen Tabulatorpositionen ausgegeben.
Ist der Hardware-Tabulator definiert, werden nur die Tabulatorpositionen ausgegeben. Ist kein Tabulatorzeichen definiert, wird die Anweisung ignoriert.

Im Stapelbetrieb wird auf SYSLST ausgegeben.

Wird @TABS:: ohne weitere Operanden angegeben, wird der Tabulator (egal ob Soft- oder Hardware-Tabulator) als nicht definiert gesetzt.

Ist @PAR EDIT LONG = ON eingestellt, so kann mit dem Hardware-Tabulator nur auf Positionen, die kleiner als die Bildschirmbreite sind, positioniert werden, größere Positionswerte werden ignoriert.

Auf der Datensichtstation 3270 wird der Hardware-Tabulator nicht unterstützt.

Ist der Hardware-Tabulator gesetzt und eingeschaltet, kann nur innerhalb der Tabulatorpositionen mit EFG und AFG ein- bzw. ausgefügt werden. Bei feldübergreifenden Zeichenfolgen ist es vorteilhaft, vorübergehend auf @TABS OFF umzuschalten.

Tabulatorzeichen werden in Anweisungen nur als solche interpretiert, wenn sie im Operanden text angegeben sind (siehe @SET, Format 6).

Enthält ein eingegebener Text mehr Tabulatorzeichen als die Anzahl der vereinbarten Tabulatorpositionen, behandelt der EDT die überzähligen Tabulatorzeichen als normale Textzeichen.

Die spaltengerechte Ausrichtung mit Hilfe der Tabulatorfunktion erfolgt nur bei der Neueingabe von Datensätzen. Wird z.B. ein Datensatz durch Kopieren eingefügt, muß mindestens ein Zeichen überschrieben, geändert bzw. eingefügt werden.

i

Hauptanwendung von @TABS ist das spaltengerechte Erstellen von Dateien, z.B. von Sourceprogramm-Dateien. Beispielsweise ist bei der Erstellung einer Datei für ein Assemblerprogramm @TABS::[:10,16,40 CHECK 71 sinnvoll. Mit CHECK wird erreicht, daß bei Angabe überlanger Namen eine Meldung ausgegeben wird (FORWARD anstelle von CHECK wäre hier nicht sinnvoll). Die Angabe einer maximalen Zeilenlänge von 71 ist deshalb sinnvoll, damit die spezielle Spalte 72, in der man Fortsetzungszeilen kennzeichnet, nicht versehentlich überschrieben wird.

Beispiel Software-Tabulator

```
  23.00 ..........................................................................
tabs ::[: 10, 16, 40 ..........................................0000.00:001(0)
```

Als Tabulatorzeichen wird [vereinbart. Die Tabulatorspalten seien 10, 16 und 40.

```
   1.00 [balr[14,15[unterprogramm [ dann zurueck ] ..............................
   2.00 sprung[dc[c' in ordnung' ................................................
   3.00 ..........................................................................
```

Text wird zusammen mit Tabulatorzeichen eingegeben. Man beachte, daß nur 3 Tabulatorpositionen vereinbart waren, aber 4 Tabulatorzeichen in der ersten eingegebenen Zeile vorhanden sind.

```
   1.00          BALR  14,15                   UNTERPROGRAMM [ DANN ZURUECK ]...
   2.00 SPRUNG   DC C  ' IN ORDNUNG'.............................................
   3.00 ..........................................................................
```

Man erkennt die Ausrichtung auf die Spalten 10, 16 und 40. Das vierte Tabulatorzeichen der ersten Zeile hat der EDT als Text interpretiert, da nur 3 Tabulatorspalten vereinbart wurden.

Beispiel Hardware-Tabulator

Das gleiche Ergebnis (bis auf die CHECK-Funktion) wird erreicht, wenn @TABS 10,16,40 als Anweisung eingegeben und mit ⇥ positioniert wird.

@TMODE Prozeßeigenschaften ausgeben

Mit @TMODE erhält der Benutzer Informationen über den Prozeß, unter dem der EDT abläuft. Die Informationen werden als Meldung ausgegeben.

Operation	Operanden	F-Modus / L-Modus
@TMODE		

Folgende Informationen über den Prozeß, unter dem der EDT läuft, werden ausgegeben:

TSN	Prozeßfolgenummer
USERID	Benutzerkennung des LOGON-Kommandos
ACCOUNT	Abrechnungsnummer des Prozesses
CPU-TIME	Verbrauchte CPU-Zeit
DATE	Datum (YY-MM-DD)
TIME	Zeit
ANWEISUNGSSYMBOL	aktuelles Anweisungssymbol (z.B. @)
TERMINAL	Typ des Bildschirms

Beispiel

```
  23.00 ..........................................................................
tmode.....................................................0000.00:001(0)
```

Informationen über die Prozeßeigenschaften werden angefordert.

```
  22.00 ..........................................................................
%  EDT0300 0582 BOND     007         15.1635 90-09-27 15:12:19 @ 9750
.....................................................................0000.00:001(0)
```

@UNLOAD Entladen eines Moduls

Mit @UNLOAD können Module, die mit @RUN oder @USE geladen wurden, wieder entladen werden.

Operation	Operanden	F-Modus / L-Modus
@**UNL**OAD	(name)	

name Namen des Bindemoduls oder der Ladeeinheit, die entladen wird.

Kann das Modul nicht entladen werden, wird @UNLOAD mit einer Fehlermeldung abgewiesen und der EDT-Fehlerschalter gesetzt.

Mögliche Ursachen:

– Falscher Modulname,
– Modul bereits entladen,
– Modul ist mehrfachbenutzbar geladen,
– name ist kein Modulname, sondern der Name eines Programmabschnittes (CSECT) oder einer Einsprungstelle (ENTRY).

@UNSAVE Datei löschen

@UNSAVE löscht eine Datei.

Operation	Operanden	F-Modus / L-Modus
@**U**NSAVE	'file' [(ver)]	

file Name der zu löschenden Datei. / nach Vergabe eines LINK-Namen durch das SET-FILE-LINK-Kommando darf nicht angegeben werden.

ver Versionsnummer der zu löschenden Datei.

Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer. Wird * angegeben, erscheint vor dem Löschen die aktuelle Versionsnummer auf dem Bildschirm. Wird eine falsche Versionsnummer angegeben, erscheint die richtige auf dem Bildschirm, und die Datei wird nicht gelöscht.

Im Gegensatz zu @ELIM wird bei @UNSAVE auch der Katalogeintrag gelöscht.

@UPDATE Datensätze ändern

@UPDATE ändert oder korrigiert Datensätze, fügt Datensätze hinzu oder gibt Datensätze für Korrekturen aufbereitet am Bildschirm aus. Es gibt drei Formate für diese Anweisung.

@UPDATE (Format 1) Datensätze ändern

Bestehende Datensätze werden ganz oder nur innerhalb eines Spaltenbereichs geändert oder gelöscht.
Neue Datensätze werden erzeugt.

Operation	Operanden	L-Modus
@**U**PDATE	ln [:domain] ; text	

ln — Zeilennummer der zu verändernden bzw. neu anzulegenden Zeile. Der Minimalwert ist 0.0001, der Maximalwert 9999.9999. ln kann auch durch Zeilennummervariablen (#L0 bis #L20) oder symbolisch (z.B. %,$) angegeben werden.

domain — Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25),

dessen Inhalt durch die Zeichenfolge ersetzt wird, die hinter ; angegeben wird. Existiert die Zeile noch nicht, oder ist die Zeile kürzer als durch den ersten Spaltenwert angegeben, wird die Zeile bis dahin mit Leerzeichen aufgefüllt.

Wird nur ein Spaltenwert angegeben, wird für den zweiten Spaltenwert 256 angenommen.
Wird kein Spaltenbereich angegeben, wird der durch @UPDATE, Format 3, eingestellte Standard-Spaltenbereich verwendet. Standard-Spaltenbereich zu Beginn des EDT-Laufs ist 1-256.

text — Die hinter dem Zeichen ; folgende Zeichenfolge ersetzt den zu korrigierenden Text im Spaltenbereich domain. text kann auch eine leere Zeichenfolge sein.

Tabulatorzeichen in text werden nicht ausgewertet.

text kann auch mit dem Anweisungssymbol (@ oder vereinbartes Zeichen) beginnen, ohne als Anweisung interpretiert zu werden.

Die Anweisung verändert die aktuelle Zeilennummer nicht, außer es wird durch Eingabe einer leeren Zeichenfolge die letzte Zeile der Datei gelöscht.

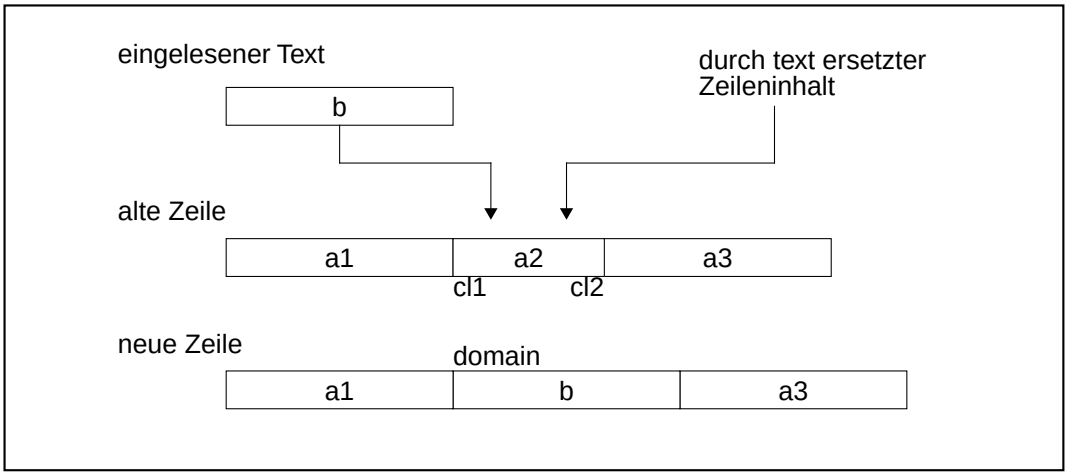


Bild 10: Korrektur einer Zeile mit @UPDATE, Format 1

Der über @UPDATE eingelesene Text b überschreibt genau den Text a2, der im angegebenen Spaltenbereich domain steht. Da der Korrekturtext b länger ist als der zu überschreibende Text, wird der nachfolgende Text a3 nach rechts verschoben. Wenn der Korrekturtext kürzer ist als der zu überschreibende Text, wird der nachfolgende Zeileninhalt nachgezogen und an den Korrekturtext angefügt.

@UPDATE (Format 2) Aufbereiten von Datensätzen

Ein Abschnitt aus einer Datei wird in aufbereiteter Form für die Eingabe mit @UPDATE, Format 1, am Bildschirm ausgegeben.

Dieses Format wird im Stapelbetrieb ignoriert und in EDT-Prozeduren abgewiesen.

Operation	Operanden	L-Modus
@**U**PDATE	[ln] [:domain]	

ln Einschließlich dieser Zeile wird in Richtung Dateiende der Inhalt der Datei aufbereitet ausgegeben.

Alle Zeilen werden bis zum Dateiende ausgegeben oder - wenn die Datei größer ist - soviele Zeilen, wie der Bildschirm faßt.

Die Anzahl der ausgegebenen Zeilen entspricht dem vom System eingestellten Standardwert für die benutzte Datensichtstation oder einem über @VDT angegebenen Wert.

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25),

dessen Inhalt ausgegeben wird.

Wird nur ein Spaltenwert angegeben, wird für den zweiten Spaltenwert 256 angenommen.
Wird kein Spaltenbereich angegeben, wird der durch @UPDATE, Format 3, eingestellte Standard-Spaltenbereich verwendet. Zeilen, die kleiner als der erste Spaltenwert sind, werden bei der Ausgabe übersprungen. Standard-Spaltenbereich zu Beginn des EDT-Laufs ist 1-256.

@UPDATE, Format 2 setzt das Arbeiten im Block-Modus voraus. Wurde der Block-Modus ausgeschaltet (BLOCK ON ist Standard), wird über diese Anweisung intern BLOCK ON eingeschaltet. Dies wird durch die Meldung BLOCK ON angezeigt.

@UPDATE, Format 2 muß immer die letzte Anweisung eines Anweisungsblocks sein, da die restlichen nicht bearbeiteten Anweisungen des Anweisungsblocks verloren gehen. Das Zeilenendekennzeichen (logisches Zeilenende) muß gesetzt werden, falls es überschrieben worden ist:

```
@Uxxxx.xxxx;text<
```

oder wenn ein anderer als der Standard-Spaltenbereich (siehe @UPDATE, Format 3) angegeben wurde:

```
@Uxxxx.xxxx:yyy-yyy;text<
```

@	Aktuelles Anweisungssymbol
xxxx.xxxx	EDT-Zeilennummer
yyy-yyy	Spaltenbereich
text	der entsprechend dem Eingabemodus aufbereitete Dateiinhalt
<	Logisches Zeilenende

Können auf einem Bildschirm nicht alle Datensätze von ln bis zum Dateiende dargestellt werden, erscheint in der letzten Bildschirmzeile die aktuelle Zeilennummer in der Form:

```
@Nxxxx.xxxx
```

Diese Ausgabe besagt, daß eine neue Zeile mit der Zeilennummer xxxx.xxxx erzeugt wird, wenn Daten eingegeben werden. Weitere Zeilen werden fortlaufend mit der aktuellen Schrittweite erstellt.

Der auf die Ausgabe folgende Eingabeblock wird ohne Einschränkungen bearbeitet. Er kann aus modifizierten @UPDATE, neuen Anweisungen oder neuen Daten bestehen. Ein Blättern mit * + – 0 ist nicht möglich, da dies keine Anweisungen sind, und daher innerhalb eines Eingabeblocks als Datensätze interpretiert werden.

Es kann auch K1 oder eine leere Eingabe zum Blättern in Richtung Dateiende verwendet werden.

Durch diese Anweisung wird der Wert des Zeilennummern-Symbols ? (Zeilennummer der 1. Trefferzeile nach @ON) verändert. Es zeigt dann immer auf die erste Zeile des nächsten auszugebenden Dateiabschnitts (Bei Erreichen des Dateiendes wird der Wert der höchsten Zeilennummer zugewiesen.)

Wird also für ln bei @UPDATE das Zeilennummern-Symbol ? angegeben, wird der nächste Dateiabschnitt aufbereitet ausgegeben.

Enthält der auszugebende Zeilenabschnitt Zeichen, die im Zeichen-Modus @INPUT CHAR nicht darstellbar sind, muß im Hexadezimal-Modus korrigiert werden (umschalten mit @INPUT HEX).

Im Zeichen-Modus werden solche Zeilen dargestellt als:

```
@Nxxxx.xxxxztext<
oder
@Nxxxx.xxxx:yyyztext<
```

z Gerätespezifisches Schmierzeichen.

Nicht abdruckbare Zeichen in text werden ebenfalls durch z (gerätespezifisches Schmierzeichen) dargestellt. Durch @N für @NOTE wird die Anweisung in einem Eingabeblock ignoriert.

@UPDATE (Format 3) Standard-Spaltenbereich definieren

Für die Formate 1 und 2 wird mit @UPDATE, Format 3 ein Standard-Spaltenbereich definiert.

Operation	Operanden	L-Modus
@**U**PDATE	**C**OLUMN [domain]	

domain Spaltenbereich bestehend aus:
- einer einzelnen Spalte (z.B. 10-10)
- einem zusammenhängenden Spaltenbereich (z.B. 15-25),

dessen Inhalt ausgegeben wird.

Wird nur ein Spaltenwert angegeben, wird für den zweiten Spaltenwert 256 angenommen.
Wird kein Spaltenbereich angegeben, wird der zu Beginn des EDT-Laufs eingestellte Standard-Spaltenbereich 1-256 angenommen.

@USE Definieren externer Anweisungsroutinen

Mit @USE kann man

– externe Anweisungsroutinen (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]) definieren und

– Fluchtsymbole vereinbaren, über die man die Anweisungsroutinen aufrufen kann.

Operation	Operanden	F-Modus / L-Modus
@USE	**COM**MAND = 'usersymb' [({ entry | * } [,modlib])]	

usersymb Benutzerfluchtsymbol für die externe Anweisungsroutine. Das Fluchtsymbol darf ein beliebiges Zeichen außer dem Anweisungssymbol, dem Semikolon und dem Leerzeichen sein.
Wird der EDT als Unterprogramm aufgerufen (CMD-Funktion, Version 2), darf als Fluchtsymbol auch der Leerstring angegeben werden (Sonderanwendung Anweisungsfilter (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]; Externe Anweisungsroutinen). Die einschließenden Hochkommas können durch @QUOTE umdefiniert werden.

entry Einsprungpunkt der externen Anweisungsroutine. entry kann auch über eine Zeichenfolgevariable angegeben werden. Das Modul oder die Ladeeinheit wird sofort geladen.

* Der Entryname der Anweisungsroutine wird bei der Eingabe der externen Anweisung angegeben (siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9]). Das Modul wird erst bei Eingabe der externen Anweisung geladen.

modlib Name der Bibliothek, in der das Modul oder die Ladeeinheit abgelegt ist. modlib kann auch über eine Zeichenfolgevariable angegeben werden.
Wird das Modul bzw. die Ladeeinheit in der angegebenen Bibliothek nicht gefunden, wird zuerst in den alternativen Bibliotheken BLSLIBxy gesucht , dann in der privaten Tasklib bzw. in der Systemtasklib $TASKLIB.
Ist keine Bibliothek angegeben, wird zunächst in der privaten Tasklib und dann in der Systemtasklib $TASKLIB gesucht.
Bei Mißerfolg wird eine Fehlermeldung ausgegeben.

Es können maximal 5 verschiedene Fluchtsymbole vereinbart werden.

Ist (entry[,modlib]) nicht angegeben, wird die mit dem angegebenen Symbol zuvor definierte Anweisungsroutine deaktiviert.

Das Modul bzw. die Ladeeinheit kann mit @UNLOAD entladen werden. Das zugehörige Fluchtsymbol wird zurückgenommen.

Schnittstelle zur externen Anweisungsroutine siehe Handbuch „EDT-Unterprogrammschnittstellen“ [9].

Der Entry IEDTCALL wird mit der Meldung abgewiesen: % EDT4933 MODULE LOADING NOT POSSIBLE, ebenso wie der Entry EDTC.

Beispiel 1

Entry durch @USE vorgegeben

```
Entryname      : JOBVAR
Syntax         : CATJV <name>
                 ERAJV <name>
                 GETJV <name>,<ln>
                 SETJV <name>,<ln>
Deklaration    : @USE COMMAND = '*' (JOBVAR,PRIVLIB)
Anwendung      : *CATJV JV.TEST   ==> Aufruf des Moduls JOBVAR mit
                                      Parameter 'CATJV JV.TEST'
                 *SETJV JV.TEST,3 ==> Aufruf des Moduls JOBVAR mit
                                      Parameter 'SETJV JV.TEST,3'
```

Beispiel 2

Entry durch externe Anweisung vorgegeben

```
Deklaration    : @USE COMMAND = '*' (*,PRIVLIB)
Anwendung      : *SORT 20-100     ==> Aufruf des Moduls SORT mit
                                      Parameter '20-100'
                 *HELP EDT5100    ==> Aufruf des Moduls HELP mit
                                      Parameter 'EDT5100'
```

@VDT Bildschirmausgabe steuern

@VDT verändert die Anzahl der Zeilen, die bei bildschirmweiser Ausgabe der virtuellen bzw. der durch @OPEN eröffneten Datei (siehe @PRINT V) auf einmal ausgegeben werden.

Operation	Operanden	F-Modus / L-Modus
@**V**DT	[int] [,] [**F1** | **F2**]	

int — Gibt an, wieviele Zeilen eine Bildschirmausgabe im F-Modus enthalten soll. Es muß ein Wert angegeben werden, der kleiner oder gleich dem Standardwert ist, den das System für die betreffende Datensichtstation festlegt. Fehlt int, nimmt der EDT den vom System vorgesehenen Standardwert.

F1 — Bestimmt den vom System vorgegebenen Standardwert mit 24 Zeilen und 80 Spalten.

F2 — Bestimmt das Bildschirmformat mit 27 Zeilen und 132 Spalten. Der Operand F2 kann nur dann gewählt werden, wenn die DSS 9763 dieses Bildschirmformat unterstützt. Ansonsten wird F2 mit einer Fehlermeldung abgewiesen (F-Modus) bzw. ignoriert (L-Modus).
Wird F1 oder F2 ohne int angegeben, wird gleichzeitig die zugehörige Zeilenzahl für den L-Modus eingestellt.

@VDT ohne Parameter wirkt global für alle Arbeitsdateien.

Beim Starten des EDT im F-Modus wird das Standardformat voreingestellt. Beim Umschalten in den L-Modus bleibt die aktuelle Formateinstellung bis zur ersten @VDT-Anweisung erhalten.

Wieviel Zeilen standardmäßig an der benutzten Datensichtstation ausgegeben werden, läßt sich zu Beginn der EDT-Sitzung mit @STATUS=VDT abfragen.

Der Operand F1 bzw. F2 wird nur bei der Datensichtstation 9763 akzeptiert. Anderfalls wird er im F-Modus mit der Meldung % EDT4945 NOT POSSIBLE ON THIS TERMINAL abgewiesen bzw. im L-Modus ignoriert.

Im F-Modus bricht @VDT die Verarbeitung einer Anweisungszeile ab, d.h. ein evtl. Rest der Anweisungszeile wird nicht abgearbeitet.

Falls der EDT mit K2 unterbrochen wurde, so wird nach Rückkehr in den F-Modus mit RESUME-PROGRAM das voreingestellte Bildschirmformat wiederhergestellt. Nach einer Unterbrechung im L-Modus wird nach Rückkehr mit SEND-MESSAGE TO=PROGRAMM das voreingestellte Bildschirmformat wieder aktiviert.

i

- @VDT baut das aktuelle Fenster mit dem gewünschten Format auf und bleibt bis zum nächsten Aufruf der Anweisung gültig.
- @VDT bewirkt implizit ein @PAR SPLIT = OFF.
- In Stapelprozessen wird die @VDT-Anweisung ignoriert.

@VTCSET Bildschirmausgabe steuern

@VTCSET bestimmt, ob bei der Ausgabe LINE-Modus-Steuerzeichen (siehe Handbuch „Makroaufrufe an den Ablaufteil“ [5], Makro WRTRD, MODE=LINE) aus Dateiinhalten ausgewertet werden oder in gerätespezifische Schmierzeichen umgesetzt werden.

Operation	Operanden	F-Modus / L-Modus
@**VT**CSET	[**ON**] | **OF**F	

ON Legt fest, daß die Ausgabe ungeprüft erfolgt und daher keine Umcodierung in Schmierzeichen erfolgt.

OFF Bewirkt, daß LINE-Modus-Steuerzeichen im Dateiinhalt bei der Ausgabe in ein gerätespezifisches Schmierzeichen umcodiert werden. Es werden auch alle nicht darstellbaren Zeichen umcodiert.

Für Ausgaben nach SYSLST (@LIST) in Stapelprozessen hat die Anweisung keine Funktion.

@WRITE Schreiben einer Datei oder eines Bibliothekselementes

@WRITE bietet in zwei Formaten folgende Funktionen:

– Schreiben des Inhalts der aktuellen Arbeitsdatei in eine SAM-Datei (Format 1).
– Schreiben des Inhalts der aktuellen Arbeitsdatei in ein Bibliothekselement (Format 2).

@WRITE (Format 1) Inhalt der aktuellen Arbeitsdatei in SAM-Datei schreiben

Mit @WRITE wird der Inhalt der virtuellen bzw. der durch @OPEN eröffneten Datei ganz oder teilweise als SAM-Datei auf Platte oder Band geschrieben.

Die SAM-Datei ist lediglich während der Zeit physikalisch geöffnet, in der @WRITE ausgeführt wird.

Operation	Operanden	F-Modus / L-Modus
@**W**RITE	['file'] [(ver)] [range*] [:col:] [**K**EY] [{**U**PDATE / **O**VERWRITE}]	

file Dateiname.
Existiert der mit file vereinbarte Dateiname noch nicht, wird eine Datei mit diesem Namen katalogisiert. Fehlt der Operand file, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag, dann der globale @FILE-Eintrag und zuletzt der implizite lokale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE); Ansonsten wird @WRITE mit einer Fehlermeldung abgewiesen.
Wenn der Dateikettungsname EDTSAM einer Datei zugeordnet ist, genügt die Angabe /, um diese Datei zurückzuschreiben (siehe Abschnitt „Dateibearbeitung“ auf Seite 40ff.).

ver Versionsnummer der Datei.
Sie kann aus bis zu drei Ziffern oder * bestehen. * bezeichnet die aktuelle Versionsnummer.

range* Zeilenbereich, bestehend aus:
– einer oder mehreren (durch Komma getrennten) Zeilennummern (z.B. 4,6,15)
– einem oder mehreren (durch Komma getrennten) Zeilenbereichen (z.B. 5-10,17-19)
– einer Kombination von einzelnen Zeilen und Zeilenbereichen (z.B. 4,7-23,8,15-30)

Der Zeilenbereich kann auch durch das aktuelle Zeilenbereichssymbol (siehe @RANGE), durch symbolische Zeilennummern (z.B. %,$) oder durch Zeilenummervariablen angegeben werden.
Zeichenfolgevariablen dürfen nicht angegeben werden.
Fehlt range*, werden alle Zeilen der Datei in die SAM-Datei geschrieben.

col Spaltenbereich bestehend aus:
- einer oder mehreren (durch Komma getrennten) Spalten (z.B. 10,15,8)
- einem oder mehreren (durch Komma getrennten) zusammenhängenden Spaltenbereichen (z.B. 15-25,18-23)
- einer Kombination von einzelnen Spalten und Spaltenbereichen (z.B. 10,14-29,23-50,17)

Wiederholungen und Überlappungen von Spalten und Spaltenbereichen sind erlaubt.
Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge gespeichert.

KEY Beim Schreiben der SAM-Datei wird jeder Zeile ein 8 Zeichen langer Schlüssel vorangestellt, der sich aus der jeweiligen Zeilennummer ergibt. Damit erreicht man, daß diese Datei später wieder mit genau denselben Zeilennummern eingelesen werden kann (siehe @READ mit Operand KEY.)

UPDATE Ist nur sinnvoll, wenn bereits eine SAM-Datei mit dem angegebenen Namen existiert.
UPDATE bewirkt, daß die abzuspeichernden Zeilen an das Ende der existierenden SAM-Datei angehängt werden.

Wird UPDATE nicht angegeben, wird die gesamte SAM-Datei überschrieben, d.h. ihr bisheriger Inhalt wird gelöscht.

OVERWRITE Unterdrückt die Abfrage OVERWRITE FILE? (Y/N). Eine bereits vorhandene Datei gleichen Namens wird überschrieben. Existiert die Datei file noch nicht, ist OVERWRITE wirkungslos.

Wird UPDATE oder OVERWRITE nicht angegeben und existiert bereits eine Datei mit gleichem Namen, reagiert der EDT mit der Frage:

```
% EDT0903 FILE 'file' IS IN THE CATALOG, FCBTYPE = fcbtyp
% EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)
```

Antwortet der Benutzer mit

N wird @WRITE nicht ausgeführt,

Y wird @WRITE ausgeführt und die bestehende Datei als SAM-Datei mit dem Inhalt der aktuellen Arbeitsdatei überschrieben.

Bei variabler Satzlänge (RECORD-FORMAT = VARIABLE) gehen beim Zurückschreiben ab Position 257 die Zeichen verloren.

i

Wird als Versionsnummer * angegeben, wird (nachdem die Datei auf Platte geschrieben wurde, nicht bei der Sicherungsabfrage) die aktuelle Versionsnummer am Bildschirm ausgegeben. Eine neue Versionsnummer entsteht dann, wenn eine Datei erstmalig angelegt oder aber eine bereits bestehende Datei verändert wird. Hierbei erhöht sich die Versionsnummer um 1. Eine neuangelegte Datei erhält nach dem Schreiben auf Platte die Versionsnummer 1. Das Hochzählen der Versionsnummer geschieht bis zur Maximalzahl 255. Die darauffolgende Versionsnummer ist 0. Versionsnummern sind vorgesehen, um Dateien vor unbeabsichtigtem Überschreiben zu schützen. Wird nämlich eine falsche Versionsnummer angegeben, wird die richtige Versionsnummer am Bildschirm ausgegeben; ein Zurückschreiben wird jedoch nicht durchgeführt.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @WRITE nach dem Zurückschreiben der Datei ein Coded Character Set Name (CCSN) als Codemerkmal mitgegeben.

Unabhängig davon, ob die Datei bereits existiert und welchen CCSN sie besitzt, wird mit @WRITE der aktuell im EDT eingestellte CCSN vergeben.

Beispiel

```
1.     EINE GANZ KURZE DATEI ---------------------------------------- (01)
2.     @WRITE 'TEST.@WRITE.1' --------------------------------------- (02)
2.     @FILE 'TEST.@WRITE.1' ---------------------------------------- (03)
2.     @WRITE UPDATE ------------------------------------------------ (04)
2.     @DELETE ------------------------------------------------------ (05)
1.     @READ -------------------------------------------------------- (06)
3.     @PRINT
1.0000 EINE GANZ KURZE DATEI
2.0000 EINE GANZ KURZE DATEI ---------------------------------------- (07)
3.
```

(01) 1 Zeile wird in die virtuelle Datei geschrieben.

(02) Diese Zeile wird als Datei TEST.@WRITE.1 auf Platte geschrieben.

(03) Der Dateiname TEST.@WRITE.1 wird über @FILE vereinbart.

(04) @WRITE bezieht sich auf den unter (03) vereinbarten Dateinamen. Mit UPDATE erreicht man, daß der Inhalt der virtuellen Datei - immer noch die unter (01) angelegte Zeile - an das Ende der Datei TEST.@WRITE.1 angehängt wird.

(05) Der Inhalt der virtuellen Datei wird gelöscht.

(06) Die Datei TEST.@WRITE.1 wird in die virtuelle Datei gebracht (auch hier braucht kein Dateiname angegeben werden.)

(07) Man sieht, daß unter (04) die Zeile an das Ende der Datei angehängt wurde.

@WRITE (Format 2)
Inhalt der aktuellen Arbeitsdatei in ein Bibliothekselement oder eine Datei schreiben

Mit @WRITE wird der Inhalt der aktuellen Arbeitsdatei in ein Bibliothekselement bzw. in eine Datei geschrieben. Die Arbeitsdatei bleibt dabei erhalten (vergleiche @CLOSE). Wenn die Bibliothek bzw. das Bibliothekselement noch nicht existiert, wird mit @WRITE die Bibliothek bzw. das Bibliothekselement oder die Datei erzeugt.

Operation	Operanden	F-Modus / L-Modus
@**W**RITE	[{ **L**IBRARY=path1 ([**E**LEMENT=]elemname [(vers)][,elemtyp]) / **E**LEMENT=elemname [(vers)][,elemtyp] / **F**ILE=path2 [,**T**YPE=**I**SAM | **S**AM] }] [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	

Ein Operand ist mindestens anzugeben.

Wird mehr als ein Operand angegeben, müssen die einzelnen Operanden durch Leerzeichen oder Komma voneinander getrennt werden.

LIBRARY = path1 (ELEMENT=elemname [(vers)][,elemtyp])
: Name des Elements mit Angabe des Bibliotheknamens.

ELEMENT = elemname [(vers)][,elemtyp]
: Namen des Elements ohne Angabe des Bibliotheknamens. Voraussetzung ist die Voreinstellung des Bibliotheknamens mit @PAR ELEMENT-TYPE.

path1
: Name der Bibliothek.
path1 kann auch als Zeichenfolgevariable angegeben werden.
Wird path1 nicht angegeben, wird der mit @PAR LIBRARY voreingestellte Bibliotheksname verwendet.

elemname
: Name des Elements. elemname kann auch als Zeichenfolgevariable angegeben werden.

vers
: Versionsbezeichnung des gewünschten Elements (siehe Handbuch „LMS“ [13]). Wird vers nicht angegeben oder *STD, wird das Element mit der höchst möglichen Version (X’FF’ dargestellt als @) erzeugt oder ersetzt.

elemtyp
: Typ des Elements.
elemtyp kann auch als Zeichenfolgevariable angegeben werden.
Zulässige Typangaben: S, M, P, J, D, X, *STD und freie Typnamen mit entsprechendem Basistyp. Falls nicht angegeben, wird der in @PAR ELEMENT-TYPE voreingestellte Wert verwendet.

Wird ein freier Typnamen verwendet, so liegt es in der Verantwortung des Benutzers, daß der zugehörige Basistyp einem zulässigen Typ S, M, P, J, D oder X entspricht.

Typ	Elementinhalt
S	Quellprogramme
M	Makros
P	Druckaufbereitete Daten
J	Prozeduren
D	Textdaten
X	Daten beliebigen Formats

*STD
Typ S ist die Voreinstellung nach Aufruf des EDT. Mit @PAR kann eine andere zulässige Typangabe als Voreinstellung festgelegt werden.

FILE = path2 Speichern der Arbeitsdatei in eine BS2000-Datei

path2 Name der Datei (vollqualifizierter Dateiname).
path2 kann auch als Zeichenfolgevariable angegeben werden.

TYPE Festlegen der Zugriffsmethode der Datei.

= SAM Standardwert. Die Datei wird als SAM-Datei gespeichert

= ISAM Die Datei wird als ISAM-Datei gespeichert

MODE Festlegen des Eröffnungsmodus für das Bibliothekselement bzw. der Datei.

Die Arbeitsdatei wird geschrieben:

= ANY in ein neues oder existierendes Bibliothekselement bzw. Datei.
Ist das Bibliothekselement bzw. die Datei noch nicht vorhanden, wird es durch @WRITE erzeugt.

= NEW in ein neu zu erstellendes Bibliothekselement bzw. eine Datei, d.h. es existiert in der Bibliothek noch kein Element bzw. keine Datei mit diesem Namen.
Ist das Bibliothekselement bzw. die Datei noch nicht vorhanden, wird es durch @WRITE erzeugt. Der Inhalt der aktuellen Arbeitsdatei wird in das Bibliothekselement bzw. die Datei geschrieben. Das Bibliothekselement wird anschließend geschlossen (implizites @OPEN und @CLOSE).

= REPLACE
in ein existierendes Bibliothekselement bzw. eine Datei, wobei der Inhalt vor dem Speichern gelöscht wird oder in ein neues Bibliothekselement bzw. eine Datei, das durch @WRITE erzeugt wird.

= UPDATE

in ein existierendes Bibliothekselement bzw. eine Datei, wobei der Inhalt vor dem Speichern gelöscht wird.
Wurde ein bestehendes Bibliothekselement bzw. eine bestehende Datei mit @OPEN (Format 2) geöffnet, werden mit @WRITE nur Zwischenstände gesichert. Das Bibliothekselement bzw. die Datei bleibt geöffnet, bis es mit @CLOSE geschlossen wird.

Wurde ein bestehendes Bibliothekselement oder eine Datei mit @OPEN (Format 2) öffnet, so kann die angabe von path, elemname und elemtyp bzw. path2 entfallen, wenn der Operand MODE angegeben ist. Der Inhalt des Bibliothekselementes bzw. der Datei wird durch den Inhalt der Arbeitsdatei ersetzt. Das Bibliothekselement bzw. die Datei bleibt geöffnet, bis es mit @CLOSE geschlossen wird.

Da MODE = ANY voreingestellt ist, werden bereits existierende Bibliothekselemente bzw. Dateien ohne Warnung überschrieben.

Beispiel

@WRITE LIBRARY = PROGLIB (ELEMENT = SYNT)

Die aktuelle Arbeitsdatei wird in das Bibliothekselement SYNT der Programmbibliothek PROGLIB geschrieben.

@WRITE ELEMENT = PROC.TSCHO, J

Die aktuelle Arbeitsdatei wird in das Bibliothekselement PROC.TSCHO mit dem Elementtyp J geschrieben (Prozedur als Inhalt). Die Bibliothek, in der das Element PROC.TSCHO stehen soll, muß dabei vorher durch @PAR LIBRARY voreingestellt werden.

Interaktion mit XHCS

Ab BS2000/OSD-BC V1.0 wird (wenn das Subsystem XHCS verfügbar ist) mit @WRITE nach dem Zurückschreiben der Datei (Bibliothekselement) ein Coded Character Set Name (CCSN) als Codemerkmal mitgegeben.

Unabhängig davon, ob die Datei (Bibliothekselement) bereits existiert und welchen CCSN sie besitzt, wird mit @WRITE der aktuell im EDT eingestellte CCSN vergeben.

@XCOPY Einlesen einer POSIX-Datei

Mit @XCOPY wird eine POSIX-Datei, die im POSIX-Dateisystem abgelegt ist, in die aktuelle Arbeitsdatei kopiert.

Diese Funktion wird erst ab BS2000/OSD-BC V2.0 unterstützt. POSIX und das zugehörige Laufzeitsystem CRTE müssen als Subsysteme aktiviert sein.

Operation	Operanden	F-Modus / L-Modus
@**XC**OPY	**F**ILE=xpath [,**C**ODE=**E**BCDIC | **I**SO]	

FILE Kopieren der Daten einer POSIX-Datei in die Arbeitsdatei.

xpath Pfadname der POSIX-Datei, deren Daten kopiert werden sollen.
Die Angabe von Unterverzeichnissen ist zulässig, solange die Länge von 256 Zeichen nicht überschritten wird.
xpath kann auch als Zeichenfolgevariable angegeben werden.

CODE Festlegung, in welchem Code die Daten in der Datei vorliegen. Fehlt der Operand CODE, so wird die Voreinstellung durch @PAR CODE herangezogen.

=EBCDIC EDT erwartet die Daten im EBCDI-Code.
Die Daten werden beim Einlesen und Schreiben nicht umcodiert, sondern binär übernommen.
Als Satztrennzeichen wird das Zeichen X'15' ausgewertet.

=ISO EDT erwartet die Daten im ISO-Code.
Die Daten werden beim Einlesen nach EBCDIC codiert.
Als Satztrennzeichen wird das Zeichen X'0A' ausgewertet.

Berechnung der Zeilennummern beim Einlesen

1. Standardnumerierung mit Standardschrittweite 1.0000 oder
2. Numerierung mit festgelegter Schrittweite gemäß @PAR INCREMENT oder
3. Automatische Numerierung bei @PAR RENUMBER=ON
   (siehe Berechnung der Zeilennummern beim Einlesen bei @OPEN).

Wird eine Zeile angelegt, deren Nummer größer als die bisher höchste Zeilennummer ist, so wird die aktuelle Zeilennummer verändert.

Interaktion mit XHCS

Soll die aktuelle Arbeitsdatei in eine BS2000-Datei mit einem bestimmten CCS-Namen geschrieben werden (mittels @WRITE, @SAVE), so muß die Einstellung des CCS-Namens mit @CODENAME vor @XCOPY erfolgen.
Ist dies nicht geschehen, muß nach dem Schreiben der BS2000-Datei der CCS-Namen mit dem Kommando SET-FILE-ATTRIBUTE selbst eingetragen werden.

@XOPEN Öffnen und Einlesen einer POSIX-Datei

Mit @XOPEN wird eine POSIX-Datei

- eröffnet, die im POSIX-Dateisystem abgelegt ist
- in die aktuelle Arbeitsdatei eingelesen
- als neue POSIX-Datei im POSIX-Dateisystem angelegt

Diese Funktion wird erst ab BS2000/OSD-BC V2.0 unterstützt. POSIX und das entsprechende Laufzeitsystem CRTE müssen als Subsysteme aktiviert sein.

Operation	Operanden	F-Modus / L-Modus
@**XO**PEN	**F**ILE=xpath [,**C**ODE=**E**BCDIC | **I**SO] [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	

FILE — POSIX-Datei eröffnen und einlesen.

xpath — Pfadname einer POSIX-Datei, bezogen auf das aktuelle Verzeichnis.
Die Angabe von Unterverzeichnissen ist zulässig, solange die Länge von 256 Zeichen nicht überschritten wird.
xpath kann auch als Zeichenfolgevariable angegeben werden.

CODE — Festlegung, in welcher Codierung die Daten in der Datei vorliegen und wie sie beim Schreiben in die Datei abgelegt werden sollen. Fehlt der Operand CODE, so wird die Voreinstellung durch @PAR CODE verwendet.
In der Arbeitsdatei liegen die Daten immer in EBCDI-Code vor.

=EBCDIC — EDT erwartet die Daten in EBCDI-Codierung.
Die Daten werden beim Einlesen und Schreiben nicht umcodiert, sondern binär übernommen.
Als Satztrennzeichen wird das Zeichen X'15' verwendet.

=ISO — EDT erwartet die Daten in ISO-Codierung.
Die Daten werden beim Einlesen nach EBCDIC codiert. Beim Schreiben mit @XWRITE auf die gleiche Datei und bei @CLOSE werden die Daten der Arbeitsdatei umcodiert und in der entsprechenden ISO-Varianten in die POSIX-Datei geschrieben.
Als Satztrennzeichen wird das Zeichen X'0A' verwendet.

MODE Festlegen des Eröffnungsmodus der Datei

= ANY Standardwert
Eine existierende oder eine neue Datei wird zur Bearbeitung eröffnet.

= UPDATE
Eine existierende Datei wird zur Bearbeitung eröffnet.

= NEW Eine Datei wird neu im aktuellen Verzeichnis angelegt.
Dabei darf die Datei noch nicht vorhanden sein.

=REPLACE
Der Inhalt der existierenden Datei soll ersetzt werden. Der Inhalt wird nicht in die Arbeitsdatei eingelesen.

Ist die Datei bereits in einer anderen Arbeitsdatei eröffnet, bzw. ist die aktuelle Arbeitsdatei nicht leer, erfolgt eine Fehlermeldung.

Berechnung der Zeilennummern beim Einlesen

1. Standardnumerierung mit Standardschrittweite 1.0000 oder
2. Numerierung mit festgelegter Schrittweite gemäß @PAR INCREMENT oder
3. Automatische Numerierung bei @PAR RENUMBER=ON
(siehe Berechnung der Zeilennummern beim Einlesen bei @OPEN).

Nach dem Einlesen wird die aktuelle Zeilennummer auf den Wert der letzten eingelesenen Zeile plus der aktuellen Schrittweite gesetzt.

Interaktion mit XHCS

Soll die aktuelle Arbeitsdatei in eine BS2000-Datei mit einem bestimmten CCS-Namen geschrieben werden (mittels @WRITE, @SAVE), so muß die Einstellung des CCS-Namens mit @CODENAME vor @XOPEN erfolgen.
Ist dies nicht geschehen, muß nach dem Schreiben der BS2000-Datei der CCS-Namen mit dem Kommando SET-FILE-ATTRIBUTE selbst eingetragen werden.

Beenden des EDT

Ist bei Beenden des EDT (@HALT, @END, @RETURN) eine Datei mit @XOPEN eröffnet und wird die Sicherungsabfrage % EDT0900 ausgegeben, so wird der POSIX-Dateiname in der Form 'X=xpath' aufgelistet.

@XWRITE Inhalt der aktuellen Arbeitsdatei in eine POSIX-Datei speichern

Mit @XWRITE wird der Inhalt der aktuellen Arbeitsdatei in eine POSIX-Datei des POSIX-Dateisystems geschrieben. Die Arbeitsdatei bleibt dabei erhalten. Wenn die Datei noch nicht existiert, wird sie erzeugt.

Diese Funktion wird erst ab BS2000/OSD-BC V2.0 unterstützt. POSIX und das entsprechende Laufzeitsystem CRTE müssen als Subsysteme aktiviert sein.

Operation	Operanden	F-Modus / L-Modus
@**XW**RITE	**F**ILE=xpath [,**C**ODE=**E**BCDIC | **I**SO] [,**M**ODE=**A**NY | **U**PDATE | **N**EW | **R**EPLACE]	

FILE Schreiben der Daten in eine POSIX-Datei.

xpath Pfadname der POSIX-Datei, in die die Daten geschrieben werden sollen. Die Angabe von Unterverzeichnissen ist zulässig, solange die Länge von 256 Zeichen nicht überschritten wird.
xpath kann auch als Zeichenfolgevariable angegeben werden.

CODE Festlegung, in welcher Codierung die Daten in der Datei abgelegt werden sollen. Fehlt der Operand CODE, so wird die Voreinstellung durch @PAR CODE verwendet.

=EBCDIC Die Daten werden beim Schreiben nicht umcodiert, sondern binär aus der Arbeitsdatei übernommen.
Als Satztrennzeichen wird das Zeichen X’15’geschrieben.

=ISO Die Daten der Arbeitsdatei werden in den entsprechenden ISO-Code umcodiert und in die POSIX-Datei geschrieben.
Als Satztrennzeichen wird das Zeichen X’0A’geschrieben.

MODE Festlegen des Eröffnungsmodus der Datei.

=ANY Standardwert. Die Arbeitsdatei wird in eine neue oder existierende Datei geschrieben. Ist die Datei noch nicht vorhanden, wird sie erzeugt.

=UPDATE Die Arbeitsdatei wird in eine existierende Datei geschrieben, wobei der Inhalt überschrieben wird.
Ist die angegebene Datei mit @XOPEN eröffnet worden, so bleibt die Codierung erhalten, eine Angabe des CODE-Operanden wird ignoriert.

=NEW Die Arbeitsdatei wird in eine neu zu erstellende Datei geschrieben.
Die Datei darf noch nicht vorhanden sein.

=REPLACE

Die Arbeitsdatei wird in eine existierende Datei geschrieben, wobei der Inhalt überschrieben wird.
Die Codierung kann sich ändern.

Wurde eine Datei mit @XOPEN eröffnet, so kann die Angabe des Dateinamens bei @XWRITE entfallen, wenn der Operand MODE angegeben ist. Der Inhalt der Datei wird durch den Inhalt der Arbeitsdatei ersetzt. Die Datei bleibt geöffnet, bis @CLOSE abgesetzt wird.

Da MODE=ANY voreingestellt ist, werden bei fehlender MODE-Angabe bereits existierende Dateien ohne Warnung überschrieben.

7 Meldungen des EDT

EDT0001 (&00) STARTED
EDT0001 (&00) GESTARTET

EDT0100 TESTMODE: NO SYNTAX ERROR
EDT0100 TESTMODUS: KEIN SYNTAX-FEHLER

Bedeutung
Der Testmodus ist eingeschaltet. Bei der syntaktischen Pruefung ist kein Fehler aufgetreten. Die Anweisungen werden nicht ausgefuehrt.
Fehlerschalter: keiner.

EDT0110 TESTMODE: SYNTAX CANNOT BE TESTED
EDT0110 TESTMODE: SYNTAX KANN NICHT GEPRUEFT WERDEN

Bedeutung
Der Testmodus ist eingeschaltet. Die Anweisung kann nur bei der Ausfuehrung syntaktisch geprueft werden, da sie z.B. indirekte Operanden, Operanden in Variablen enthaelt oder eine Benutzeranweisung ist.
Fehlerschalter: keiner.

EDT0120 TESTMODE: CHARACTER(S) SKIPPED
EDT0120 TESTMODUS: ZEICHEN WURDEN UEBERLESEN
(B) Routing code: * Weight: 99

Bedeutung
Der Testmodus ist eingeschaltet. Bei der Syntaxpruefung im Line-Modus wurden Zeichen ueberlesen, was bei einer strengen Pruefung mit SECURITY=HIGH zu Fehler fuehren koennte.
Fehlerschalter: keiner.

Maßnahme
Informieren Sie sich im EDT-Handbuch ueber die korrekte Syntax der Anweisung. Korrigieren Sie die Anweisung, wenn diese auch in spaeteren EDT Versionen ablaufen soll. Nur die im Manual beschriebene Form ist garantiert.

EDT0160 FILE '(&00)' WRITTEN
EDT0160 DATEI '(&00)' GESCHRIEBEN

EDT0170 MEMBER '(&00)' IN LIBRARY '(&01)' REPLACED AND WRITTEN
EDT0170 ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' ERSETZT UND GESCHRIEBEN

```
EDT0171    FILE '(&00)' REPLACED AND WRITTEN
EDT0171    DATEI '(&00)' ERSETZT UND GESCHRIEBEN

EDT0172    MEMBER '(&00)' IN LIBRARY '(&01)' CREATED AND WRITTEN
EDT0172    ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' ERZEUGT UND GESCHRIEBEN

EDT0173    FILE '(&00)' CREATED AND WRITTEN
EDT0173    DATEI '(&00)' ERZEUGT UND GESCHRIEBEN

EDT0178    FILE '(&00)' CLOSED
EDT0178    DATEI '(&00)' GESCHLOSSEN

EDT0190    WORK FILE (&00) EMPTY
EDT0190    ARBEITSDATEI (&00) LEER

EDT0192    FILE '(&00)' OPENED REAL IN WORK FILE (&01)
EDT0192    DATEI '(&00)' REAL GEOEFFNET IN ARBEITSDATEI (&01)
```

(B) Routing code: * Weight: 99

```
EDT0193    WORK FILE (&00) CLEARED
EDT0193    ARBEITSDATEI (&00) GELOESCHT

EDT0194    FILE '(&00)' CREATED AND OPENED REAL IN WORK FILE (&01)
EDT0194    DATEI '(&00)' ERZEUGT UND REAL GEOEFFNET IN ARBEITSDATEI (&01)

EDT0195    FILE '(&00)' REPLACED AND OPENED REAL IN WORK FILE (&01)
EDT0195    DATEI '(&00)' ERSETZT UND REAL GEOEFFNET IN ARBEITSDATEI (&01)

EDT0200    CCS CHANGED TO '(&00)'
EDT0200    CCS GEAENDERT AUF '(&00)'
```

(B) Routing code: * Weight: 99

Bedeutung
Da eine Datei oder ein Bibliothekselement mit der Code-Eigenschaft (&00) gelesen oder eroeffnet wurde, verwendet EDT nun dieses Coded Character Set.
Fehlerschalter: wird nicht gesetzt.

```
EDT0210    ELEMENT(S) ADDED TO S-VARIABLE '(&00)'
EDT0210    S-VARIABLE '(&00)' UM EIN ODER MEHRERE ELEMENTE ERWEITERT
```

(B) Routing code: * Weight: 99

Bedeutung
Die SDF-P-Listenvariable (&00) wurde erweitert, indem am Ende oder am Anfang ein oder mehrere Elemente hinzugefuegt wurden.
Fehlerschalter: keiner.

EDT0211 /FREE-VARIABLE COMMAND PROCESSED FOR S-VARIABLE '(&00)'
EDT0211 /FREE-VARIABLE KOMMANDO FUER S-VARIABLE '(&00)' AUSGEFUEHRT
(B) Routing code: * Weight: 99

Bedeutung
Der Inhalt der SDF-P-Variablen (&00) wurde geloescht. In der Anweisung @SETLIST mit dem Operand MODE=NEW wurde ein leerer Zeilen- und Spaltenbereich angegeben.
Fehlerschalter: keiner.

EDT0227 ISAM FILE '(&00)' CREATED AND OPENED IN WORK FILE (&01)
EDT0227 ISAM-DATEI '(&00)' ERZEUGT UND GEOEFFNET IN ARBEITSDATEI (&01)

EDT0228 ISAM FILE '(&00)' REPLACED AND OPENED IN WORK FILE (&01)
EDT0228 ISAM-DATEI '(&00)' ERSETZT UND GEOEFFNET IN ARBEITSDATEI (&01)

EDT0229 ISAM FILE '(&00)' OPENED IN WORK FILE (&01)
EDT0229 ISAM-DATEI '(&00)' GEOEFFNET IN ARBEITSDATEI (&01)

EDT0230 FILE '(&00)' OPENED IN CURRENT WORK FILE (&01)
EDT0230 DATEI '(&00)' IN AKTUELLER ARBEITSDATEI (&01) GEOEFFNET

EDT0231 FILE '(&00)' CREATED AND OPENED IN CURRENT WORK FILE (&01)
EDT0231 DATEI '(&00)' IN AKTUELLER ARBEITSDATEI (&01) ANGELEGT UND GEOEFFNET

EDT0232 FILE '(&00)' REPLACED AND OPENED IN WORK FILE (&01)
EDT0232 DATEI '(&00)' ERSETZT UND EROEFFNET IN ARBEITSDATEI (&01)

EDT0235 FILE '(&00)' WRITTEN AND CLOSED
EDT0235 DATEI '(&00)' GESCHRIEBEN UND GESCHLOSSEN

EDT0236 FILE '(&00)' CLOSED UNCHANGED
EDT0236 DATEI '(&00)' UNVERAENDERT GESCHLOSSEN

EDT0242 FILE '(&00)' COPIED
EDT0242 DATEI '(&00)' KOPIERT

EDT0258 MEMBER '(&00)' IN LIBRARY '(&01)' OPENED
EDT0258 ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' GEOEFFNET

EDT0259 MEMBER '(&00)' IN LIBRARY '(&01)' CREATED AND OPENED
EDT0259 ELEMENT '(&00)' IN DER BIBLIOTHEK '(&01)' ANGELEGT UND GEOEFFNET

EDT0264 MEMBER '(&00)' IN LIBRARY '(&01)' WRITTEN AND CLOSED
EDT0264 ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' GESCHRIEBEN UND GESCHLOSSEN

EDT0265 MEMBER '(&00)' IN LIBRARY '(&01)' CLOSED UNCHANGED
EDT0265 ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' UNVERAENDERT GESCHLOSSEN

```
EDT0266    WORK FILE EMPTY: MEMBER '(&00)' CLOSED UNCHANGED
EDT0266    LEERE ARBEITSDATEI: ELEMENT '(&00)' UNVERAENDERT GESCHLOSSEN
```

Bedeutung
Die in der CLOSE bzw. WRITE-Anweisung angegebene Arbeitsdatei ist leer.
Das Element (&00) wurde geschlossen, aber nicht zurueckgeschrieben.

```
EDT0268    MEMBER '(&00)' IN LIBRARY '(&01)' OPENED FOR REPLACEMENT
EDT0268    ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' ZUM ERSETZEN GEOEFFNET
```

```
EDT0274    MEMBER '(&00)' IN LIBRARY '(&01)' COPIED
EDT0274    ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' KOPIERT
```

```
EDT0281    /DELETE-FILE COMMAND PROCESSED FOR FILE '(&00)'
EDT0281    /DELETE-FILE KOMMANDO FUER DATEI '(&00)' AUSGEFUEHRT
```

Bedeutung
Die Datei wurde aus dem Katalog entfernt.

```
EDT0282    DELETE PROCESSED FOR MEMBER '(&00)'
EDT0282    LOESCHEN DES ELEMENTES '(&00)' AUSGEFUEHRT
```

Bedeutung
Das Element wurde aus der Bibliothek geloescht.

```
EDT0285    SDF: SYNTAX TESTED. (&00) ERROR(S) IN RANGE
EDT0285    SDF: SYNTAX GEPRUEFT. (&00) FEHLER IM ZEILENBEREICH
```
(B) Routing code: * Weight: 99

Bedeutung
Bei Abarbeitung der Anweisung @SDFTEST wurden (&00) fehlerhafte Zeilen entdeckt.
Fehlerschalter: keiner.

```
EDT0290    ALL LINES ARE DIFFERENT
EDT0290    ALLE ZEILEN UNGLEICH
```

Bedeutung
Alle zu vergleichenden Zeilen sind ungleich.
Fehlerschalter: EDT.

```
EDT0291    ALL LINES ARE EQUAL
EDT0291    ALLE ZEILEN GLEICH
```

Bedeutung
Alle zu vergleichende Zeilen sind gleich.
Fehlerschalter: wird nicht gesetzt.

EDT0292 COPY BUFFER CLEARED
EDT0292 KOPIERPUFFER GELOESCHT

Bedeutung
Quittung nach '*' in der Markierungsspalte.
Fehlerschalter: wird nicht gesetzt.

EDT0293 FILE NOT WRITTEN
EDT0293 DATEI NICHT GESCHRIEBEN

Bedeutung
Nach OVERWRITE-Abfrage wurde N eingegeben, oder ein Fehler trat beim Schreiben der Datei auf.
Fehlerschalter: wird nicht gesetzt.

EDT0294 MAXIMUM LINE NUMBER
EDT0294 MAXIMALE ZEILENNUMMER

Bedeutung
Beim Bildschirmaufbau wird die Zeilennummer 9999 ueberschritten.
Am Dateiende werden keine Leerzeilen zum Einfuegen angeboten.
Naehere Information darueber, wann die maximale Zeilennummer (9999.9999) erzeugt wird, kann dem EDT-Handbuch entnommen werden.
Fehlerschalter: wird nicht gesetzt.

EDT0295 OLD COPY BUFFER CLEARED, NEW COPY BUFFER FILLED
EDT0295 ALTER KOPIERPUFFER GELOESCHT, NEUER KOPIERPUFFER GEFUELLT

Bedeutung
Auf eine R-Markierung folgte eine C- oder M-Markierung.
Der durch die R-Markierung(en) angelegte Kopierpuffer wurde geloescht.
Fehlerschalter: wird nicht gesetzt.

EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)
EDT0296 DATEI UEBERSCHREIBEN? ANTWORT (Y=JA; N=NEIN)

Bedeutung
Abfrage nach den Anweisungen @WRITE bzw. @SAVE, wenn diese Datei schon existiert.
Fehlerschalter: wird nicht gesetzt.

Maßnahme
Y: Die Datei wird ueberschrieben
N: Die Datei wird nicht ueberschrieben.

```
EDT0297    COMPARE RESULT IN WORK FILE (&00)
EDT0297    VERGLEICHSERGEBNIS IN ARBEITSDATEI (&00)
```

Bedeutung
Das Ergebnis der erfolgreich durchgefuehrten Anweisung @COMPARE (Format 2) wird in der Arbeitsdatei (&00) ausgegeben.
Fehlerschalter: EDT.

```
EDT0298    ERASE ALL JOB VARIABLES '(&00)'? REPLY (Y=YES; N=NO)
EDT0298    ALLE JOBVARIABLEN '(&00)' ENTFERNEN? ANTWORT (Y=JA; N=NEIN)
```

Bedeutung
Abfrage nach der Anweisung @ERAJV, wenn der angegebene Name teilqualifiziert oder in Wildcard-Syntax angegeben wurde und mehr als eine Jobvariable davon betroffen ist.
Fehlerschalter: wird nicht gesetzt.

Maßnahme
Y: Die betroffenen Jobvariablen werden aus dem Katalog entfernt.
N: Die Anweisung wird abgebrochen undkeine Jobvariable wird entfernt.

```
EDT0299    JOB VARIABLES NOT ERASED
EDT0299    JOBVARIABLEN NICHT ENTFERNT
```

Bedeutung
Meldung EDT0298 (ALLE JV'S ENTFERNEN?) wurde mit N beantwortet.
Fehlerschalter: wird nicht gesetzt.

```
EDT0300    (&00)
EDT0300    (&00)
```

Bedeutung
Nach der @TMODE-Anweisung werden die Eigenschaften des Prozesses in dieser Reihenfolge von links nach rechts ausgegeben:

TSN	- Task-Folgenummer
USER-ID	- Benutzerkennung des /LOGON-Kommandos
ACCOUNT	- Abrechnungsnummer des Tasks
CPU-TIME	- verbrauchte CPU-Zeit
DATE	- Datum (JJ-MM-TT)
TIME	- Uhrzeit
'@'SYMBOL	- aktuelles Anweisungssymbol
TERMINAL	- Typ der Datensichtstation.

EDT0600 LOGICAL LINELENGTH > LENGTH OF SCREENLINE
EDT0600 LOGISCHE ZEILENLAENGE > BILDSCHIRMBREITE
(B) Routing code: * Weight: 99

Bedeutung
Bei der Initialisierung kann die logische Zeilenlaenge nicht mit der Laenge der Bildschirmzeile in Einklang gebracht werden.
Fehlerschalter: keiner.

EDT0610 BUFFER SIZE UNCHANGED
EDT0610 BUFFER GROESSE NICHT VERAENDERT
(B) Routing code: * Weight: 99

Bedeutung
Der Buffer fuer die Ausgabe am Bildschirm konnte von EDT nicht angepasst werden.
Fehlerschalter: keiner.

EDT0650 UNABLE TO SUPPORT NATIONAL TERMINAL. STANDARD WILL BE USED
EDT0650 UNTERSTUETZUNG DES NATIONALEN TERMINALS NICHT MOEGLICH. STANDARD EINSTELLUNG WIRD VERWENDET
(B) Routing code: * Weight: 99

Bedeutung
Die angeschlossene DSS ist ein nationales 7-bit Terminal, kann aber von EDT nicht als solches unterstuetzt werden. Moegliche Ursachen:
- Die DSS wurde mit falschen Parametern generiert, bzw. eine Variante, angegeben, die EDT noch nicht beruecksichtigt.
- Es liegt ein XHCS- oder VTSU-Fehler vor.

Maßnahme
Terminal neu generieren.

EDT0651 CCS '(&00)' INCOMPATIBLE WITH TERMINAL. STANDARD WILL BE USED
EDT0651 CCS '(&00)' UNVERTRAEGLICH MIT DATENSICHTSTATION. NUR STANDARDBETRIEB MOEGLICH

Bedeutung
Die Datei oder das Bibliothekselement, das eingelesen oder eroeffnet werden sollte, hatte das Codemerkmal (&00), oder in der Anweisung @CODENAME war der CCS Name (&00) angegeben worden. An dieser Datensichtstation ist es aber nur moeglich, Dateien mit dem CCS-Attribut EDF03IRV oder ohne CCS-Attribut zu bearbeiten.

Maßnahme
Mit dem /MODIFY-FILE-ATTRIBUTES -Kommando den CCS-Namen der Datei auf EDF03IRV aendern oder loeschen.
Die Anweisung @CODENAME sollte an dieser Datensichtstation nicht verwendet werden.

```
EDT0800    STATEMENT '(&00)' ONLY SUPPORTED UP TO THIS VERSION
EDT0800    ANWEISUNG '(&00)' WIRD LETZTMALIG UNTERSTUETZT
```

Bedeutung
Die Anweisung (&00) wird in der naechsten EDT-Version in dieser Form nicht mehr unterstuetzt.

Maßnahme
Handbuch zur Hilfe nehmen und Anweisung durch korrekte Form ersetzen.

```
EDT0900    EDITED FILE(S) NOT SAVED!
EDT0900    EDITIERTE DATEI(EN) NICHT GESICHERT!
```

Bedeutung
Die EDT-Sitzung sollte mit @HALT beendet werden, doch es gibt noch ungesicherte Daten. EDT gibt eine Liste der Arbeitsdateien aus, in denen sich ungesicherte Daten befinden.
Fehlerschalter: wird nicht gesetzt.

```
EDT0901    NO MATCH IN RANGE
EDT0901    KEIN TREFFER
```

Bedeutung
Keine Uebereinstimmung fuer die 1. Zeichenfolge in der @ON-Anweisung. Tritt dieser Fehler bei der Ausfuehrung einer EDT-Prozedur (@DO) oder INPUT-Datei auf, so wird diese Meldung nicht ausgegeben und der EDT-Fehlerschalter nicht gesetzt, sofern nicht die Protokollierung ueber den PRINT-Operanden eingeschaltet ist.
Fehlerschalter: EDT (siehe Bedeutung).

```
EDT0902    FILE (&00) VERSION (&01)
EDT0902    DATEI (&00) VERSION (&01)
```

Bedeutung
Bei der Anweisung @WRITE, @SAVE,... wurde ein * bzw. eine Versionsnummer angegeben. Die Version stimmte ueberein, und EDT gibt die aktuelle Versionsnummer aus.

```
EDT0903    FILE '(&00)' IS IN THE CATALOG, FCBTYPE = (&01)
EDT0903    DATEI '(&00)' IST IM KATALOG, FCBTYPE = (&01)

EDT0904    TERMINATE EDT? REPLY (Y=YES; N=NO)
EDT0904    EDT BEENDEN? ANTWORT (Y=JA; N=NEIN)
```

Bedeutung
EDT gibt eine Dialogabfrage aus, ob EDT beendet werden soll.
Fehlerschalter: wird nicht gesetzt.

Maßnahme
Y: EDT wird beendet
N: EDT wird nicht beendet.

```
EDT0905    EDITED MEMBER TO BE ADDED? REPLY (Y=YES; N=NO)
EDT0905    SOLL ELEMENT IN BIBLIOTHEK AUFGENOMMEN WERDEN? ANTWORT (Y=JA; N=NEIN)
```

Bedeutung
Bevor EDT zum LMS zurueckkehrt, gibt er eine Dialogabfrage aus,
ob die editierte Arbeitsdatei von LMS gesichert werden soll.

```
EDT0906    REPEAT ATTEMPT? REPLY (Y=YES; N=NO)
EDT0906    VERSUCH WIEDERHOLEN? ANTWORT (Y=JA; N=NEIN)
```

Bedeutung
Ist fuer eine Anweisung zuwenig virtueller Speicher verfuegbar, besteht die Moeglichkeit, die Anweisung nach geeigneten Massnahmen zu wiederholen.

Maßnahme
Y: Versuch wird wiederholt.
N: Anweisung wird abgebrochen.

```
EDT0907    NO PROCEDURE FILES DECLARED
EDT0907    KEINE PROZEDURDATEIEN VORHANDEN
```

Bedeutung
Eine '@DROP ALL'-Anweisung wurde gegeben,
aber es sind keine Prozedurdateien vorhanden.

```
EDT0909    AUTOSAVE ABORTED. ERASE SAVING FILES? REPLY(Y=YES; N=NO)
EDT0909    AUTOSAVE WURDE ABGEBROCHEN. SICHERUNGSDATEIEN LOESCHEN? ANTWORT (Y=JA; N=NEIN)
```
(B) Routing code: * Weight: 99

Bedeutung
Das Schreiben der Sicherungsdateien konnte nicht ausgefuehrt werden.
Die Autosave-Funktion muss abgebrochen werden.
Moegliche Ursachen: Speicherengpass, unvorhergesehene DVS-Fehler.
Fehlerschalter: keiner.

Maßnahme
Y: Die vorhandenen Sicherungsdateien werden geloescht.
N: Die vorhandenen Sicherungsdateien bleiben erhalten.

```
EDT0910    '@RENUMBER': LINES WILL BE LOST
EDT0910    '@RENUMBER': ZEILEN WERDEN VERLOREN GEHEN
```
(B) Routing code: * Weight: 99

Bedeutung
Eine @RENUMBER-Anweisung wurde eingegeben, um die Zeilen umzunumerieren. Wenn EDT in der angegebenen Weise umnumeriert, wird die groesstmoegliche Zeilennummer (9999.9999) erreicht und der Rest der Datei ginge verloren.

Maßnahme
Bevor neu numeriert wird, kannmit der Anweisung @LIMIT die Information ueber die Anzahl der Zeilen in der Datei eingeholt werden.

```
EDT0911    CONTINUE PROCESSING? REPLY (Y=YES; N=NO)
EDT0911    ABARBEITUNG FORTSETZEN? ANTWORT (Y=JA; N=NEIN)
```
(B) Routing code: * Weight: 99

Bedeutung
Bei der Abarbeitung einer Anweisung wurde eine fehlerhafte Situation entdeckt. EDT fraegt nach, ob die Anweisung noch weiter bearbeitet werden soll.

Maßnahme
Y: Die Abarbeitung wird fortgesetzt.
N: Die Anweisung wird abgebrochen.

```
EDT0999    (&00)
EDT0999    (&00)
```

Bedeutung
(&00): Meldung von externer Routine.

```
EDT1115    'COPY': NO RECORD EXISTS IN SPECIFIED RANGE
EDT1115    'COPY': IM ANGEGEBENEN SATZBEREICH KEIN SATZ VORHANDEN
```

Bedeutung
Der in der COPY-Anweisung angegebene Satzbereich kann nicht kopiert werden, da die Saetze nicht vorhanden sind.

```
EDT1137    SPECIFIED WORK FILE IGNORED IN CONJUNCTION WITH 'SPLIT'
EDT1137    ARBEITSDATEI-ANGABE IN VERBINDUNG MIT 'SPLIT' IGNORIERT
```

Bedeutung
In der @PAR-Anweisung wurde die Arbeitsdateivariable (adatvar) als erster Operand angegeben. Die Aktionen der @PAR-Anweisung sollen nur fuer diese Arbeitsdatei durchgefuehrt werden.
Der Operand SPLIT wirkt hingegen immer global.

```
EDT1150    NAME OF PLAM LIBRARY MEMBER TRUNCATED AFTER 64 CHARACTERS
EDT1150    NAME DES PLAM-BIBLIOTHEKS-ELEMENTS NACH 64 ZEICHEN ABGESCHNITTEN
```

EDT1151 VERSION OF PLAM LIBRARY MEMBER TRUNCATED AFTER 24 CHARACTERS
EDT1151 VERSION DES PLAM-BIBLIOTHEKS-ELMENTS NACH 24 ZEICHEN ABGESCHNITTEN

EDT1174 FILE ATTRIBUTES IGNORED
EDT1174 DATEIATTRIBUTE IGNORIERT

Bedeutung
Mit der @WRITE-Anweisung (Format 2) wird eine interne Arbeitsdatei in die die zugeordnete externe BS2000-Datei zurueckgeschrieben. Die Vergabe von Dateiattributen wird ignoriert, da fuer die externe Datei bereits Attribute definiert sind und sie mit der @WRITE Anweisung nicht geaendert werden duerfen.

EDT1180 CODE ATTRIBUTE IGNORED
EDT1180 CODE-ATTRIBUT IGNORIERT
(B) Routing code: * Weight: 99

Bedeutung
Mit der XWRITE-Anweisung wird die aktuelle Arbeitsdatei in die vorher mit XOPEN eroeffnete POSIX-Datei zurueckgeschrieben. Die Vergabe des CODE-Attributes wird ignoriert, da fuer diese Datei bereits ein CODE-Attribut definiert ist.
Bei Angabe von MODE=UPDATE kann dieses Attribut nicht geaendert werden.
Fehlerschalter: keiner.

Maßnahme
Eine Aenderung des CODE-Merkmals ist folgendermassen zu erreichen: Rueckschreiben der Arbeitsdatei mit MODE=REPLACE und gewuenschtem CODE-Operand und anschliessend Datei mit @CLOSE NOWRITE schliessen.

EDT1190 WORK FILE (&00) IS EMPTY. COPY OPERATION NOT PERFORMED
EDT1190 ARBEITSDATEI (&00) IST LEER. KOPIEREN NICHT DURCHGEFUEHRT

EDT1226 SPECIFIED FCBTYPE IGNORED: '(&00)' IS ASSUMED
EDT1226 ANGABE DES FCBTYPES IGNORIERT. '(&00)' WIRD ANGENOMMEN

Bedeutung
Der in der @OPEN- oder @WRITE-Anweisung (Format 2) angegebene FCBTYPE stimmt nicht mit dem Katalogeintrag ueberein. Die Angabe wird ignoriert und FCBTYPE (&00) aus dem Katalogeintrag uebernommen.

EDT1227 CCS ATTRIBUTE CANNOT BE SET.
EDT1227 CCS-ATTRIBUT KANN NICHT VERGEBEN WERDEN
(B) Routing code: * Weight: 99

Bedeutung
Die Datei wurde erzeugt oder zurueckgeschrieben, aber das CCS-Attribut kann nicht vergeben werden. Moegliche Ursachen sind z.B., dass die Datei einem fremden Benutzer gehoert oder in einem fremden System liegt (bei Zugriff ueber RFA).

```
EDT1243    FILE '(&00)' TO BE COPIED IS EMPTY
EDT1243    KOPIERDATEI '(&00)' LEER

EDT1244    FILE '(&00)' EMPTY
EDT1244    DATEI '(&00)' LEER

EDT1245    JV IS EMPTY
EDT1245    JV LEER
```

(B) Routing code: * Weight: 99

Bedeutung
Es wurde versucht, mit @GETJV den Wert einer Jobvariablen zu lesen, aber der Wertebereich ist leer, d.h. die Laenge des Jobvariablen-Wertes ist Null.
Fehlerschalter: EDT.

```
EDT1253    (SOME) RECORD(S) TRUNCATED
EDT1253    (EINIGE) SAETZE ABGESCHNITTEN
```

Bedeutung
Beim Einlesen in die interne Arbeitsdatei werden Saetze abgeschnitten, die laenger sind als 256 Byte.

```
EDT1254    NO MARKS SET FOR FILE TO BE PROCESSED IN REAL MODE
EDT1254    KEINE MARKIERUNGEN IN EINER REAL ZU BEARBEITENDEN DATEI
```

Bedeutung
In einer Arbeitsdatei, die real bearbeitet wird (mit @OPEN Format 1 eingelesen), koennen keine Markierungen gesetzt werden.
Fehlerschalter: EDT.

```
EDT1901    ISAM FILE. '@GET' STATEMENT PROCESSED
EDT1901    ISAM DATEI. '@GET'-ANWEISUNG AUSGEFUEHRT
```

Bedeutung
Eine @READ-Anweisung wurde fuer eine ISAM-Datei eingegeben. Der EDT fuehrt automatisch eine @GET-Anweisung aus.
Fehlerschalter: EDT.

```
EDT1902    SAM FILE. '@READ' STATEMENT PROCESSED
EDT1902    SAM DATEI. '@READ'-ANWEISUNG AUSGEFUEHRT
```

Bedeutung
Eine @GET-Anweisung wird auf eine SAM-Datei gegeben. Der EDT fuehrt automatisch ein @READ aus.
Fehlerschalter: EDT.

EDT1903 INPUT TRUNCATED
EDT1903 EINGABE ABGESCHNITTEN

Bedeutung
Mehr als 256 Zeichen werden fuer eine Zeile gelesen.
Fehlerschalter: EDT.

EDT1904 SOME LINES > 256
EDT1904 EINIGE ZEILEN > 256

Bedeutung
Einige der Zeilen, auf die durch eine @GET- oder @READ-Anweisung zugegriffen wird, sind laenger als 256 Zeichen. Die Zeilen werden nach 256 Zeichen gekuerzt.
Fehlerschalter: EDT.

EDT1905 INPUT TOO LONG. CORRECT INPUT
EDT1905 EINGABE ZU LANG. EINGABE KORRIGIEREN

Bedeutung
Ein Abbruchfehler beim Lesen tritt unter einer der folgenden Bedingungen auf:
- bei der Anweisung @CREATE...READ, wenn die Eingabe > 256 Byte ist, oder
- wenn bei @PRINT und der Eingabeaufforderung * + -0 eine Eingabe > 284 erfolgt, oder
- bei Angabe von indirekten Operanden, falls die Gesamtlaenge aus Operation und der Zeichenfolge in der Zeichenfolgevariablen 256 uebersteigt.

Fehlerschalter: wird nicht gesetzt.

EDT1906 TOO MANY NAMES. LIST INCOMPLETE
EDT1906 ZU VIELE NAMEN. LISTE UNVOLLSTAENDIG

Bedeutung
Die fuer den FSTAT bereitgestellten 15 Seiten reichen fuer alle Dateinamen nicht aus, oder die fuer STAJV bereitgestellten 8 Seiten reichen fuer alle Jobvariablennamen nicht aus, oder die fuer CMD bereitgestellten 8 Seiten reichen fuer die Ausgabe in den Puffer nicht aus.
Die Liste (der Namen) ist unvollstaendig.
Fehlerschalter: EDT.

```
EDT1907    MODULE CANNOT BE UNLOADED
EDT1907    MODUL KANN NICHT ENTLADEN WERDEN
```

Bedeutung
Der Modul, der in der @RUN- oder @UNLOAD-Anweisung angegeben wurde, konnte nicht entladen werden. Es wurde ein falscher Modulname angegeben, oder der Modul ist entladen.
Fehlerschalter: EDT.

```
EDT1936    MODIFIED LINE >256 CHARACTERS
EDT1936    MODIFIZIERTE ZEILE > 256 ZEICHEN
```

Bedeutung
Eine aufbereitete Zeile wurde beim Veraendern zu lang. Dieser Fehler kann durch eine @ON- , @PREFIX-, @COL- oder @CREATE-Anweisung hervorgerufen werden. Ferner kann eine erweiterte Prozedurzeile mit formalen Operanden zu lang werden. Die Zeile wird nach 256 Zeichen abgeschnitten.
Wenn bei einer @SETJV-Anweisung die aufbereitete Zeichenkette zu lang wird, werden die ersten 256 Zeichen der Jobvariablen als Wert zugewiesen.
Fehlerschalter: EDT.

```
EDT2169    WORK FILE (&00) IS EMPTY. WRITE OPERATION NOT PERFORMED
EDT2169    ARBEITSDATEI (&00) IST LEER. SCHREIBEN NICHT DURCHGEFUEHRT
```

Bedeutung
Die Anweisung @WRITE oder @XWRITE konnte nicht ausgefuehrt werden, da die Arbeitsdatei (&00) leer ist.
Fehlerschalter: nicht gesetzt.

```
EDT2266    WORK FILE IS EMPTY: MEMBER '(&00)' CLOSED UNCHANGED
EDT2266    ARBEITSDATEI IST LEER: ELEMENT '(&00)' UNVERAENDERT GESCHLOSSEN
```

Bedeutung
Da die in der @CLOSE- bzw. @WRITE-Anweisung (Format 2) angegebene Arbeitsdatei leer ist, wurde das Element (&00) geschlossen aber nicht zurueckgeschrieben.

```
EDT2267    LINE TRUNCATED AFTER (&00) CHARACTERS
EDT2267    ZEILE NACH (&00) ZEICHEN ABGESCHNITTEN
```

Bedeutung
Der eingegebene Satz ueberschreitet die in der @PAR-Anweisung als LIMIT angegebene Laenge und wird abgeschnitten.
(&00): max. zulaessige Satzlaenge.
Fehlerschalter: wird nicht gesetzt.

EDT2301 COPY BUFFER OVERFLOW
EDT2301 KOPIERPUFFER VOLL

Bedeutung
Der Kopierpuffer kann nur 256 Zeilennummern aufnehmen.
Fehlerschalter: wird nicht gesetzt.

EDT2900 A KEY WAS ZERO AND HAS BEEN SET TO 0.0001
EDT2900 EIN SCHLUESSEL WAR NULL UND WURDE AUF 0.0001 GESETZT

Bedeutung
Ein Schluessel mit dem Wert 0 wurde waehrend einer @GET- oder @READ-Anweisung (mit KEY-Funktion) entdeckt. Der Schluessel wird vom EDT auf 0.0001 gesetzt.
Fehlerschalter: EDT.

EDT2901 DATA LOSS DUE TO TABULATOR FUNCTION. CHECK LINE LENGTH
EDT2901 DATENVERLUST DURCH TABULATOR-FUNKTION. ZEILENLAENGE PRUEFEN

Bedeutung
Text geht aufgrund der Tabulatordefinition verloren.
Fehlerschalter: EDT.

EDT2902 CHECK TAB COLUMNS
EDT2902 TABULATORSPALTEN PRUEFEN

Bedeutung
In der @TABS-Anweisung wurde die CHECK-Funktion angegeben. So wurde bemerkt, dass die gerade mit Tabellierzeichen eingegebene Zeile eine Rueckwaertstabellierung bewirkt, d.h. dass die angelegte Textzeile ueberschrieben wurde.
Fehlerschalter: EDT.

Maßnahme
Zeile pruefen, da wahrscheinlich nicht korrekt.

EDT2903 FILE IS EMPTY
EDT2903 DATEI IST LEER

Bedeutung
Die in der Anweisung angegebene Datei ist leer. Diese Meldung wird unter einer der folgenden Bedingungen ausgegeben:
- Auf eine leere Datei auf der Platte wird mit einer @READ-, @GET-, @INPUT- oder @ELIM-Anweisung zugegriffen.
- Die Arbeitsdatei ist leer, und eine @SAVE-, @WRITE-, @XWRITE- oder @SETLIST-Anweisung wird gegeben bzw. eine bei @COMPARE angegebene Prozedurdatei ist leer.

Fehlerschalter: EDT.

EDT2904 MAXIMUM LINE NUMBER WHEN PROCESSING '@RENUMBER'. SOME LINES ARE LOST
EDT2904 MAX. ZEILENNUMMER BEI @RENUMBER-ANWEISUNG. ZEILEN VERLOREN

Bedeutung
Die max. Zeilennummer (9999.9999) wurde waehrend der @RENUMBER-Anweisung ueberschritten. Der EDT kann keine doppelten Zeilennummern in derselben Arbeitsdatei zulassen. Deshalb wird der Rest der Daten geloescht.
Fehlerschalter: EDT.

EDT3002 OPERAND ERROR
EDT3002 OPERANDEN-FEHLER

Bedeutung
EDT meldet, dass in der Anweisung entweder ein Operand falsch angegeben wurde oder ein syntaktischer Fehler vorliegt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3003 '(' MISSING
EDT3003 '(' FEHLT

Maßnahme
Fehlende Klammer einfuegen und Anweisung wiederholen.

EDT3004 ')' MISSING
EDT3004 ')' FEHLT

Maßnahme
Fehlende Klammer einfuegen und Anweisung wiederholen.

EDT3040 INVALID NAME OR NAME MISSING
EDT3040 UNGUELTIGER NAME ODER NAME FEHLT
(B) Routing code: * Weight: 99

Bedeutung
Die Zeichenfolge ist laenger als 8 Zeichen, entspricht nicht den syntaktischen Anforderungen des Operanden oder fehlt ganz.
Fehlerschalter: EDT.

EDT3050 INVALID SYSLST-NUMMER
EDT3050 UNGUELTIGE SYSLST-NUMMER
(B) Routing code: * Weight: 99

Bedeutung
Die in der @LOG-Anweisung angegebene SYSLST-Nummer ist nicht gueltig. Sie muss zwischen 1 und 99 liegen.
Fehlerschalter: EDT.

EDT3065 NUMBER OF LINES OR 'OFF' OR 'O' EXPECTED
EDT3065 ZEILENANZAHL ODER 'OFF' BZW. 'O' ERWARTET

Bedeutung
Bei dem Operanden SPLIT in der @PAR-Anweisung muss angegeben werden:
- Entweder die Zeilenanzahl des zweiten Fensters und die Arbeitsdatei, die im zweiten Fenster gezeigt werden soll, oder
- OFF bzw. O, um die Ausgabe auf ein Fenster zurueckzusetzen.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3066 WRONG COLUMN NUMBER
EDT3066 FALSCHE SPALTENANGABE

Bedeutung
Die Spaltenangabe in der angegebenen @SETF-Anweisung ist falsch.
Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3067 UPPER RANGE LIMIT INVALID OR MISSING
EDT3067 OBERGRENZE DES SATZBEREICHES FEHLT ODER IST UNGUELTIG

Bedeutung
In der COPY- oder DELETE-Anweisung fehlt die Angabe der Obergrenze des Satzbereiches, oder sie ist nicht korrekt. Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3068 POSITION INVALID OR MISSING
EDT3068 POSITION FEHLT ODER IST UNZULAESSIG

Bedeutung
Die Angabe der Position in der @SETF-Anweisung ist unbedingt erforderlich.
Die Anweisung konnte nicht ausgefuehrt werden.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3069 STRING TO BE INSERTED IS MISSING
EDT3069 EINZUFUEGENDE ZEICHENKETTE FEHLT

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, da die Angabe der einzufuegenden Zeichenkette fehlt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3070 'EDIT-LONG' EXPECTED
EDT3070 'EDIT-LONG' ERWARTET

Bedeutung
Der Operand in der @PAR-Anweisung ist falsch.
Korrekte Form des Operanden: @PAR EDIT-LONG = ...

Maßnahme
Korrigierte Anweisung eingeben.

EDT3071 'ON', 'OFF' OR 'O' EXPECTED
EDT3071 'ON', 'OFF' ODER 'O' ERWARTET

Bedeutung
EDT erwartet in einer Anweisung ein ON, OFF oder O als Operand.
Moegliche Ursache:
- Die Angabe des Operanden der @PAR-Anweisung ist fehlerhaft, nach dem Gleichheitszeichen fehlt ein ON, OFF oder O.
- Es wurde versucht, in der @TABS-Anweisung mehr als 8 Positionen zu definieren. Es ist aber nur mehr der Operand ON, OFF oder O zulaessig.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3072 NUMBER INVALID OR MISSING
EDT3072 NUMMER FEHLT ODER IST UNZULAESSIG

Bedeutung
Ein numerischer Wert wurde in einem falschen Format angegeben, oder der Wert wurde ueberhaupt nicht angegeben. Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3073 TARGET POSITION IS INVALID OR MISSING
EDT3073 ZIELPOSITION FEHLT ODER IST NICHT RICHTIG

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, da in der COPY- bzw. INSERT-Anweisung die Angabe der Zielposition fehlt oder fehlerhaft ist.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3074 'KEEP' OPERAND EXPECTED IN 'COPY' STATEMENT
EDT3074 'COPY'-ANWEISUNG MIT OPERAND 'KEEP' ERWARTET

Bedeutung
Die COPY-Anweisung wurde nicht ausgefuehrt, da der Operand KEEP fehlt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3075 RECORD RANGE CANNOT BE SPECIFIED
EDT3075 SATZBEREICHS-ANGABE NICHT MOEGLICH

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, da in der Anweisung kein Satzbereich angegeben werden kann.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3076 'COPY KEEP' PERMISSIBLE ONLY FOR ISAM FILES
EDT3076 'COPY KEEP' NUR FUER ISAM-DATEIEN ZULAESSIG

Bedeutung
Der Operand KEEP in der COPY-Anweisung ist nur fuer ISAM-Dateien zulaessig. Die Anweisung wurde nicht ausgefuehrt.

EDT3077 OPERAND 'STRUCTURE=' INCORRECT
EDT3077 OPERAND 'STRUCTURE=' FEHLERHAFT

Bedeutung
In der @PAR-Anweisung fehlt das Symbol fuer STRUCTURE, oder es wurde nicht in Hochkommata angegeben.

Maßnahme
Korrigierte Anweisung wiederholen.

```
EDT3078    SPECIFIED NUMBER INVALID (VALID RANGE: 1..256)
EDT3078    FEHLERHAFTER ZAHLWERT (ZULAESSIGER BEREICH: 1..256)
```

Bedeutung
Der Wert fuer die LIMIT-Angabe in der Anweisung @PAR oder der Wiederholungsfaktor n der #-Anweisung liegt nicht im zulaessigen Wertebereich.

Maßnahme
Korrigierte Anweisung wiederholen.

```
EDT3079    COLUMN '0' NOT PERMISSIBLE
EDT3079    SPALTEN-NUMMER '0' NICHT ERLAUBT
```

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, da '0' nicht als Spaltennummer erlaubt ist.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3080    SPECIFIED COLUMN INVALID, OR ':' MISSING IN COLUMN RANGE
EDT3080    FALSCHE SPALTENANGABE ODER ':' IM SPALTENBEREICH FEHLT
```

Bedeutung
In der Anweisung fehlt entweder das Zeichen ':' innerhalb der Spaltenbereichsangabe, oder der angegebene Spaltenbereich ist ungueltig. Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3081    LINE NUMBER > 9999.9999
EDT3081    ZEILENNUMMER > 9999.9999
```

Bedeutung
Die angegebene Zeilennummer ist zu gross. Die maximal erlaubte Zeilennummer ist 9999.9999.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3082    LINE NUMBER 0 INVALID
EDT3082    ZEILENNUMMER 0 UNZULAESSIG
```

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, weil die Zeilennummer 0 unzulaessig ist.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3085 '(&00)' NOT POSSIBLE FOR PLAM ELEMENT TYPE '(&01)'
EDT3085 '(&00)' NICHT MOEGLICH FUER PLAM ELEMENT TYP '(&01)'

Bedeutung
Es ist nicht moeglich, PLAM-Bibliothekselemente des Typs (&01) mit der Anweisung (&00) zu bearbeiten.
z.B. (&01): R, C, H, L, U, F oder entsprechender freier Typname
und (&00): @COPY, @OPEN, @WRITE oder @INPUT jeweils Format 2.

EDT3086 INVALID PLAM TYPE
EDT3086 UNZULAESSIGER PLAM-TYP

Bedeutung
In der Anweisung wurde ein unzulaessiger PLAM-Typ angegeben.
Zulaessig: S,M,J,P,D,X,R,C,H,L,U,F oder entsprechender freier Typname.
Ein freier Typname darf nicht mit $ oder SYS beginnen und besteht aus 2 bis 8 Zeichen.

EDT3087 INVALID JOB VARIABLE NAME
EDT3087 UNGUELTIGER JVNAME

Bedeutung
Die Zeichenfolge, die zur Angabe eines JV Namens angegeben wurde, entspricht nicht der Syntax fuer einen JV Namen, oder der JV Name in einer @SETJV- oder @GETJV-Anweisung war nicht vollqualifiziert, oder die @ERAJV-Anweisung war unzulaessig.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3088 INVALID NAME OF S-VARIABLE
EDT3088 UNGUELTIGER NAME FUER S-VARIABLE
(B) Routing code: * Weight: 99

Bedeutung
Die Zeichenfolge, die zur Angabe eines SDF-P-Variablennamens in einer @GETVAR- oder @SETVAR-Anweisung angegeben wurde, entspricht nicht der Syntax fuer eine SDF-P-Variable.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3089    INVALID NAME OF UFS FILE
EDT3089    UNGUELTIGER NAME EINER UFS DATEI
```
(B) Routing code: * Weight: 99

Bedeutung
Die Zeichenfolge, die zur Angabe eines POSIX-Dateinamens eingegeben wurde, entspricht nicht der Syntax fuer einen gueltigen Namen einer Datei im POSIX-Dateisystem oder ein Unterverzeichnis, das angegeben wurde, ist nicht vorhanden.
Fehlerschalter: EDT.

```
EDT3101    INVALID STATEMENT
EDT3101    UNZULAESSIGE ANWEISUNG
```

Bedeutung
Das erste Zeichen der Anweisung wurde von EDT als ungueltig erkannt. Die Anweisung wurde nicht ausgefuehrt.

```
EDT3106    SPECIFIED WORK FILE INVALID. (VALID RANGE: 0..9)
EDT3106    FALSCHE ANGABE DER ARBEITSDATEI. (ZULAESSIGER BEREICH: 0..9)
```

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3110    EQUATION MARK EXPECTED. STATEMENT NOT PROCESSED
EDT3110    GLEICHHEITSZEICHEN ERWARTET. ANWEISUNG NICHT AUSGEFUEHRT
```

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3111    'TYPE' OPERAND IS MISSING
EDT3111    'TYPE'-OPERAND FEHLT
```

Bedeutung
In der OPEN- oder WRITE-Anweisung wurde der Operand TYPE ohne gueltigen Operandenwert angegeben.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3112 'TYPE' OPERAND ALREADY DEFINED
EDT3112 'TYPE'-OPERAND BEREITS DEFINIERT

Bedeutung
In der OPEN- oder WRITE-Anweisung wurde versucht, den TYPE-Operanden nochmals zu definieren.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3116 'CODE' OPERAND MISSING OR INVALID
EDT3116 'CODE'-OPERAND FEHLT ODER IST UNGUELTIG
(B) Routing code: * Weight: 99

Bedeutung
In der @XOPEN-, @XWRITE- oder @XCOPY-Anweisung wurde der Operand CODE ohne gueltigen Operandwert angegeben.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3117 'MODE' OPERAND MISSING OR INVALID
EDT3117 'MODE'-OPERAND FEHLT ODER IST UNGUELTIG

Bedeutung
In einer der Anweisungen @OPEN (Format 2), @WRITE (Format 2), @XOPEN, @XWRITE oder @SETVAR wurde der Operand MODE ohne gueltigen Operandenwert angegeben.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3118 'MODE' OPERAND ALREADY DEFINED
EDT3118 'MODE'-OPERAND BEREITS DEFINIERT

Bedeutung
In der OPEN- oder WRITE-Anweisung wurde versucht, den MODE-Operanden nochmals zu definieren.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3119 WORK FILE ALREADY DEFINED
EDT3119 ARBEITSDATEI BEREITS DEFINIERT

Bedeutung
In der OPEN- oder WRITE- Anweisung wurde versucht, die Arbeitsdatei nochmals zu definieren.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3120    FILE NAME ALREADY DEFINED
EDT3120    DATEINAME WURDE SCHON VEREINBART
```

Bedeutung
In der OPEN- oder WRITE-Anweisung wurde versucht, den Dateinamen nochmals zu definieren.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3121    LIBRARY NAME MISSING OR FORMAT OF SPECIFIED LIBRARY NAME INVALID
EDT3121    BIBLIOTHEKSNAME FEHLT ODER FORMAT DER ANGABE IST FALSCH
```

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3122    FILE NAME MISSING OR FORMAT OF SPECIFIED FILE NAME INVALID
EDT3122    DATEINAME FEHLT ODER FORMAT DER ANGABE IST FALSCH
```

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3123    NO VALID NAME OF PLAM MEMBER
EDT3123    KEIN GUELTIGER PLAM-ELEMENTNAME
```

Bedeutung
Bei der Bearbeitung einer PLAM-Bibliothek wurde kein gueltiger Elementname angegeben.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3124    VERSION NUMBER MISSING OR INVALID
EDT3124    VERSIONSNUMMER FEHLT ODER IST UNGUELTIG
```

Bedeutung
Die Versionsnummer eines PLAM-Bibliothekselements fehlt oder enthaelt ungueltige Zeichen.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3125    'OPEN REAL' PERMISSIBLE ONLY FOR ISAM FILES
EDT3125    'OPEN REAL' NUR FUER ISAM-DATEIEN ZULAESSIG
```

Bedeutung
Es ist nicht moeglich, die angegebene Datei mit OPEN REAL zu bearbeiten, da dieser Modus nur fuer ISAM-Dateien zulaessig ist.

Maßnahme
Datei virtuell bearbeiten.

EDT3126 FILE ATTRIBUTES CANNOT BE SPECIFIED
EDT3126 KEINE VERGABE VON DATEI-ATTRIBUTEN MOEGLICH

Bedeutung
Da die Vergabe von Dateiattributen in dieser Anweisung nicht moeglich ist, wurde die Anweisung nicht ausgefuehrt.

EDT3127 NAME OF WORK FILE IS INVALID OR MISSING
EDT3127 BEZEICHNUNG DER ARBEITSDATEI FEHLT ODER IST UNGUELTIG

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, da fuer die Arbeitsdatei kein Name angegeben wurde oder der angegebene Name unzulaessig ist.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3128 PLAM LIBRARY NAME INVALID OR MISSING
EDT3128 PLAM-BIBLIOTHEKSNAME FEHLT ODER IST UNGUELTIG

Maßnahme
Korrigierte Anweisung eingeben.

EDT3129 NO FILE ATTRIBUTES CAN BE DEFINED FOR PLAM LIBRARIES
EDT3129 VERGABE VON DATEIATTRIBUTEN FUER PLAM-BIBLIOTHEKEN NICHT MOEGLICH

Bedeutung
Bei der Bearbeitung einer PLAM-Bibliothek wurde versucht, das Dateiattribut FCBTYPE=ISAM oder SAM zu vergeben. Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3132 PLAM TYPE IS MISSING OR INVALID
EDT3132 PLAM-TYPE FEHLT ODER IST UNGUELTIG

Bedeutung
Das TYPE-Attribut des PLAM-Elementes wurde in der Anweisung nicht oder nicht richtig angegeben.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3133 NUMBER OF PLAM VERSION INVALID
EDT3133 NUMMER DER PLAM-VERSION IST UNGUELTIG

Maßnahme
Anweisung mit gueltiger Versionsnummer wiederholen.

EDT3134 '*STD' EXPECTED
EDT3134 '*STD' ERWARTET

Bedeutung
Bei der Bearbeitung einer PLAM-Bibliothek wurde fuer Typ oder Version ein '*' vergeben.

Maßnahme
'*' durch '*STD' ersetzen und Anweisung wiederholen.

EDT3135 MODUL NAME MISSING
EDT3135 MODULNAME FEHLT

Bedeutung
In der @UNLOAD-Anweisung wurde kein Modulname angegeben.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3136 'INCREMENT=0' NOT PERMISSIBLE
EDT3136 'INCREMENT=0' NICHT ERLAUBT

Bedeutung
Die Angabe INCREMENT=0 in der @PAR-Anweisung ist nicht erlaubt. Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3138 ONLY ONE CHARACTER POSSIBLE AS SYMBOL
EDT3138 NUR EIN ZEICHEN ALS SYMBOL MOEGLICH

Bedeutung
In einer Anweisung wurde fuer ein Symbol mehr als ein Zeichen angegeben:
- fuer das Strukturzeichen oder das Separatorsymbol in @PAR, oder
- fuer das ASTERISK-, SLASH- oder FILLER-Symbol in @SYMBOLS.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3170 SYNTAX ERROR IN LINE NUMBER
EDT3170 SYNTAX-FEHLER IN ZEILENNUMMER
(B) Routing code: * Weight: 99

Bedeutung
Der Operand entspricht nicht dem Syntax fuer eine Zeilennummer.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3171 NO EDT V15 OR EDT V16.0 STATEMENTS IN 'CONTROL' MODE
EDT3171 KEINE EDT-V15- ODER EDT-V16.0-ANWEISUNGEN IM 'CONTROL'-MODUS

Bedeutung
Es ist nicht moeglich, Anweisungen des EDT der Versionen V15 oder V16.0 im CONTROL-Modus anzugeben.

EDT3172 MODULE NAME TOO LONG
EDT3172 MODULNAME IST ZU LANG

Bedeutung
Der in der @UNLOAD-Anweisung angegebene Modulname ist laenger als 8 Zeichen.
Fehlerschalter: EDT.

EDT3173 NUMBER OF WORK FILE FOR COMPARE OPERATION MISSING OR INVALID
EDT3173 NUMMER DER ARBEITSDATEI FUER VERGLEICH FEHLT ODER IST UNGUELTIG

Bedeutung
Fehlerschalter: EDT.

EDT3174 NAME TOO LONG
EDT3174 NAME ZU LANG

Bedeutung
Die Zeichenfolge, die zur Angabe eines Datei- oder Jobvariablen-Namens verwendet wurde, besteht aus mehr als 54 Zeichen.
Fehlerschalter: EDT.

EDT3175 SYNTAX ERROR IN SPECIFIED RANGE
EDT3175 SYNTAX-FEHLER IN ZEILENBEREICHSANGABE

EDT3176 STATEMENT SYMBOL INVALID OR TOO LONG
EDT3176 ANWEISUNGSSYMBOL UNGUELTIG ODER ZU LANG

Bedeutung
Das Anweisungssymbol in der @USE-Anweisung muss in Hochkommata angegeben werden und darf nur 1 Zeichen lang sein.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3177 ENTRY NAME TOO LONG
EDT3177 ENTRY-NAME ZU LANG

Bedeutung
Der ENTRY-Name darf maximal 8 Zeichen lang sein.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3178 LIBRARY NAME TOO LONG
EDT3178 BIBLIOTHEKSNAME ZU LANG

Bedeutung
Der Bibliotheksname darf maximal 54 Zeichen lang sein.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3179 ENTRY NAME MISSING
EDT3179 KEIN ENTRY-NAME ANGEGEBEN

Bedeutung
Bei Verwendung der externen Anweisungsroutinen als Anweisungsfilter muss in der @USE-Anweisung ein konstanter ENTRY-Name angegeben werden.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3180 JOKER SYMBOL EQUALS QUOTE
EDT3180 JOKER-SYMBOL ENTSPRICHT QUOTE-ZEICHEN

Bedeutung
Moegliche Ursachen:
- In einer @SYMBOLS-Anweisung wurde ein Wert fuer das ASTERISK- oder das SLASH-Zeichen angegeben, der nicht gueltig ist, da er mit einem der QUOTE-Zeichen identisch ist.
- Eine @ON-Anweisung mit dem Schluesselwort PATTERN kann nicht ausgefuehrt werden, da eines der QUOTE-Zeichen mit dem ASTERISK- oder dem SLASH-Zeichen identisch ist.

Fehlerschalter: EDT.

Maßnahme
Fuer ASTERISK-, SLASH-, QUOTE1- und QUOTE2-Zeichen verschiedene Symbole angeben.

EDT3181 BOTH JOKER SYMBOLS ARE THE SAME
EDT3181 BEIDE JOKER-SYMBOLE IDENTISCH

Bedeutung
In einer @SYMBOLS-Anweisung sollte ein Jokersymbol umdefiniert werden. Die Anweisung wurde abgewiesen, da fuer ASTERISK und SLASH verschiedene Werte definiert sein muessen.
Fehlerschalter: EDT.

Maßnahme
@SYMBOLS-Anweisung mit verschiedenen Werten fuer ASTERISK und SLASH wiederholen.

EDT3182 CCSN TOO LONG
EDT3182 CCSN ZU LANG

Bedeutung
Die Zeichenfolge, die zur Angabe eines Coded-Character-Set-Namens verwendet wurde, besteht aus mehr als 8 Zeichen.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3183 LINE NUMBER EXPECTED
EDT3183 ZEILENNUMMER ERWARTET

Bedeutung
Nach dem Schluesselwort TO in der Anweisung @FSTAT, @STAJV,.. muss eine gueltige Zeilennummer angegeben werden.

EDT3901 ILLEGAL BINARY CONSTANT
EDT3901 UNZULAESSIGE BINAER-KONSTANTE

Bedeutung
Eine Zeichenfolge mit einem 'B' vor dem ersten Hochkomma enthaelt einen Fehler. Die Zeichen muessen die Ziffern 0 oder 1 sein, und die Zeichenfolge darf nicht leer sein.
Fehlerschalter: EDT.

EDT3902 ILLEGAL HEX CONSTANT
EDT3902 UNGUELTIGE HEX-KONSTANTE

Bedeutung
Eine Zeichenfolge mit einem 'X' vor dem 1. Hochkomma enthaelt einen Fehler. Die Zeichen muessen die Ziffern 0 bis 9 oder die Buchstaben A bis F sein, und die Zeichenfolge darf nicht leer sein.
Fehlerschalter: EDT.

EDT3903 INVALID RANGE
EDT3903 UNGUELTIGER BEREICH

Bedeutung
Die Zeilennummern in einer Bereichsangabe sind fehlerhaft, oder nach einem Bindestrich (-) folgt keine zweite Zeilennummer.
Fehlerschalter: EDT.

EDT3904 INVALID SUBSTRING
EDT3904 UNGUELTIGE TEILZEICHENFOLGE

Bedeutung
Eine Teilzeichenfolge in einer @SET-Anweisung ist fehlerhaft.
Sie sollte dem Syntax ln oder +/-int entsprechen.
Fehlerschalter: EDT.

EDT3905 INVALID VARIABLE
EDT3905 UNGUELTIGE VARIABLE

Bedeutung
Eine Zeilennummer-, Zeichenfolge- oder Ganzzahlvariable wurde fehlerhaft angegeben.
Fehlerschalter: EDT.

EDT3906 LINE NUMBER INVALID
EDT3906 UNGUELTIGE ZEILENNUMMER

Bedeutung
Der Wert einer Zeilennummer ist ungueltig und zwar entweder
- in der Ganzzahl der Anweisung @SET ln-var=int-var oder
- in der ersten Zeilennummern-Variable der Anweisung @SET ln-var,cl=... oder
- in der Zielangabe der Anweisung @GETJV.

Ferner kann eine Zeilennummer fuer die angegebene KEYLEN einer durch @OPEN eroeffneten Datei zu gross sein.
Fehlerschalter: EDT.

EDT3907 EMPTY STRING NOT PERMISSIBLE
EDT3907 LEERE ZEICHENKETTE NICHT ERLAUBT

Bedeutung
Die Zeichenkette, die in einer Anweisung direkt oder indirekt (z.B. in einer EDT-Zeichenfolgevariablen oder in einer SDF-P-Variablen) angegeben wurde, ist leer. Sie ist aber an dieser Stelle nicht erlaubt.
Fehlerschalter: EDT.

EDT3908 STRING MISSING OR INVALID
EDT3908 ZEICHENFOLGE FEHLT ODER IST UNGUELTIG

Bedeutung
Eine Zeichenfolge fehlt in einer Anweisung oder ist ungueltig.
Die haeufigsten Faelle sind:
- eine Zeichenfolge fehlt in einer Anweisung, die einen wahlfreien Dateinamen (file) enthaelt, und es ist keine @FILE-Anweisung wirksam
- ein Hochkomma fehlt fuer eine Zeichenfolge, die zwei benoetigt.

Fehlerschalter: EDT.

EDT3909 @PARAMETER ERROR
EDT3909 @PARAMETER-FEHLER

Bedeutung
Einige der haeufigsten Fehlerursachen sind:
- die Zeilennummer (ln) oder das Inkrement (inc) ist ungueltig
- Operand(en) fehlt/fehlen in einer Anweisung
- die Nummer einer Prozedurdatei ist '0'
- ein ungueltiges ON/OFF
- der Wert in der @SETSW-Anweisung ist >31.

Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3910 DUPLICATE FORMAL OPERAND
EDT3910 FORMALER OPERAND ZU OFT ANGEGEBEN
(B) Routing code: * Weight: 99

Bedeutung
In der @PARAMS-Anweisung wurde ein formaler Operand (&id) mindestens zweimal angegeben.
Fehlerschalter: EDT.

EDT3911 DUPLICATE KEYWORD
EDT3911 SCHLUESSELWORT MEHRFACH ANGEGEBEN
(B) Routing code: * Weight: 99

Bedeutung
In einer @DO-Anweisung wurde ein Schluesselwort mindestens zweimal angegeben.
Fehlerschalter: EDT.

EDT3922 INVALID COLUMN (RANGE)
EDT3922 SPALTEN-(BEREICH) UNGUELTIG
(B) Routing code: * Weight: 99

Bedeutung
Der fuer eine Spalte angegebene Wert ist ungueltig, oder die Angabe der Spalte bzw. des Spaltenbereichs ist syntaktisch fehlerhaft.
Fehlerschalter: EDT.

EDT3951 PROCEDURE NUMBER > 22
EDT3951 PROZEDUR-NUMMER > 22

Bedeutung
Fehlerschalter: EDT.

EDT3952 INVALID SYMBOL
EDT3952 UNGUELTIGES SYMBOL
(B) Routing code: * Weight: 99

Bedeutung
Fuer ein Symbol, das ein Sonderzeichen sein muss, wurde ein anderes Zeichen eingegeben:
- zur Vereinbarung eines neuen Anweisungssymbols ("@:"), oder
- fuer das Zeilenbereichssymbol in @RANGE, oder
- fuer das 1.Begrenzersymbol ("Hochkomma") in @QUOTE, oder
- fuer die Jokersymbole in @SYMBOLS, oder
- fuer das Schleifensymbol in @DO.

Maßnahme
Ein gueltiges Sonderzeichen als Symbol angeben.

EDT3991 SYNTAX ERROR IN EXTERNAL STATEMENT
EDT3991 SYNTAX-FEHLER IN EXTERNER ANWEISUNG

Bedeutung
Die externe Routine meldet einen Syntaxfehler in der angegebenen Anweisung.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT3999 (&00)
EDT3999 (&00)

Bedeutung
Syntaxfehler in externer Anweisung.
(&00): Meldung von externer Routine.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT4100 (&00)
EDT4100 (&00)

Bedeutung
EDT-Meldung der Version 16.0.

```
EDT4200    '(&00)': DMS ERROR CODE: '(&01)'
EDT4200    '(&00)': DVS-FEHLERCODE: '(&01)'
```

Bedeutung

Alle DVS-Fehler werden in dieser Form ausgedruckt, wobei gilt:
(&00): DVS-Makro (OPEN,etc.) bei dessen Ausfuehrung der Fehler auftritt
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den DVS-Fehler kann mit dem ISP-Kommando /HELP DMS(&01) oder dem SDF-Kommando /HELP-MESS DMS(&01) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. einem der BS2000 DVS-Handbuecher entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

```
EDT4201    '(&00)': JVS ERROR CODE: '(&01)'
EDT4201    '(&00)': JVS-FEHLERCODE: '(&01)'
```

Bedeutung

Alle JVS-Fehler werden in dieser Form ausgedruckt, wobei gilt:
(&00): JVS-Makro (STAJV, etc.), bei dessen Ausfuehrung der Fehler auftritt
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den JVS-Fehler kann mit dem ISP-Kommando /HELP JVS(&01) oder dem SDF-Kommando /HELP-MESS JVS(&01) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. dem BS2000 JVS-Handbuch entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

```
EDT4202    '(&00)': SDF-P ERROR CODE: '(&01)'
EDT4202    '(&00)': SDF-P-FEHLERCODE: '(&01)'
```

Bedeutung

Alle SDF-P-Fehler werden in dieser Form ausgedruckt, wobei gilt:
(&00): SDF-P-Makro (PUTVAR, etc.), bei dessen Ausfuehrung der Fehler auftritt.
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den SDF-P-Fehler kann mit dem ISP-Kommando /HELP SDP(&01) oder dem SDF-Kommando /HELP-MESS SDP(&00) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. dem BS2000 SDF-P-Handbuch entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

EDT4203 '(&00)': XHCS ERROR CODE: '(&01)'
EDT4203 '(&00)': XHCS-FEHLERCODE: '(&01)'

Bedeutung
Alle XHCS-Fehler werden in dieser Form ausgedruckt, wobei gilt:
(&00): XHCS-Makro (NLSCODE, etc.), bei dessen Ausfuehrung der Fehler auftritt
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den XHCS-Fehler kann mit dem ISP-Kommando /HELP XHC(&01) oder dem SDF-Kommando /HELP-MESS XHC(&01) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. dem BS2000 XHCS-Handbuch entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

EDT4204 '(&00)': TIAM ERROR CODE: '(&01)'
EDT4204 '(&00)': TIAM-FEHLERCODE: '(&01)'
(B) Routing code: * Weight: 99

Bedeutung
Alle TIAM-Fehler werden in dieser Form ausgedruckt, wobei gilt:
(&00): TIAM-Makro (WRLST, etc.), bei dessen Ausfuehrung der Fehler auftritt.
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den TIAM-Fehler kann dem BS2000-Handbuch 'TIAM' bzw. dem BS2000-Handbuch 'Makroaufrufe an den Ablaufteil' entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

EDT4205 '(&00)': BLS ERROR CODE: '(&01)'
EDT4205 '(&00)': BLS-FEHLER-CODE: '(&01)'
(B) Routing code: * Weight: 99

Bedeutung
Alle Fehler des Binder-Lader-Systems werden in dieser Form ausgedruckt, wobei gilt:
(&00): BLS-Makro (BIND), bei dessen Ausfuehrung der Fehler auftritt.
(&00): Fehlercode in hexadezimaler Form.
Naehere Information ueber den BLS-Fehler kann dem BS2000-Handbuch 'Binder und Lader' bzw. dem BS2000-Handbuch 'Makroaufrufe an den Ablaufteil' werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

EDT4206 POSIX-CALL '(&00)': ERROR '(&01)'
EDT4206 POSIX-FUNKTION '(&00)': FEHLER '(&01)'
(B) Routing code: * Weight: 99

Bedeutung
Alle Fehler bei Aufrufen von C-Funktionen werden in dieser Form.
ausgedruckt.
(&00): Name der POSIX-Bibliotheksfunktion, bei deren Ausfuehrung der Fehler auftrat.
(&01): Fehler, der in der C-Variablen ERRNO gemeldet wird.
Naehere Information ueber den Fehler kann dem BS2000-Handbuch 'C-Bibliotheksfunktionen' bzw. dem BS2000-Handbuch 'POSIX' entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

EDT4207 '(&00)': SDF ERROR CODE: '(&01)'
EDT4207 '(&00)': SDF-FEHLERCODE: '(&01)'
(B) Routing code: * Weight: 99

Bedeutung
Alle Fehler von SDF-Makros werden in dieser Form ausgedruckt, wobei gilt:
(&00): SDF-Makro (CMDSTA, etc.), bei dessen Ausfuehrung der Fehler auftritt.
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den Fehler kann dem BS2000-Handbuch 'SDF-A' entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

EDT4300 ERROR AT SYSTEM COMMAND: ERROR CODE '(&00)'
EDT4300 FEHLER BEI SYSTEMKOMMANDO: FEHLERCODE '(&00)'
(B) Routing code: * Weight: 99

Bedeutung
Bei Aufruf eines Systemkommandos mittels @SYSTEM-Anweisunglieferte der CMD-Makro denFehlercode X'10' oder X'14'.
Naehere Information ueber die Fehlerursache kann mit dem ISP-Kommando /HELP (&00) oder dem SDF-Kommando /HELP-MESS (&00) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' entnommen werden.
Fehlerschalter: DVS.

```
EDT4310    SDF: SYNTAX ERROR IN LINE (&00)
EDT4310    SDF: SYNTAX FEHLER IN ZEILE (&00)
```
(B) Routing code: * Weight: 99

Bedeutung
Bei der Pruefung von Datenzeilen mittels @SDFTEST wurde ein Syntaxfehler in Zeile (&00) entdeckt und konnte im SDF-Fehlerdialog nicht korrigiert werden.
Fehlerschalter: EDT.

Maßnahme
SDF-Fehlerdialog ermoeglichen, indem z.B. mittels @SYSTEM-Anweisung das Kommando /MODIFY-SDF-OPTIONS GUIDANCE=MIN eingegeben wird.

```
EDT4900    /SET-FILE-LINK IS IN EFFECT
EDT4900    /SET-FILE-LINK IST AKTIV
```

Bedeutung
Ein /SET-FILE-LINK mit einem von EDT verwendeten Dateikettungsnamen (EDTSAM, EDTISAM, EDTMAIN) ist aktiv. Der Dateiname in der EDT-Anweisung (@GET, @READ, @INPUT, @OPEN, @ELIM, @WRITE oder @SAVE) stimmt jedoch nicht mit dem im /SET-FILE-LINK-Kommando ueberein.
Die Anweisung wird nicht ausgefuehrt.
Fehlerschalter: EDT.

```
EDT4901    ONE INPUT FILE IS ALREADY ACTIVE
EDT4901    EINE INPUT-DATEI IST BEREITS AKTIVIERT
```

Bedeutung
2 aktive @INPUT-Anweisungen sind im EDT nicht erlaubt.
Fehlerschalter: EDT.

```
EDT4903    BOTH OPERANDS IN '@QUOTE' STATEMENT ARE THE SAME
EDT4903    BEIDE OPERANDEN IN @QUOTE-ANWEISUNG IDENTISCH
```

Bedeutung
Werden in einer @QUOTE-Anweisung beide Operanden (q1 und q2) angegeben, muessen sie verschieden sein.
Fehlerschalter: EDT.

```
EDT4904    BTAM FILES NOT SUPPORTED
EDT4904    BTAM DATEIEN NICHT UNTERSTUETZT
```

Bedeutung
Es wurde versucht, mit einer der Anweisungen @GET, @SAVE, @READ, @WRITE, @INPUT, @OPEN oder @ELIM eine BTAM-Datei zu bearbeiten.
BTAM-Dateien unterstuetzt der EDT nicht.
Fehlerschalter: EDT.

EDT4906 '(&00)' NOT POSSIBLE FOR CURRENT PROCEDURE FILE
EDT4906 '(&00)' FUER AKTUELLE PROZEDURDATEI NICHT MOEGLICH

Bedeutung
Die Anweisung (&00) bezieht sich auf die aktuelle Prozedurdatei und kann deshalb nicht ausgefuehrt werden (z.B. @DO, @DROP).
Fehlerschalter: EDT.

EDT4907 '@DROP' NOT POSSIBLE DURING PROCEDURE FILE PROCESSING
EDT4907 '@DROP' WAEHREND AUSFUEHRUNG EINER PROZEDURDATEI NICHT MOEGLICH
(B) Routing code: * Weight: 99

Bedeutung
Fehlerschalter: EDT.

EDT4908 INVALID COMMAND
EDT4908 UNGUELTIGES KOMMANDO

Bedeutung
Das in der @SYSTEM-Anweisung angegebene Kommando ist fehlerhaft oder kann nicht ueber den CMD-Makro abgegeben werden.
Fehlerschalter: DVS.

Maßnahme
Korrigiertes Kommando wiederholen oder mit @SY ins System verzweigen.

EDT4909 PROCEDURE FILE ALREADY ACTIVE
EDT4909 PROZEDURDATEI BEREITS AKTIV

Bedeutung
Eine @PROC-Anweisung wurde auf die aktuelle Prozedurdatei gegeben.
Fehlerschalter: EDT.

EDT4910 S-VARIABLE MUST BE OF TYPE LIST
EDT4910 S-VARIABLE NICHT VOM TYP LIST
(B) Routing code: * Weight: 99

Bedeutung
Die SDF-P-Variable, die in einer @LOG-, @GETLIST- or @SETLIST-Anweisung angegeben wurde, ist nicht vom Typ LIST oder wurde noch nicht deklariert.
Fehlerschalter: EDT.

Maßnahme
Falls die Variable noch nicht deklariert wurde, setzen Sie das Systemkommando /DECL-VAR NAME=..,MULT-ELEM=LIST ab, bevor Sie die Anweisung wiederholen.

EDT4912 EAM OPEN ERROR
EDT4912 EAM-DATEI KANN NICHT GEOEFFNET WERDEN

Bedeutung
Eine EAM-Datei kann waehrend einer @LIST-Anweisung mit 'I'-Operand nicht eroeffnet werden.
Fehlerschalter: EDT.

EDT4913 EAM WRITE ERROR
EDT4913 EAM-SCHREIBFEHLER

Bedeutung
Waehrend des Schreibens einer EAM-Datei (@LIST-Anweisung mit 'I'-Operanden) tritt ein Schreibfehler auf.
Fehlerschalter: EDT.

EDT4914 EDT OR FILE FORMAT ERROR WITH INTERRUPT WEIGHT=60
EDT4914 EDT- ODER DATEIFORMAT-FEHLER MIT UNTERBRECHUNGSGEWICHT=60

Bedeutung
Datenfehler koennen sich ergeben, wenn versucht wird, auf eine Plattendatei mit ungueltigen Schluesseln zuzugreifen. EDT verwendet beim Suchen nach Schluesseln grundsaetzlich die ersten acht Zeichen des Satzes. Ergibt sich dabei eine ungueltige Zeilennummer, kann dies zu einem Datenfehler fuehren.
Naehere Information ueber den Fehler kann dem EDT-Handbuch entnommen werden.
Fehlerschalter: EDT, DVS.

EDT4916 FILE NOT IN CATALOG
EDT4916 DATEI NICHT IM KATALOG

Bedeutung
In einer @FSTAT-, @GET-, @READ-, @INPUT-, @ELIM-, @SAVE-, @WRITE-, @UNSAVE- oder @COPY-Anweisung wurde ein Dateiname angegeben, der jedoch nicht im Katalog steht. Tritt dieser Fehler in der @FSTAT-Anweisung auf, so wird aus Kompatibilitaetsgruenden auch der DVS-Fehlerschalter gesetzt; die Ausfuehrung von @INPUT-Dateien wird jedoch nicht abgebrochen. In neu geschriebenen EDT-Prozeduren sollte jedoch nur der EDT-Fehlerschalter abgefragt werden.
Fehlerschalter: EDT, DVS (siehe Bedeutung).

EDT4918 FORMAL OPERAND MISSING
EDT4918 FORMALER OPERAND FEHLT

Bedeutung
In der @PARAMS-Anweisung wurde ein formaler Operand (&id) erwartet, aber nicht gefunden.
Fehlerschalter: EDT.

EDT4919 REQM ERROR FOR (&00) BUFFER
EDT4919 REQM-FEHLER BEI (&00)-PUFFER

Bedeutung
Waehrend der Ausfuehrung einer @FSTAT-, @STAJV- ,@ERAJV- ,@SYSTEM- oder @LIST I-Anweisung konnten fuer den Systemaufruf (&00) nicht genuegend Seiten des virtuellen Adressraums durch REQM bereitgestellt werden.
Z.B. fuer FSTAT 15 Seiten, fuer STAJV 8 Seiten, fuer CMD 8 Seiten, fuer EAM 1 Seite.
Fehlerschalter: EDT.

EDT4920 STATEMENT ILLEGAL DURING PROCEDURE FILE PROCESSING
EDT4920 ANWEISUNG UNZULAESSIG WAEHREND PROZEDUR-AUSFUEHRUNG

Bedeutung
Eine der folgenden Anweisungen sollte bearbeitet werden, waehrend eine Prozedurdatei ausgefuehrt wurde: @INPUT, @UPDATE (Format 2), @CODENAME oder @SETF GLOBAL.
Fehlerschalter: EDT.

EDT4921 STATEMENT ILLEGAL DURING '@INPUT' PROCESSING
EDT4921 ANWEISUNG UNGUELTIG WAEHREND '@INPUT'-AUSFUEHRUNG

Bedeutung
Die Anweisung @UPDATE Format 2, @CODENAME oder @GOTO wurde aus einer mit @INPUT eroeffneten Datei gelesen.
Fehlerschalter: EDT.

EDT4923 INVALID FILE NAME
EDT4923 UNGUELTIGER DATEINAME

Bedeutung
Der in einer @FSTAT-, @GET-, @READ-, @INPUT-, @OPEN-, @ELIM-, @WRITE-, @COPY-, @SAVE-, @UNSAVE- oder @DELETE-Anweisung angegebene Dateiname entspricht nicht den Konventionen fuer die Vergabe von Dateinamen.
Eine moegliche Fehlerursache:
Der in Hochkommata eingeschlossene oder in einer Zeichenfolgevariablen angegebene Dateiname beginnt mit einem Leerzeichen.
Tritt dieser Fehler bei @FSTAT auf, so wird auch der DVS-Schalter gesetzt, die Ausfuehrung von @INPUT-Dateien wird jedoch nicht abgebrochen.
In neu geschriebenen EDT-Prozeduren sollte jedoch nur der EDT-Fehlerschalter abgefragt werden.

EDT4924 INVALID FORMAL OPERAND
EDT4924 UNGUELTIGER FORMALER OPERAND

Bedeutung
Ein formaler Operand (&id) in einer @PARAMS-Anweisung ist ungueltig.
Fehlerschalter: EDT.

EDT4925 STATEMENT ONLY PERMITTED IN WORK FILE 0
EDT4925 ANWEISUNG NUR IN ARBEITSDATEI 0 MOEGLICH
(B) Routing code: * Weight: 99

Bedeutung
@OPEN (Format 1) ist in einer Prozedurdatei nicht erlaubt.
Eine Datei kann nur in der Arbeitsdatei 0 real eroeffnet werden.
Fehlerschalter: EDT.

EDT4926 INVALID KEY
EDT4926 UNGUELTIGER SCHLUESSEL

Bedeutung
In der Anweisung @GET '...' N, @READ '...' KEY oder @ELIM '...' wurde auf einen Satz mit ungueltigem Schluessel zugegriffen. Die Verarbeitung der Anweisung wurde abgebrochen.
Die Verarbeitung von @INPUT-Dateien wird wie bei DVS-Fehlern abgebrochen Im Stapelbetrieb, oder wenn der EDT mit RDATA von SYSDTA liest, beendet sich der EDT mit der Meldung 'EDT8001 EDT ABNORMAL BEENDET'.
Fehlerschalter: EDT, DVS.

EDT4927 INVALID KEY IN FILE OPENED IN REAL MODE
EDT4927 ZUGRIFF AUF SATZ MIT UNGUELTIGEM SCHLUESSEL

Bedeutung
Eine ISAM-Datei ist real eroeffnet (@OPEN), und es wurde auf einen Satz mit ungueltigem Schluessel zugegriffen. Die Verarbeitung der Anweisung wurde abgebrochen, und die gerade bearbeitete Datei wurde geschlossen.
Die Verarbeitung von @INPUT-Dateien wird wie bei DVS-Fehlern abgebrochen.
Im Stapelbetrieb, oder wenn der EDT mit RDATA von SYSDTA liest, beendet sich der EDT mit der Meldung 'EDT8001EDT ABNORMAL BEENDET'.
Fehlerschalter: EDT, DVS.

EDT4928 INVALID VALUE
EDT4928 UNGUELTIGER WERT

Bedeutung
Ein Wert in einem @PARAMS-Schluesselwort oder einem @DO-Operanden ist ungueltig. Der haeufigste Grund fuer diesen Fehler ist, dass ein Hochkomma nicht paarweise angegebenen wurde.
Fehlerschalter: EDT.

EDT4929 ISAM 'RECORD-FORMAT=FIXED' NOT SUPPORTED
EDT4929 ISAM-'RECORD-FORMAT=FIXED' NICHT UNTERSTUETZT

Bedeutung
Es wurde versucht, eine Datei mit fester Satzlaenge mit der Anweisung @OPEN und einem /SET-FILE-LINK-Kommando mit LINK-NAME=EDTMAIN zu bearbeiten.
Fehlerschalter: EDT.

EDT4930 'KEY-POSITION <>1' AND 'RECORD-FORMAT=FIXED' NOT SUPPORTED
EDT4930 'KEY-POSITION <> 1' UND 'RECORD-FORMAT=FIXED' NICHT UNTERSTUETZT

Bedeutung
In einer @GET- oder @SAVE-Anweisung wurde eine ISAM-Datei mit dem Kommando /SET-FILE-LINK ..,LINK-NAME=EDTISAM,ACCESS-METHOD=ISAM (REC-FORM= FIXED..) zugeordnet, die Datei hat aber nicht 'KEY-POSITION=1'.
Fehlerschalter: EDT.

EDT4931 KEY-LENGTH TOO BIG
EDT4931 KEY-LENGTH ZU GROSS

Bedeutung
Es wurde versucht, mit einer @GET-, @SAVE-, @ELIM- oder @OPEN-Anweisung auf eine Datei mit KEY-LENGTH > 8 zuzugreifen.
Fehlerschalter: EDT.

EDT4932 LINE NUMBER NOT FOUND
EDT4932 ZEILENNUMMER NICHT GEFUNDEN

Bedeutung
- Fuer <string> wurde eine Zeilennummer angegeben, aber die Zeile ist nicht in der Datei oder die Datei ist leer.
  Tritt der Fehler bei der Ausfuehrung einer EDT-Prozedur (@DO) bzw. @INPUT-Datei auf, so wird diese Meldung nicht ausgegeben, ausser die Protokollierung ist ueber den PRINT-Operanden eingeschaltet.
  In diesem Fall wird auch der EDT-Fehlerschalter nicht gesetzt.
- In der @COMPARE-Anweisung wird ein ungueltiger Zeilenbereich angegeben. In diesem Fall wird die Meldung immer ausgegeben, der EDT-Fehlerschalter wird gesetzt.

Fehlerschalter: EDT (siehe Bedeutungstext).

EDT4933 MODULE LOADING NOT POSSIBLE
EDT4933 LADEN DES MODULS NICHT MOEGLICH

Bedeutung
Es ist nicht moeglich, den Modul (z.B. IEDTCALL) mit der @RUN- oder @USE-Anweisung zu laden.
Fehlerschalter: EDT.

EDT4934 MAIN FILE IS SAM
EDT4934 HAUPTDATEI IST SAM

Bedeutung
Der erste in der @OPEN-Anweisung angegebene Dateiname ist der einer SAM-Datei. Durch die Angabe von '@OPEN <file1> AS <file2>' kann eine Kopie der SAM-Datei real bearbeitet werden.
Fehlerschalter: EDT.

EDT4935 MAIN FILE OPENED REAL
EDT4935 HAUPTDATEI ALS REAL-DATEI EROEFFNET

Bedeutung
Es wurde versucht, eine Anweisung auszufuehren, die nicht erlaubt ist, solange die Hauptdatei durch @OPEN (Format 1) real eroeffnet ist (z.B. @RENUMBER).
Fehlerschalter: EDT.

EDT4936 'KEY-POSITION <>5' AND 'RECORD-FORMAT=VARIABLE' NOT SUPPORTED
EDT4936 'KEY-POSITION <>5' UND 'RECORD-FORMAT=VARIABLE' NICHT UNTERSTUETZT
(B) Routing code: * Weight: 99

Bedeutung
Eine Datei mit diesen Katalogeigenschaften kann mit @OPEN (Format 2) nicht bearbeitet werden.
Fehlerschalter: EDT.

EDT4937 NO MORE SPACE FOR OPERAND VALUES
EDT4937 KEIN PLATZ FUER OPERANDEN-WERTE

Bedeutung
Ein aktueller Operandenwert oder ein formaler Schluesselwortwert bestehend aus n Zeichen, belegt (n+1) Byte auf einer Seite des virtuellen Adressraums, die fuer Prozedurdateiargumente (Werte) reserviert sind.
Mit Ausnahme von leeren Operanden wird diese Meldung ausgegeben, wenn ein Wert bewirkt, dass mehr als 4096 Byte verwendet werden.
Naehere Information ueber den Fehler kann dem EDT-Handbuch entnommen werden.
Fehlerschalter: EDT.

EDT4938 NO MORE SPACE FOR OPERANDS
EDT4938 KEIN PLATZ MEHR FUER OPERANDEN

Bedeutung
Ein formaler Operand der Laenge n, einschliesslich des '&'-Zeichens, belegt (n+4) Byte auf einer Seite des virtuellen Adressraums, die fuer die formalen Operanden von Prozedurdateien reserviert ist. Bewirkt ein Operand, dass mehr als 4096 Byte (eine Seite) verwendet werden, so wird diese Meldung ausgegeben.
Die Seite fuer formale Operanden wird nicht zugewiesen, wenn keine Operanden benutzt werden. Sie wird zurueckgegeben, nachdem alle Prozedurdateien mit der @DROP-Anweisung geloescht wurden oder wenn keine @INPUT-Dateien mehr aktiv sind.
Naehere Information ueber den Fehler kann dem EDT-Handbuch entnommen werden.
Fehlerschalter: EDT.

EDT4939 '@END' WITHOUT '@PROC' STATEMENT
EDT4939 '@END' OHNE '@PROC' ANWEISUNG

Bedeutung
Eine @END-Anweisung wurde gegeben, aber es gibt keine aufgeklappte Prozedurdatei, die zu beenden ist, d.h. man befindet sich schon in der Arbeitsdatei 0.
Fehlerschalter: EDT.

EDT4940 POSITION VALUES NOT ASCENDING
EDT4940 POSITIONSWERTE NICHT AUFSTEIGEND

Bedeutung
Die Positionswerte, die in einer @TABS-Anweisung zur Definition der Hardware-Tabulatoren angegeben werden, muessen aufsteigend sein.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT4941 NO POSITIONS DEFINED
EDT4941 POSITIONEN NICHT DEFINIERT

Bedeutung
Vor Verwendung der Tabulatoren muessen die Positionen definiert werden.
Fehlerschalter: EDT.

Maßnahme
Positionen mittels @TABS-Anweisung definieren.

EDT4942 STATEMENT ONLY POSSIBLE IN PROCEDURE FILE
EDT4942 ANWEISUNG NUR IN PROZEDURDATEI MOEGLICH

Bedeutung
Eine @RETURN-, @GOTO- oder @IF-Anweisung kann nur waehrend der Ausfuehrung einer Prozedurdatei bearbeitet werden.
Fehlerschalter: EDT.

EDT4943 CHANGE OF CCS NOT POSSIBLE – WORK FILES NOT EMPTY
EDT4943 CCS KANN NICHT GEWECHSELT WERDEN – NICHT ALLE ARBEITSDATEIEN LEER
(B) Routing code: * Weight: 99

Bedeutung
Ein Wechsel des kodierten Zeichensatzes ist nur moeglich, wenn alle Arbeitsdateien leer sind. Entweder wurde ein @CODENAME-Anweisung eingegeben, oder eine Datei mit einem CCS verschieden vom aktuellen CCS sollte eingelesen oder eroeffnet werden (@READ, @OPEN,..).
Fehlerschalter: EDT.

Maßnahme
Eroeffnete Dateien schliessen (@CLOSE), Arbeitsdateien loeschen (@DELETE) und danach Anweisung wiederholen.

EDT4944 @PARAMS STATEMENT MISSING
EDT4944 @PARAMS–ANWEISUNG FEHLT

Bedeutung
Die @DO-Anweisung enthaelt Operanden, aber die Prozedurdatei enthaelt keine @PARAMS-Anweisung, oder sie ist nicht die erste Zeile der Prozedurdatei.
Fehlerschalter: EDT.

EDT4945 NOT POSSIBLE ON THIS TERMINAL
EDT4945 AUF DIESER DATENSTATION NICHT MOEGLICH

Bedeutung
Die @UPDATE-Anweisung im Format 2 wurde auf einer Datenschreibstation eingegeben, oder es wurde versucht, mit einer @VDT-Anweisung das Bildschirmformat zu aendern, was nur fuer eine Datensichtstation 9763 moeglich ist.
Fehlerschalter: EDT.

EDT4946 OVERFLOW ERROR
EDT4946 UEBERLAUF-FEHLER

Bedeutung
Bei einer Rechenoperation mit der Anweisung @SET (Format 1) wurde der positive bzw. negative Maximalwert einer Ganzzahlvariablen (2^31-1, -2^31) ueberschritten.
Fehlerschalter: EDT.

EDT4947 PAM FILE NOT SUPPORTED
EDT4947 PAM-DATEIEN WERDEN NICHT UNTERSTUETZT

Bedeutung
Es wurde versucht, mit einer der Anweisungen @GET, @READ, @INPUT, @OPEN, @ELIM, @SAVE oder @WRITE eine PAM-Datei zu bearbeiten. PAM-Dateien unterstuetzt der EDT nicht.
Fehlerschalter: EDT.

EDT4948 POSITIONAL OPERAND AFTER KEYWORD OPERAND
EDT4948 STELLUNGSOPERAND NACH SCHLUESSELWORTOPERAND

Bedeutung
In der @DO-Anweisung wurde ein Stellungsoperand hinter einem Schluesselwortoperand angegeben.
Fehlerschalter: EDT.

EDT4949 PROCEDURE FILE IS EMPTY
EDT4949 PROZEDURDATEI IST LEER

Bedeutung
Eine mit @DO gestartete EDT-Prozedur ist leer. Die Prozedurdatei ist zwar mit @PROC und @END definiert worden, enthaelt aber keine Datensaetze bzw. EDT-Anweisungen.
Fehlerschalter: EDT.

EDT4950 PROCEDURE FILE IS UNDEFINED
EDT4950 PROZEDURDATEI IST NICHT DEFINIERT

Bedeutung
In einer Anweisung (@DO, @COMPARE) wird eine Prozedurdatei angegeben, die nicht mit @PROC definiert wurde.
Fehlerschalter: EDT.

EDT4951 WORK FILE IS EMPTY. STATEMENT NOT PROCESSED
EDT4951 ARBEITSDATEI LEER. ANWEISUNG NICHT AUSGEFUEHRT

Bedeutung
Die Anweisung bezieht sich auf eine Zeilennummer, die nicht gefunden werden kann, da die Arbeitsdatei leer ist.

EDT4954 REQM ERROR. PLEASE RECEIPT WITH "Y"
EDT4954 REQM-FEHLER. BITTE MELDUNG MIT 'Y' QUITTIEREN

Bedeutung
Der Versuch des EDT, zusaetzlichen Speicherplatz anzufordern, wird mit Returncode abgewiesen, oder von einem EDT-Unterprogramm (@RUN) wird die ENTRLINE-Routine aufgerufen, es ist aber kein virtueller Speicher verfuegbar.
Fehlerschalter: wird nicht gesetzt.

Maßnahme
Y: Der EDT meldet sich mit der naechsten freien Zeilennummer.
Sonst: Die Abfrage wird wiederholt.
Tritt der Speichermangel waehrend des Ablaufs einer EDT-Prozedur auf, kann diese mittels K2 und /INTR vom Benutzer abgebrochen werden.

EDT4955 PROCEDURE FILE(S) NOT YET TERMINATED
EDT4955 PROZEDUR-DATEI(EN) NOCH NICHT BEENDET

Bedeutung
Die @DROP-Anweisung ist nicht erlaubt, wenn noch Prozedurdateien im Kellerungseintrag abgespeichert sind, d.h. eine weitere @PROC-Anweisung wurde gegeben bevor die vorherige beendet war.
Fehlerschalter: EDT.

EDT4956 SYSDTA EOF
EDT4956 SYSDTA EOF

Bedeutung
Der EDT gab einen Lesebefehl, aber eine Dateiendebedingung trat auf.
Wird 'EOF' von RDATA (@EDIT ONLY-Modus) gemeldet, so schaltet der EDT um auf WRTRD (@EDIT-Modus). Wird 'EOF' vom WRTRD bzw. im Stapelbetrieb gemeldet, so gibt der EDT einen BKPT. Die Dateiendebedingung kann dann zurueckgesetzt und mit dem /RESUME-PROGRAM-Kommando fortgefahren werden.
Fehlerschalter: EDT.

EDT4957 SYSDTA NOT ASSIGNED OR READ ERROR
EDT4957 SYSDTA NICHT ZUGEWIESEN ODER FEHLER BEIM LESEN

Bedeutung
Der RDATA-Makro lieferte den Returncode X'14' oder X'18'. Der EDT-Lauf wird daraufhinmit der Meldung 'EDT8001EDT ABNORMAL BEENDET' abgebrochen.
Fehlerschalter: EDT.

EDT4958 @SYSTEM STATEMENT INCORRECT
EDT4958 FEHLER IN @SYSTEM-ANWEISUNG

Bedeutung
Das in der @SYSTEM-Anweisung angegebene Kommando enthaelt einen fehlerhaften Operanden oder gab einen DVS-Fehlers aus.
Es wird von CMD-Makro mit dem Returncode X'10' zurueckgewiesen.
Fehlerschalter: DVS.

EDT4959 PROCEDURE FILE ALREADY ACTIVE
EDT4959 PROZEDUR-DATEI BEREITS AKTIV

Bedeutung
Eine @PROC-Anweisung fuer eine Prozedurdatei, die schon ueber eine @DO-Anweisung aktiviert ist, ist nicht erlaubt.
Fehlerschalter: EDT.

EDT4960 TIAM MACRO ERROR
EDT4960 TIAM-MAKRO-FEHLER

Bedeutung
Der Returncode X'04' oder X'08' wird vom Makro WROUT, WRTRD, RDATA oder MSG7
gemeldet. Bei Returncode X'08' wird zusaetzlich ein Areadump ausgegeben.
Nach diesem Fehler wird der EDT immer mit der Meldung beendet 'EDT8001EDT ABNORMAL BEENDET'.
Fehlerschalter: wird nicht gesetzt.

EDT4961 TOO MANY PROCEDURE FILES ACTIVE
EDT4961 ZU VIELE AKTIVE PROZEDUR-DATEIEN

Bedeutung
Fehler in der @DO-Anweisung: Es werden mehr als 22 Prozedurdateien gleichzeitig bearbeitet.
Fehlerschalter: EDT.

EDT4962 TOO MANY FILES
EDT4962 ZU VIELE DATEIEN

Bedeutung
Es gibt keinen weiteren Speicherplatz fuer verschachtelte (nicht beendete) Definitionen von Prozedur-Dateien oder fuer eine INPUT-Datei.
Fehlerschalter: EDT.

EDT4963 TOO MANY OPERANDS
EDT4963 ZU VIELE OPERANDEN

Bedeutung
Es gibt mehr aktuelle Operanden in der @DO-Anweisung als formale Operanden in der @PARAMS-Anweisung.
Fehlerschalter: EDT.

EDT4964 TOO MANY POP OPERATIONS
EDT4964 OEFTER AUF- ALS ABGESTIEGEN

Bedeutung
Eine '@'-Anweisung wurde eingegeben, um im 3-stufigen Kellerungseintrag aufzusteigen. Dadurch wird oefter auf- als abgestiegen, oder es wird nie dreimal im Kellerungseintrag abgestiegen. (Es wird nur dann von vorne begonnen, wenn der Bereich voll ist.)
Fehlerschalter: EDT.

EDT4965 TOO MANY POSITIONAL OPERANDS
EDT4965 ZU VIELE STELLUNGS-OPERANDEN

Bedeutung
In einer @DO-Anweisung gibt es mehr Stellungsoperanden als in der @PARAMS-Anweisung angegeben.
Fehlerschalter: EDT.

EDT4966 'UPDATE' FOR ISAM FILE NOT POSSIBLE
EDT4966 'UPDATE' FUER ISAM-DATEI NICHT MOEGLICH

Bedeutung
Eine @WRITE-Anweisung mit UPDATE-Funktion wurde fuer eine ISAM-Datei eingegeben.
Fehlerschalter: EDT.

EDT4967 'UPDATE' FOR SAM FILE NOT POSSIBLE
EDT4967 'UPDATE' FUER SAM-DATEI NICHT MOEGLICH

Bedeutung
Eine @SAVE-Anweisung mit UPDATE-Funktion wurde fuer eine SAM-Datei eingegeben.
Fehlerschalter: EDT.

EDT4968 WORK FILE NOT EMPTY
EDT4968 ARBEITSDATEI NICHT LEER

Bedeutung
Bei @OPEN befinden sich noch Zeilen in der Arbeitsdatei.
@OPEN ist nur dann erlaubt, wenn die Arbeitsdatei leer ist.
Fehlerschalter: EDT.

EDT4969 WRONG VERSION: (&00) (&01)
EDT4969 FALSCHE VERSION: (&00) (&01)

Bedeutung
In einer Anweisung wurde ein Dateiname mit falscher Versionsnummer angegeben. Der EDT gibt in dieser Meldung den Dateinamen mit der richtigen Versionsnummer aus. Wird die Datei nur gelesen, so wird die Anweisung ausgefuehrt. Erfolgt ein Schreibzugriff, so wird die Anweisung nicht ausgefuehrt. Wird eine Anweisung mit falscher Versionsnummer (Schreib- und Lesezugriff) aus einer @INPUT-Datei gelesen, so bricht die Prozedur ab.
Fehlerschalter: DVS.

EDT4971 FIRST FILE EMPTY OR NOT CATALOGED
EDT4971 ERSTE DATEI LEER ODER NICHT IM KATALOG

Bedeutung
Eine AS-Datei wurde in der @OPEN-Anweisung angegeben, aber die erste Datei ist entweder leer oder nicht katalogisiert.
Fehlerschalter: EDT.

EDT4972 @ELIM STATEMENT FOR SAM FILE ILLEGAL
EDT4972 @ELIM-ANWEISUNG FUER SAM DATEI UNZULAESSIG

Bedeutung
Fehlerschalter: EDT.

EDT4973 @UPDATE STATEMENT IN BINARY MODE NOT POSSIBLE
EDT4973 @UPDATE-ANWEISUNG IM BINAER-MODUS NICHT MOEGLICH

Bedeutung
Mit der @INPUT-Anweisung wurde der Binaer-Modus eingeschaltet und dann fuer eine Korrektur eine @UPDATE-Anweisung (Format 2) gegeben.
Fehlerschalter: EDT.

EDT4974 LINE NOT IN PROCEDURE FILE
EDT4974 ZEILE NICHT IN PROZEDUR-DATEI

Bedeutung
Die in einer @GOTO-Anweisung angegebene Zeilennummer existiert nicht in der Prozedurdatei.
Fehlerschalter: EDT.

EDT4975 BIND NOT SUCCESSFUL
EDT4975 BIND NICHT ERFOLGREICH

Bedeutung
Der DBL konnte keinen Modul mit dem in der @RUN-Anweisung angegebenen ENTRY- oder CSECT-Namen (ENTRY) in der angegebenen Modulbibliothek finden.
Fehlerschalter: EDT.

```
EDT4976    STATEMENT INHIBITED FOR USER
EDT4976    ANWEISUNG FUER BENUTZER GESPERRT
```

Bedeutung
Der Benutzer gab eine Anweisung ein (z.B. @RUN, @LOAD,...),
die vom rufenden Programm gesperrt wurde.

```
EDT4977    'RECORD-FORMAT=UNDEFINED' ILLEGAL
EDT4977    'RECORD-FORMAT=UNDEFINED' NICHT ZULAESSIG

EDT4978    INVALID IN F-MODE
EDT4978    NICHT ZULAESSIG FUER F-MODUS

EDT4980    ILLEGAL OR UNKNOWN CCS NAME
EDT4980    UNZULAESSIGER ODER UNBEKANNTER CCS-NAME
```

Bedeutung
Der in der @CODENAME angegebene CCSN oder der CCSN der einzulesenden Datei bzw. Bibliothekselementes (@READ, @OPEN, @COPY, @INPUT) ist unzulaessig oder im System nicht bekannt.
Fehlerschalter: EDT.

```
EDT4981    RECORD-SIZE > 256. FILE NOT WRITTEN
EDT4981    RECORD-SIZE > 256. DATEI NICHT GESCHRIEBEN

EDT4982    REQUESTED JV NOT CATALOGED
EDT4982    JV NICHT IM KATALOG
```

Bedeutung
In einer @STAJV-, @ERAJV- oder @GETJV-Anweisung wurde der Name einer Jobvariablen angegeben, die nicht im Katalog vorhanden ist.
Fehlerschalter: EDT, DVS.

```
EDT5065    INVALID RANGE: LOWER LIMIT > UPPER LIMIT
EDT5065    UNGUELTIGER SATZBEREICH: UNTERE GRENZE > OBERE GRENZE
```

Bedeutung
Die erste im Satzbereich angegebene Zeilennummer ist groesser als die zweite.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT5078    RECORD WITH SAME KEY EXISTS: 'INSERT' NOT POSSIBLE
EDT5078    SATZ MIT GLEICHEM SATZSCHLUESSEL EXISTIERT. KEIN 'INSERT' MOEGLICH
```

Bedeutung
Der in der 'INSERT <...> AT'- Anweisung angegebene Satzschluessel existiert bereits. Es kann kein neuer Satz eingefuegt werden.

Maßnahme
Satzschluessel aendern und Anweisung wiederholen.

```
EDT5079    STRING TO BE INSERTED IS EMPTY. 'INSERT' NOT POSSIBLE
EDT5079    ZEICHENKETTE IST LEERSTRING. KEIN 'INSERT' MOEGLICH
```

Bedeutung
Es ist nicht moeglich, mit der INSERT-Anweisung eine leere Zeichenkette als Satz einer Arbeitsdatei einzufuegen.

```
EDT5080    OPERANDS '$0'..'$9', 'FIRST', 'LAST' NOT SUPPORTED
EDT5080    OPERANDEN '$0'..'$9', 'FIRST', 'LAST' NICHT ERLAUBT
```

Bedeutung
Die Angabe der Operanden <adatvar>, FIRST (bzw.FI) und LAST (bzw. LA) hier unzulaessig.

Maßnahme
Anstatt @SETF FI kann @SETF oder @SETF % verwendet werden.
Anstatt @SETF LA kann @SETF $ verwendet werden.
Korrigierte Anweisung wiederholen.

```
EDT5121    'CLOSE REAL' STATEMENT VALID ONLY IN WORK FILE 0
EDT5121    'CLOSE REAL'-ANWEISUNG NUR IN ARBEITSDATEI 0 ZULAESSIG
```

Maßnahme
Die Datei, die real geschlossen werden soll, in der Arbeitsdatei 0 bearbeiten.

```
EDT5122    NO FILE NAME
EDT5122    KEIN DATEINAME
```

Bedeutung
Es wurde kein Dateiname angegeben, oder das Format fuer die Angabe des Dateinamens ist falsch.

Maßnahme
Dateinamen in der Form F=<dateiname>, L=<bibliotheksname> oder E=<elementname> angeben.

```
EDT5123    'CLOSE REAL' NOT POSSIBLE: '(&00)' IS NOT OPENED REAL
EDT5123    KEIN 'CLOSE REAL' MOEGLICH: '(&00)' IST NICHT REAL GEOEFFNET
```

Bedeutung
(&00): Datei.

Maßnahme
Anweisung ohne Operand REAL wiederholen.

```
EDT5124    'OPEN REAL' STATEMENT VALID ONLY IN WORK FILE 0
EDT5124    'OPEN REAL'-ANWEISUNG NUR IN ARBEITSDATEI 0 MOEGLICH
```

Maßnahme
Die Datei, die real geoeffnet werden soll, in der Arbeitsdatei 0 bearbeiten.

```
EDT5125    'OPEN REAL' VALID ONLY FOR ISAM FILES
EDT5125    'OPEN REAL'-ANWEISUNG NUR FUER ISAM-DATEIEN ZULAESSIG
```

Bedeutung
Es ist nicht moeglich, die angegebene Datei mit OPEN REAL zu bearbeiten, da sie keine ISAM-Datei ist.

Maßnahme
Datei virtuell bearbeiten.

```
EDT5126    '(&00)' NOT POSSIBLE: WORK FILE 0 IS OPEN
EDT5126    KEIN '(&00)' MOEGLICH: ARBEITSDATEI 0 IST OFFEN
```

Bedeutung
In der Arbeitsdatei 0 wurde eine ISAM-Datei mit der @OPEN-Anweisung (Format 1) real eroeffnet. Ein anschliessendes @OPEN (Format 2) oder @XOPEN wird abgewiesen.
Fehlerschalter: EDT.

Maßnahme
Die mit @OPEN geoeffnete Datei mit der @CLOSE-Anweisung schliessen.

```
EDT5170    TEXT IN HALT STATEMENT NOT PERMISSIBLE IN CONTROL MODE
EDT5170    TEXT IN HALT-ANWEISUNG IM CONTROL-MODUS NICHT ERLAUBT

EDT5171    NO EDT V15 OR EDT V16.0 STATEMENTS IN 'CONTROL' MODE
EDT5171    KEINE EDT-V15 ODER EDT-V16.0 ANWEISUNGEN IM CONTROL-MODUS
```

Bedeutung
Es ist nicht moeglich, EDT-V15- oder EDT-V16.0-Anweisungen im CONTROL-Modus anzugeben.

```
EDT5177    NO FILE TO CLOSE
EDT5177    KEINE OFFENE DATEI VORHANDEN

EDT5179    PLAM MEMBER MISSING. STATEMENT NOT PROCESSED
EDT5179    PLAM-ELEMENT FEHLT. ANWEISUNG NICHT AUSGEFUEHRT

EDT5180    '@CLOSE' OR '@CLOSE NOWRITE' EXPECTED
EDT5180    '@CLOSE' ODER '@CLOSE NOWRITE' ERWARTET
```

Bedeutung
Es wurde versucht, mit der @OPEN- oder @XOPEN-Anweisung eine Datei oder ein Bibliothekselement zu bearbeiten, obwohl in dieser Arbeitsdatei bereits eine Datei oder ein Bibliothekselement eroeffnet ist. Aus dem gleichen Grund ist auch ein @DROP dieser Arbeitsdatei nicht moeglich.

Maßnahme
Die bearbeitete Datei oder das Bibliothekselement mit @CLOSE oder @CLOSE NOWRITE schliessen und die Anweisung wiederholen.

EDT5181 NO LIBRARY NAME DEFINED
EDT5181 KEIN BIBLIOTHEKSNAME DEFINIERT

Bedeutung
Die Anweisung konnte nicht ausgefuehrt werden, da der Bibliotheksname nicht definiert wurde.

EDT5182 'CLOSE REAL' NOT POSSIBLE FOR PLAM MEMBERS
EDT5182 KEIN 'CLOSE REAL' MOEGLICH FUER PLAM-ELEMENTE

Maßnahme
Das eroeffnete PLAM-Element mit @CLOSE oder @CLOSE NOWRITE schliessen.

EDT5183 'CLOSE REAL' INVALID FOR SAM FILES
EDT5183 'CLOSE REAL' UNZULAESSIG FUER SAM-DATEIEN

Maßnahme
Die SAM-Datei mit CLOSE oder CLOSE NOWRITE schliessen.

EDT5188 NUMBER OF LINES NOT PERMISSIBLE
EDT5188 ZEILENANZAHL NICHT ERLAUBT

Bedeutung
Im Operand SPLIT der @PAR-Anweisung wurde eine Zeilenanzahl fuer das zweite Arbeitsfenster angegeben, bei der eines der Arbeitsfenster eine Zeilenanzahl kleiner 2 haette.

EDT5189 '(&00)' NOT POSSIBLE: A FILE IS OPENED IN WORK FILE 9
EDT5189 '(&00)' NICHT MOEGLICH: IN ARBEITSDATEI 9 IST EINE DATEI EROEFFNET

Bedeutung
Eine @SHOW-Anweisung konnte nicht ausgefuehrt werden, da in Arbeitsdatei 9 eine Datei eroeffnet ist.

Maßnahme
Die in der Arbeitsdatei 9 eroeffnete Datei schliessen.

EDT5191 '(&00)' NOT POSSIBLE. WORK FILE (&01) IS NOT EMPTY
EDT5191 '(&00)' NICHT MOEGLICH. ARBEITSDATEI (&01) IST NICHT LEER

Bedeutung
Die Anweisung (&00) (z.B. @OPEN Format 2, @XOPEN,...) kann nur in einer leeren Arbeitsdatei ausgefuehrt werden.
Fehlerschalter: EDT.

Maßnahme
Andere Arbeitsdatei waehlen oder angegebene Arbeitsdatei loeschen und Anweisung wiederholen.

```
EDT5221    READ ERROR ((&00)): DMS ERROR CODE: '(&01)'
EDT5221    LESE-FEHLER ((&00)): DVS-FEHLERCODE: '(&01)'
```

Bedeutung
Die @OPEN- oder @COPY-Anweisung wurde nicht ausgefuehrt, da ein Lesefehler in der Zugriffsmethode (&00) aufgetreten ist.
(&01): DVS-Fehlercode.
Naehere Information ueber den DVS-Fehler kann mit dem ISP-Kommando /HELP DMS(&01) oder dem SDF-Kommando /HELP-MESS DMS(&01) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. einem der BS2000 DVS-Handbuecher entnommen werden.

```
EDT5224    INVALID ACCESS-METHOD
EDT5224    'ACCESS-METHOD' UNGUELTIG
```

Bedeutung
Die Datei, die mit einer @COPY-, @OPEN- oder @WRITE-Anweisung (Format 2) angesprochen wurde, kann vom EDT nicht verarbeitet werden, da in EDT diese Zugriffsmethode nicht moeglich ist.
EDT kann derzeit nur SAM- und ISAM-Dateien bearbeiten.

Maßnahme
Datei in eine SAM- oder ISAM-Datei konvertieren.

```
EDT5225    INVALID RECORD-FORMAT
EDT5225    'RECORD-FORMAT' UNGUELTIG
```

Bedeutung
Das RECORD-FORMAT-Attribut einer Datei, die mit einer @COPY-, @OPEN- oder @WRITE-Anweisung (Format 2) angesprochen wurde, kann vom EDT nicht verarbeitet werden.
EDT unterstuetzt derzeit nur RECORD-FORMAT=VARIABLE.

Maßnahme
Datei in eine Datei mit RECORD-FORMAT=VARIABLE konvertieren.

```
EDT5226    '@OPEN' NOT POSSIBLE: RECORD SIZE > 256
EDT5226    KEIN '@OPEN' MOEGLICH: RECORD-SIZE > 256
```
(B) Routing code: * Weight: 99

Bedeutung
Eine Datei mit fester Satzlaenge groesser als 256 Zeichen kann nicht mit @OPEN behandelt werden, da der Inhalt der Datei ab Spalte 257 verloren gehen wuerde.
Fehlerschalter: EDT.

EDT5233 SET ERROR (ISAM): DMS ERROR CODE: '(&00)'
EDT5233 SET-FEHLER (ISAM): DVS-FEHLERCODE: '(&00)'

Bedeutung
Die @COPY-Anweisung (Format 2) wurde nicht ausgefuehrt, da ein SET-Fehler aufgetreten ist.
(&00): DVS-Fehlercode.
Naehere Information ueber den DVS-Fehler kann mit dem ISP-Kommando /HELP DMS(&00) oder dem SDF-Kommando /HELP-MESS DMS(&00) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. einem der BS2000 DVS-Handbuecher entnommen werden.

EDT5237 WRITE ERROR ((&00)): DMS ERROR CODE: '(&01)'
EDT5237 SCHREIB-FEHLER ((&00)): DVS-FEHLERCODE: '(&01)'

Bedeutung
Die @CLOSE- oder @WRITE- Anweisung wurde nicht ausgefuehrt, da ein Schreib-Fehler in der Zugriffsmehtode (&00) aufgetreten ist.
(&01): DVS-Fehlercode.
Naehere Information ueber den DVS-Fehler kann mit dem ISP-Kommando /HELP DMS(&01) oder dem SDF-Kommando /HELP-MESS DMS(&01) im Systemmodus erfragt oder dem BS2000-Handbuch "Systemmeldungen" bzw. einem der BS2000 DVS-Handbuecher entnommen werden.

EDT5241 FILE '(&00)' FOR COPY OPERATION DOES NOT EXIST
EDT5241 KOPIERDATEI '(&00)' EXISTIERT NICHT

Bedeutung
Die in der COPY-Anweisung angegebene Datei (&00) existiert nicht. Die Anweisung wurde nicht ausgefuehrt.
(&00): Datei.

EDT5244 'COPY' STATEMENT WITH 'KEEP' ONLY VALID FOR ISAM FILES
EDT5244 'COPY'-ANWEISUNG MIT 'KEEP' NUR FUER ISAM-DATEIEN ZULAESSIG

EDT5245 INVALID RECORD KEY
EDT5245 UNZULAESSIGER SATZSCHLUESSEL

Bedeutung
Es ist nicht moeglich, eine ISAM-Datei mit alphanumerischem Schluessel einzulesen.

```
EDT5246    SECONDARY KEY(S) INCOMPLETLY SET
EDT5246    SEKUNDAER-SCHLUESSEL UNVOLLSTAENDIG
```
(B) Routing code: * Weight: 99

Bedeutung
Die NKISAM-Datei wurde geschlossen. Danach trat beim Setzen der Sekundaerschluessel ein Fehler auf.
Fehlerschalter: EDT.

Maßnahme
Datenbereiche, die als Sekundaerschluessel dienen, ueberpruefen.

```
EDT5250    ERROR CODE '(&00)' IN PLAM FUNCTION '(&01)'
EDT5250    FEHLERCODE '(&00)' IN PLAM-FUNKTION '(&01)'
```

Bedeutung
Die waehrend der Bearbeitung der Anweisung aufgerufene PLAM-Funktion (&01) (z.B. DETACH, ATTACH,..) lieferte den Fehlercode (&00).
Die Anweisung wurde nicht ausgefuehrt.

```
EDT5251    ERROR CODE '(&00)' IN PLAM FUNCTION 'CLOSE'
EDT5251    FEHLERCODE '(&00)' IN PLAM-FUNKTION 'CLOSE'
```

Bedeutung
Die waehrend der Bearbeitung der CLOSE-Anweisung aufgerufene PLAM-Funktion CLOSE lieferte den Fehlercode (&00).
Die Anweisung wurde nicht ausgefuehrt.

```
EDT5252    MAXIMUM LINE NUMBER
EDT5252    MAXIMALE ZEILENNUMMER
```

Bedeutung
Die Zeilennummer 9999.9999 ist erzeugt oder ueberschritten worden.
Beim Einlesen ist die Anzahl der Saetze oder Listenelemente zu gross.
Fehlerschalter: EDT.

```
EDT5253    SPECIFIED FILE IS NOT A PLAM LIBRARY
EDT5253    ANGEGEBENE DATEI IST KEINE PLAM-BIBLIOTHEK
```

Bedeutung
Auf die Datei, die im Operand LIBRARY der @OPEN-, @COPY-, @DELETE-, @INPUT- oder @SHOW-Anweisung angegeben wurde oder in einer @PAR-Anweisung als LIBRARY vordefiniert wurde, kann mit PLAM nicht zugegriffen werden.

```
EDT5254    (&00) NOT IN SYSTEM
EDT5254    (&00) NICHT IM SYSTEM
```

Bedeutung
Die Anweisung konnte nicht ausgefuehrt werden, da das Subsystem (&00) nicht im System verfuegbar ist.
Fehlerschalter: EDT, DVS.

```
EDT5255    ERROR CODE '(&00)' IN PLAM FUNCTION 'GETA'
EDT5255    FEHLERCODE '(&00)' IN PLAM-FUNKTION 'GETA'
```

Bedeutung
Die waehrend der Bearbeitung der Anweisung aufgerufene PLAM-Funktion GETA lieferte den Fehlercode (&00). Die Anweisung wurde nicht ausgefuehrt.

```
EDT5256    ERROR CODE '(&00)' IN PLAM FUNCTION 'ATTACH' / DMS ERROR CODE '(&01)'
EDT5256    FEHLERCODE '(&00)' IN PLAM-FUNKTION 'ATTACH' / DVS-FEHLERCODE '(&01)'
```

Bedeutung
Die waehrend der Bearbeitung der Anweisung aufgerufene PLAM-Funktion ATTACH lieferte den Fehlercode (&00).
Die Anweisung wurde nicht ausgefuehrt.
(&01): DVS-Fehlercode.
Naehere Information ueber den DVS-Fehler kann mit dem ISP-Kommando /HELP DMS(&01) oder dem SDF-Kommando /HELP-MESS DMS(&01) im Systemmodus erfragt oder dem BS2000-Handbuch 'Systemmeldungen' bzw. einem der BS2000 DVS-Handbuecher entnommen werden.

```
EDT5257    ERROR CODE '(&00)' IN PLAM FUNCTION 'OPEN'
EDT5257    FEHLERCODE '(&00)' IN PLAM-FUNKTION 'OPEN'
```

Bedeutung
Die waehrend der Bearbeitung der OPEN-Anweisung aufgerufene PLAM-Funktion OPEN lieferte den Fehlercode (&00).
Die Anweisung wurde nicht ausgefuehrt.

```
EDT5258    FILE '(&00)' ALREADY EXISTS
EDT5258    DATEI '(&00)' EXISTIERT BEREITS
```

Bedeutung
Die in der @OPEN-Anweisung (Format 2) angegebene Datei (&00) existiert bereits. Die Anweisung wurde nicht ausgefuehrt.

```
EDT5259    CCS '(&00)' INCOMPATIBLE WITH TERMINAL
EDT5259    CCS '(&00)' UNVERTRAEGLICH MIT DER DATENSICHTSTATION
```

Bedeutung
Die Datei oder das Bibliothekselement, das eingelesen oder eroeffnet werden sollte, hatte das Codemerkmal (&00), oder in der Anweisung @CODENAME war der CCS Name (&00) angegeben worden. Es ist nicht moeglich, dieses Coded Character Set einzustellen, da die Datensichtstation nicht dazu faehig ist.
Fehlerschalter: EDT.

EDT5261 'DELETE' NOT PROCESSED. LIBRARY '(&00)' DOES NOT EXIST
EDT5261 'DELETE' NICHT AUSGEFUEHRT. BIBLIOTHEK '(&00)' EXISTIERT NICHT

EDT5263 ERROR CODE '(&00)' IN PLAM FUNCTION 'PUTA'
EDT5263 FEHLER CODE '(&00)' IN PLAM-FUNKTION 'PUTA'

Bedeutung
Die waehrend der Bearbeitung der Anweisung aufgerufene PLAM-Funktion PUTA lieferte den Fehlercode (&00).
Das Element wurde geschlossen, aber nicht zurueckgeschrieben.

EDT5266 LIBRARY '(&00)' LOCKED
EDT5266 BIBLIOTHEK '(&00)' GESPERRT

Bedeutung
Die in der Anweisung angegebene Bibliothek ist lesegeschuetzt.
Die Anweisung wurde nicht ausgefuehrt.

EDT5267 SPECIFIED LIBRARY '(&00)' DOES NOT EXIST
EDT5267 ANGEGEBENE BIBLIOTHEK '(&00)' EXISTIERT NICHT

EDT5268 MEMBER '(&00)' IS LOCKED
EDT5268 ELEMENT '(&00)' IST GESPERRT

Bedeutung
Das angegebene Element konnte nicht angesprochen werden, da es entweder geschuetzt oder bereits eroeffnet ist.

EDT5270 MEMBER '(&00)' IN LIBRARY '(&01)' NOT FOUND FOR UPDATE OPERATION
EDT5270 ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' ZUM AENDERN NICHT GEFUNDEN

Bedeutung
Das in der @OPEN-Anweisung angegebene Element (&00) der Bibliothek (&01) wurde nicht gefunden.
Die Anweisung wurde nicht ausgefuehrt.

Maßnahme
PLAM-Typ des Elementes ueberpruefen.

EDT5271 S-VARIABLE NOT FOUND FOR UPDATE
EDT5271 S-VARIABLE ZUM AENDERN NICHT GEFUNDEN
(B) Routing code: * Weight: 99

Bedeutung
Es wurde eine @SETVAR-Anweisung mit dem Operanden MODE=UPDATE eingegeben, die angegebene Variable war aber nicht definiert.
Fehlerschalter: EDT.

EDT5272 S-VARIABLE ALREADY DECLARED
EDT5272 S-VARIABLE SCHON DEKLARIERT
(B) Routing code: * Weight: 99

Bedeutung
Eine @SETVAR-Anweisung mit dem Operanden MODE=NEW wurde eingegeben, aber die SDF-P-Variable existiert schon.
Fehlerschalter: EDT.

EDT5273 MEMBER '(&00)' IN LIBRARY '(&01)' ALREADY EXISTS
EDT5273 ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' EXISTIERT BEREITS

Bedeutung
Das in der @OPEN-Anweisung angegebene Element (&00) in der Bibliothek (&01) existiert bereits. Es war aber der Operand MODE=NEW angegeben.
Die Anweisung wurde nicht ausgefuehrt.
Fehlerschalter: EDT.

EDT5274 S-VARIABLE NOT DECLARED
EDT5274 S-VARIABLE NICHT DEFINIERT
(B) Routing code: * Weight: 99

Bedeutung
Eine @GETVAR-, @GETLIST- oder @SETLIST-Anweisung konnte nicht ausgefuehrt werden, da die angegebene Variable nicht definiert ist.
Fehlerschalter: EDT.

EDT5275 'COPY' NOT POSSIBLE: MEMBER '(&00)' DOES NOT EXIST
EDT5275 'COPY' NICHT MOEGLICH: ELEMENT '(&00)' NICHT GEFUNDEN

Bedeutung
Die @COPY-Anweisung wurde nicht ausgefuehrt, da das in der Anweisung angegebene Element nicht existiert.

Maßnahme
PLAM-Typ des Elementes ueberpruefen.

EDT5278 FILE '(&00)' PROTECTED BY PASSWORD
EDT5278 DATEI '(&00)' DURCH PASSWORT GESCHUETZT

Maßnahme
Sich mit Dateieigentuemer in Verbindung setzen.

EDT5279 FILE '(&00)' LOCKED
EDT5279 DATEI '(&00)' GESPERRT

Bedeutung
Die in der Anweisung angesprochene Datei ist durch ACCESS=READ geschuetzt oder bereits geoeffnet.

EDT5281 FILE '(&00)' DOES NOT EXIST
EDT5281 DATEI '(&00)' EXISTIERT NICHT

EDT5282 FILE '(&00)' IS EMPTY OR LOCKED
EDT5282 DATEI '(&00)' LEER ODER GESPERRT

Bedeutung
Die in der Anweisung angegebene Datei hat Lastpage-Pointer 0.
Moegliche Ursachen sind:
- Die Datei ist leer.
- Die Datei ist SYSLST oder SYSOUT zugewiesen.

Maßnahme
Anweisung spaeter wiederholen.

EDT5283 ERROR CODE '(&00)' IN PLAM FUNCTION 'DELETE'
EDT5283 FEHLERCODE '(&00)' IN PLAM-FUNKTION 'DELETE'

Bedeutung
Beim Versuch, ein PLAM-Element zu loeschen, lieferte der PLAM-DELETE-Makro den Fehlercode (&00).

EDT5284 MEMBER '(&00)' DOES NOT EXIST
EDT5284 ELEMENT '(&00)' EXISTIERT NICHT

Bedeutung
Die @DELETE-Anweisung fuer das Element (&00) konnte nicht ausgefuehrt werden, da das angegebene Element nicht existiert.

Maßnahme
Anweisung mit richtigem Elementnamen und PLAM-Typ wiederholen.

EDT5285 'SHOW': PLAM ERROR CODE '(&00)'
EDT5285 'SHOW': PLAM-FEHLERCODE '(&00)'

Bedeutung
Bei Bearbeitung der @SHOW-Anweisung lieferte ein PLAM-Makro den Fehlercode (&00).

EDT5286 INVALID USER TYPE
EDT5286 UNGUELTIGER BENUTZER-TYP
(B) Routing code: * Weight: 99

Bedeutung
Das angesprochene Bibliothekselement ist nicht editierbar. Der angegebene Benutzertyp entspricht nicht einem der folgenden PLAM-Typen: S,M,P,J,D,X.
Fehlerschalter: EDT.

EDT5287 NO MEMBERS OF SPECIFIED TYPE OR LIBRARY IS EMPTY
EDT5287 KEINE ELEMENTE DES ANGEGEBENEN TYPS ODER DIE BIBLIOTHEK IST LEER

Bedeutung
Die @SHOW-Anweisung kann nicht ausgefuehrt werden, da entweder kein Element des angegebenen Typs existiert oder die Bibliothek leer ist.

EDT5289 JV LINK NAME NOT DEFINED
EDT5289 JV LINKNAME NICHT DEFINIERT

Bedeutung
Es wurde versucht, eine Jobvariable mittels ihres Linknamens anzusprechen, aber es war kein solcher Linkname vereinbart.
Fehlerschalter: EDT, DVS.

EDT5290 BUFFER TOO SMALL
EDT5290 PUFFER ZU KLEIN

Bedeutung
Fuer die Ausgabe eines Systemmakros stellt EDT einen Puffer bereit: z.B. fuer FSTAT 15 Speicherseiten und fuer STAJV 8 Speicherseiten. Dieser Puffer ist aber nicht gross genug, die Ausgabe zu fassen, daher wurde die Bearbeitung des Systemmakros mit einem Returncode abgewiesen.
Fehlerschalter: EDT.

Maßnahme
Die Anweisung (@FSTAT, @STAJV oder @ERAJV) mit einem teilqualifizierten Datei- oder Jobvariablennamen angeben, damit sich der Umfang der Ausgabe verringert, oder im Fall @SDFTEST die SDF-Optionen aendern.

EDT5291 SYSDTA EOF
EDT5291 SYSDTA EOF

Bedeutung
Beim Versuch, die naechste Anweisung von SYSDTA einzulesen, wurde ’Dateiende’ (EOF) erkannt.
Fehlerschalter: EDT.

Maßnahme
Fuer normale Beendigung die HALT-Anweisung verwenden.

EDT5293 REQM ERROR
EDT5293 REQM–FEHLER

Bedeutung
Fuer die Dateibearbeitung ist kein virtueller Speicher verfuegbar.

EDT5294 RELM ERROR
EDT5294 RELM-FEHLER

Bedeutung
Fehler bei der Freigabe von virtuellem Speicher.

EDT5300 INTERNAL EDT ERROR '(&00)'
EDT5300 INTERNER EDT-FEHLER '(&00)'

Bedeutung
Interner EDT-Laufzeitfehler.
(&00): Fehlercode.

Maßnahme
Systemkundendienst verstaendigen.

EDT5310 UFS FILE '(&00)' DOES NOT EXIST
EDT5310 UFS-DATEI '(&00)' EXISTIERT NICHT
(B) Routing code: * Weight: 99

Bedeutung
Die angegebene Datei (&00) aus dem POSIX-Dateisystem kann nicht bearbeitet werden, da sie nicht existiert.
Fehlerschalter: EDT.

EDT5311 UFS FILE ALREADY EXISTS
EDT5311 UFS DATEI EXISTIERT BEREITS
(B) Routing code: * Weight: 99

Bedeutung
Die Datei, die in einer @XOPEN- oder @XWRITE-Anweisung angegeben wurde, kann nicht mit MODE=NEW bearbeitet werden, da sie bereits im POSIX-Dateisystem vorhanden ist.
Fehlerschalter: EDT.

EDT5312 INVALID ACCESS TO UFS-FILE
EDT5312 UNGUELTIGER ZUGRIFF AUF UFS-DATEI
(B) Routing code: * Weight: 99

Bedeutung
Die Datei, die in einer @XOPEN-, @XCOPY- oder @XWRITE-Anweisung angegeben wurde, kann nicht eroeffnet werden, da ein lesender oder schreibender Zugriff nicht erlaubt wurde.
Fehlerschalter: EDT.

EDT5313 UNABLE TO CREATE UFS FILE '(&00)'
EDT5313 UFS DATEI '(&00)' KANN NICHT ERZEUGT WERDEN

Bedeutung
Die POSIX-Datei kann nicht mit MODE=NEW erzeugt werden, da ein Unterverzeichnis fehlt.
Fehlerschalter: EDT.

EDT5320 SDF: NO PROGRAM NAME FOR TEST OF STATEMENTS DEFINED
EDT5320 SDF: KEIN PROGRAMMNAME FUER TEST EINER ANWEISUNG DEFINIERT
(B) Routing code: * Weight: 99

Bedeutung
Der Anwender gab @SDFTEST PROGRAM ein oder markierte eine Bildschirmzeile, die mit '//' beginnt, mit der Kurzmarkierung t, es ist aber noch kein interner Programmname definiert.
Fehlerschalter: EDT.

Maßnahme
Anweisung @SDFTEST mit dem Operand PROGRAM=name eingeben oder einen internen Programmnamen mit der Anweisung @PAR SDF-PROGRAM=name definieren.

EDT5321 SDF: PROGRAM NAME UNKNOWN
EDT5321 SDF: PROGRAMMNAME UNBEKANNT
(B) Routing code: * Weight: 99

Bedeutung
Der interne Programmname, der in der Anweisung @SDFTEST PROGRAM verwendet werden sollte, ist in keiner aktuellen Syntaxdatei bekannt.
Fehlerschalter: EDT.

EDT5322 SDF: TEST OPERATION ABORTED
EDT5322 SDF: TEST OPERATION ABGEBROCHEN
(B) Routing code: * Weight: 99

Bedeutung
Die Abarbeitung von @SDFTEST oder der Kurzanweisung t wurde abgebrochen.
Moegliche Ursache: Mehr als 255 Fortsetzungszeilen.
Fehlerschalter: EDT.

EDT5340 S-VARIABLE EMPTY
EDT5340 S-VARIABLE LEER
(B) Routing code: * Weight: 99

Bedeutung
Die in der @GETVAR angegebene Variable war zwar definiert, hatte aber keinen Inhalt.

EDT5341 S-VARIABLE LONGER THAN 256 CHARACTERS
EDT5341 S-VARIABLE LAENGER ALS 256 ZEICHEN
(B) Routing code: * Weight: 99

Bedeutung
Die @GETVAR-Anweisung konnte nicht ausgefuehrt werden, das der angegebenen Variablen eine Zeichenkette laenger als 256 Zeichen zugewiesen ist.
Fehlerschalter: EDT.

EDT5342 WRONG TYPE OF S-VARIABLE
EDT5342 FALSCHER TYP DER S-VARIABLE
(B) Routing code: * Weight: 99

Bedeutung
Die @GETVAR- oder @SETVAR-Anweisung konnte nicht ausgefuehrt werden, da der Typ der angegebenen Variablen nicht mit dem Wert des Operanden an der rechte Seite des Gleichheitszeichens uebereinstimmt.
Fehlerschalter: EDT.

Maßnahme
Einer Ganzzahlvariablen kann nur der Wert einer SDF-P-Variablen vom Typ INTEGER zugewiesen werden und umgekehrt.

EDT5343 WRONG TYPE OF LIST ELEMENT
EDT5343 FALSCHER TYP VON LIST-ELEMENTEN
(B) Routing code: * Weight: 99

Bedeutung
Die Anweisung @GETLIST oder @SETLIST konnte nicht ausgefuehrt werden, da die Elemente der angegebenen Listenvariable nicht vom Typ STRING sind.
Fehlerschalter: EDT.

EDT5350 COMPARE RESULT CANNOT BE SHOWN
EDT5350 VERGLEICHSERGEBNIS KANN NICHT ANGEZEIGT WERDEN

Bedeutung
Die Ausgabedatei ist eine der zu vergleichenden Arbeitsdateien.
Fehlerschalter: EDT.

EDT5351 COMPARE OPERATION ABORTED
EDT5351 VERGLEICH ABGEBROCHEN

Bedeutung
Beim Bearbeiten der Anweisung @COMPARE (Format 2) trat ein nicht behebbarer Fehler auf, sodass der Vergleich abgebrochen werden musste.
Fehlerschalter: wird nicht gesetzt.

```
EDT5352    COMPARE OPERATION ABORTED, RENUMBER
EDT5352    VERGLEICH ABGEBROCHEN, NEU NUMERIEREN
```

Bedeutung
Die Verarbeitung der Anweisung @COMPARE (Format 2) wurde abgebrochen. Die letzte Stelle einer Zeilennummer, die beim Vergleich intern verwendet wird, ist ungleich 0. Die Arbeitsdateien muessen vor dem Vergleich neu numeriert werden.
Fehlerschalter: EDT.

```
EDT5353    UNRECOVERABLE FORMAT ERROR ON SCREEN DISPLAY
EDT5353    NICHT BEHEBBARER FORMAT-FEHLER BEI BILDSCHIRM-AUSGABE
```

Bedeutung
Fehlerschalter: wird nicht gesetzt.

```
EDT5354    STRUCTURE SYMBOL '(&00)' NOT FOUND
EDT5354    STRUKTURSYMBOL '(&00)' NICHT GEFUNDEN
```

Bedeutung
Das definierte Struktursymbol (&00) wurde in der angegebenen Zeile nicht gefunden. Das Positionieren wurde nicht durchgefuehrt.
(&00): Struktursymbol.

Maßnahme
Korrigierte Anweisung wiederholen.

```
EDT5356    K-LINE NOT COPIED BECAUSE OF TERMINAL CONTROL CHARACTERS
EDT5356    K-ZEILE NICHT UEBERNOMMEN, DA SIE BILDSCHIRMSTEUERZEICHEN ENTHAELT
```

Bedeutung
Die mit K markierte Zeile kann nicht in die Anweisungszeile uebernommen werden, da sie Bildschirmsteuerzeichen enthaelt.
Fehlerschalter: wird nicht gesetzt.

```
EDT5357    LINE DOES NOT EXIST
EDT5357    ZEILE EXISTIERT NICHT
```

Bedeutung
Im Format 2 der @CODE-Anweisung existiert die mit <zlnr> angegebene Zeile nicht.
Fehlerschalter: EDT.

```
EDT5358    LINE SHORTER THAN 256 BYTES
EDT5358    ZEILE KUERZER ALS 256 BYTE
```

Bedeutung
Im Format 1 der @CODE-Anweisung ist die mit <zlnr> angegebene Zeile kuerzer als 256 Byte.
Fehlerschalter: EDT.

EDT5359 MAXIMUM LINE NUMBER. COPY INCOMPLETE
EDT5359 MAXIMALE ZEILENNUMMER. KOPIE UNVOLLSTAENDIG

Bedeutung
Die groesste moegliche Zeilennummer wird ueberschritten, das Kopieren wird abgebrochen.
(Siehe Meldung: EDT5252 MAXIMALE ZEILENNUMMER.)
Fehlerschalter: EDT.

EDT5360 NO COPY. BUFFER EMPTY
EDT5360 KEINE KOPIE. PUFFER LEER

Bedeutung
Der Kopierpuffer ist leer. A/B/O kann nicht ausgefuehrt werden.
Fehlerschalter: wird nicht gesetzt.

EDT5362 <TEXT> SPECIFICATION ILLEGAL IN CURRENT STATEMENT
EDT5362 <TEXT>-EINGABE IN DIESER ANWEISUNG UNZULAESSIG

EDT5364 NO INSERT: MAXIMUM LINE NUMBER
EDT5364 KEIN EINFUEGEN: MAXIMALE ZEILENNUMMER

Bedeutung
Das Einfuegen von Datenzeilen mit einer Anweisung oder Kurzanweisung wird nicht ausgefuehrt, da die groesste moegliche Zeilennummer ueberschritten werden muesste.
(Siehe Meldung EDT5252 MAXIMALE ZEILENANZAHL.)
Fehlerschalter: EDT.

EDT5365 NO INSERT: RENUMBERING INHIBITED
EDT5365 KEIN EINFUEGEN: NEUNUMERIEREN VERBOTEN

Bedeutung
Die gewuenschten Datenzeilen koennen nicht eingefuegt werden, ohne bestehende Datenzeilen neu zu numerieren. Dies ist nicht moeglich, da in einer @PAR-Anweisung der Operand RENUMBER=NO angegeben war.
Fehlerschalter: EDT.

EDT5366 NO P-KEYS ON THIS TERMINAL
EDT5366 KEINE P-KEYS AUF DIESER DATENSICHTSTATION

Bedeutung
Die Anweisung @P-KEYS wurde an einer Datensichtstation 8161 oder 3270 aufgerufen.
Fehlerschalter: wird nicht gesetzt.

EDT5368 SECOND STATEMENT LINE NOT EMPTY
EDT5368 ZWEITE ANWEISUNGSZEILE NICHT LEER

Bedeutung
Bei gesplittetem Bildschirm wurde in der ersten Anweisungszeile SPLIT OFF oder @PAR mit Operand SPLIT=OFF eingegeben, obwohl die zweite Anweisungszeile
eine Anweisung enthaelt.
Fehlerschalter: wird nicht gesetzt.

EDT5371 TARGET FILE IS CURRENT WORK FILE
EDT5371 ZIELDATEI IST AKTUELLE ARBEITSDATEI

Bedeutung
Das Kommando konnte nicht ausgefuehrt werden, da die Zieldatei identisch ist mit der aktuellen Arbeitsdatei.

EDT5372 ENTRY DOES NOT EXIST IN SPECIFIED LIBRARY OR TASKLIB
EDT5372 ENTRY IN DER ANGEGEBENEN BIBLIOTHEK ODER TASKLIB NICHT GEFUNDEN

Bedeutung
Der angegebene ENTRY wurde nicht gefunden und konnte deshalb nicht nachgeladen werden.

Maßnahme
Anweisung korrigieren oder Bibliothek einrichten.

EDT5373 NO MORE THAN 5 'USE' ENTRIES ARE PERMITTED
EDT5373 MAXIMAL 5 'USE'-EINTRAEGE MOEGLICH

Bedeutung
Mit der @USE-Anweisung koennen maximal 5 Eintraege definiert werden.

Maßnahme
Einen Eintrag mit @USE-Anweisung loeschen und dann neu definieren.

EDT5375 NO 'USE' ENTRY DEFINED WITH SPECIFIED SYMBOL
EDT5375 KEIN 'USE'-EINTRAG MIT ANGEGEBENEM SYMBOL VORHANDEN

Bedeutung
Es wurde versucht, einen nicht definierten USE-Eintrag zu loeschen.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT5380 SOME JOB VARIABLES NOT ERASED
EDT5380 EINIGE JOBVARIABLEN NICHT GELOESCHT

Bedeutung
Es wurde versucht, alle JV zu loeschen, die einen angegebenen Teilstring im Jobvariablennamen besitzen, bzw. der Name wurde teilqualifiziert angegeben, doch das Loeschen einiger JV wurde vom System nicht zugelassen.
Moegliche Ursachen:
- Jobvariable ist nur lesend zugreifbar.
- Jobvariable ist als MONJV geschuetzt.

Fehlerschalter: EDT.

EDT5400 NOT SUPPORTED ON THIS INTERFACE
EDT5400 AN DIESER SCHNITTSTELLE NICHT UNTERSTUETZT

Bedeutung
Diese Form der Anweisung ist an dieser Schnittstelle nicht moeglich.

Maßnahme
Bitte die an dieser Schnittstelle erlaubte Form verwenden:
- z.B. @PAR HEX=ON statt HEX ON
  siehe @PAR-Anweisung
- Zur Bearbeitung von DVS-Dateien sollten die Anweisungen @READ, @WRITE, @GET oder @SAVE verwendet werden.

EDT5402 ENTER AT LEAST 2 CHARACTERS FOR '@DELETE'
EDT5402 MINDESTENS 2 ZEICHEN FUER '@DELETE' EINGEBEN

Bedeutung
Im F-Modus muessen in der @DELETE-Anweisung mindestens 2 Zeichen angegeben werden (@D oder DE), wenn sie ohne Operanden eingegeben wird.

EDT5409 STATEMENT ILLEGAL IN THIS ENVIRONMENT
EDT5409 ANWEISUNG IN DIESER UMGEBUNG NICHT ERLAUBT

EDT5410 UNDEFINED ERROR IN USER PROGRAM
EDT5410 UNDEFINIERTER FEHLER IM ANWENDER-PROGRAMM

Bedeutung
EDT erhaelt vom Anwenderprogramm einen undefinierten Returncode.
(&00): Meldung der externen Routine.

Maßnahme
Anwenderprogramm korrigieren.

```
EDT5419    (&00)
EDT5419    (&00)
```

Bedeutung
EDT erhaelt vom Anwenderprogramm einen undefinierten Returncode.
(&00): Meldung der externen Routine.

```
EDT5450    NATIONAL EDT: INTERNAL ERROR. RETURNCODE = X'(&00)' AT CCS (&01)
EDT5450    NATIONALER EDT: INTERNER FEHLER. FEHLERCODE = X'(&00)' BEI CCS (&01)
```
(B) Routing code: * Weight: 99

Bedeutung
Die Ausfu^hrung des Benutzer-CCS (&01) wurde an einer nationalen Sichtstation wegen eines internen Fehlers abgebrochen.
(&00): Fehlercode.

Maßnahme
Systemkundendienst verstaendigen.

```
EDT5500    STATEMENT PROCESSING INTERRUPTED BY /INTR
EDT5500    ABARBEITUNG DER ANWEISUNGSFOLGE DURCH /INTR ABGEBROCHEN
```
(B) Routing code: * Weight: 99

Bedeutung
Im F-Modus Dialog wurde eine Anweisungsfolge eingegeben. Die Abarbeitung wurde vom Benutzer durch die Eingabe von /SEND-MESSAGE TO=PROGRAM bzw. /INTR abgebrochen.
Fehlerschalter: wird nicht gesetzt.

Maßnahme
Der nicht verarbeitete Rest der Anweisungsfolge wird in der Kommandozeile ausgegeben.

```
EDT5991    RUNTIME ERROR IN EXTERNAL STATEMENT
EDT5991    LAUFZEIT-FEHLER IN EXTERNER ANWEISUNG
```
(B) Routing code: * Weight: 99

Bedeutung
Die externe Routine meldet einen Laufzeitfehler in der Abarbeitung der angegebenen Anweisung.

```
EDT5999    (&00)
EDT5999    (&00)
```
(B) Routing code: * Weight: 99

Bedeutung
Die externe Routine meldet einen Laufzeitfehler in der Abarbeitung der angegebenen Anweisung.
(&00): Meldung der externen Routine.

```
EDT8000    EDT TERMINATED
EDT8000    EDT NORMAL BEENDET
```

Bedeutung
EDT-Endemeldung bei normaler Programmbeendigung.

```
EDT8001    EDT TERMINATED ABNORMALLY
EDT8001    EDT ABNORMAL BEENDET
```

Bedeutung
EDT-Endemeldung bei abnormaler Programmbeendigung (Programmfehler).

Maßnahme
Systemverwalter verstaendigen.

```
EDT8002    (&00) TO EDT UNSUCCESSFULLY. RETURNCODE = X'(&01)'
EDT8002    FEHLER BEI (&00) DES EDTS. RETURNCODE = X'(&01)'
```

Bedeutung
Beim Nachladen des Moduls EDT trat ein Fehler auf. Der Makro (&00) lieferte den Fehlercode (&01).
Fehlerschalter: nicht gesetzt.

```
EDT8003    NO VIRTUAL MEMORY AVAILABLE
EDT8003    KEIN VIRTUELLER SPEICHER VERFUEGBAR
```

Maßnahme
Speicher freigeben.

```
EDT8005    ERROR ON EDT INITIALIZATION
EDT8005    FEHLER BEI ERSTAUFRUF DES EDT
```

Bedeutung
Fehlerschalter: wird nicht gesetzt.

```
EDT8006    ERROR ON INSTALLATION OF EDT
EDT8006    FEHLERHAFTE INSTALLATION VON EDT
```

Bedeutung
Der Grossmodul EDTF aus der Nachladebibliothek des EDTs kann nicht aufgerufen werden, da er eine unzulaessige Versionnummer besitzt.

Maßnahme
EDT-Installation ueberpruefen und korrigieren.

EDT8100 EDT INTERRUPTED BY USER
EDT8100 EDT VOM BENUTZER UNTERBROCHEN
(B) Routing code: * Weight: 99

Bedeutung
Diese Meldung dient als Information fuer SDF-P-Prozeduren.
EDT ist geladen, wurde aber durch @SYSTEM (ohne Operand) oder durch K2 unterbrochen.
Fehlerschalter: nicht gesetzt.

EDT8101 USER TERMINATED EDT ABNORMALLY
EDT8101 EDT VOM ANWENDER ABNORMAL BEENDET
(B) Routing code: * Weight: 99

Bedeutung
Der Anwender beendete den EDT mit der Anweisung '@HALT ABNORMAL'.

EDT8200 STXIT ROUTINE FOR RUNOUT ACTIVATED
EDT8200 STXIT ROUTINE FUER RUNOUT AKTIVIERT
(B) Routing code: * Weight: 99

Bedeutung
Das Ende der Programmlaufzeit wurde erreicht, deshalb wurde EDT beendet.

EDT8292 UNRECOVERABLE RDATA ERROR. PROGRAM ABORTED
EDT8292 NICHT BEHEBBARER RDATA-FEHLER. PROGRAMM-ABBRUCH

Maßnahme
Systemverwalter verstaendigen.

EDT8300 INTERNAL EDT ERROR '(&00)'
EDT8300 INTERNER EDT FEHLER '(&00)'

Bedeutung
Programmfehler im EDT.

Maßnahme
Systemverwalter verstaendigen.

EDT8900 NO VIRTUAL ADDRESS SPACE AVAILABLE
EDT8900 KEIN VIRTUELLER ADRESSRAUM VERFUEGBAR

Bedeutung
Wenn der EDT geladen wird, fordert er fuer einen 4 Seiten langen Daten- und Variablenbereich ueber REQM Platz im virtuellen Adressraum an. Laeuft der REQM auf einen Fehler, so wird diese Meldung ausgegeben und der EDT beendet. Wird der EDT als Unterprogramm aufgerufen, so wird mit dem Returncode X'10' im rechtsbuendigen Byte des Registers 15 zurueckgekehrt.

```
EDT8901    ERROR RECOVERY FAILED. EDT ABORTED
EDT8901    FEHLERBEHANDLUNG GESCHEITERT. EDT ABGEBROCHEN
```

Bedeutung
Die Unterbrechungs-Fehlerbehandlung nach einem Datenfehler konnte nicht erfolgreich abgeschlossen werden.
Fehlerschalter: wird nicht gesetzt.

```
EDT8902    '@HALT' STATEMENT PROCESSED
EDT8902    '@HALT'-ANWEISUNG AUSGEFUEHRT
```

Bedeutung
Ein Datenfehler oder ein nicht behebbarer Fehler trat in einem EDT-Stapelprozess auf.
Fehlerschalter: EDT.

```
EDT8910    EDT INTERRUPTED AT LOCATION '(&00)', INTERRUPT WEIGHT=(&01)
EDT8910    EDT AN DER ADRESSE '(&00)' UNTERBROCHEN, UNTERBRECHUNGSGEWICHT=(&01)
```
(B) Routing code: * Weight: 99

Bedeutung
Durch das Ereignis "Programmueberpruefung" oder "Nicht behebbarer Programmfehler" trat an der Adresse (&00) eine Programmunterbrechung auf.
Die naehere Ursache wird durch das Unterbrechungsgewicht (&01) angegeben.
Fehlerschalter: wird nicht gesetzt.

8 Installationshinweise

Dieses Kapitel betrifft nur die Systemverwaltung.

Installation des EDT in BS2000 V10

Installationskennung : defluid

Wenn das Subsystem EDTCON nicht aktiv ist, muß die Modulbibliothek $.SYSLNK.EDT.165 auf $EDTLIB kopiert werden. LMS-Objektkorrekturen müssen in die $.SYSLNK.EDT.165 eingebracht werden, anschließend muß die Datei auf $EDTLIB kopiert werden.

Installation des EDT in BS2000/OSD-BC V1.0

Installationskennung : defluid

Wenn das Subsystem ACS aktiv ist, ist ein Kopieren von $.SYSLNK.EDT.165 auf $EDTLIB nicht notwendig.

Der Inhalt des Alias-Katalogs in der Datei SYSACF.EDT.165 muß in den system-globalen Alias-Katalog übernommen werden. Im Alias-Katalog wird zum Dateinamen $.SYSLNK.EDT.165 der Alias-Name $EDTLIB definiert.

Ab BS2000/OSD-BC V2.0 wird von EDT IMON unterstützt
Installation des EDT mit IMON

Installationskennung : beliebig

Das Subsystem EDTCON wird gestartet:
– keine zusätzliche Maßnahmen sind erforderlich.

Das Subsystem EDTCON wird nicht gestartet:
– das Subsystem ACS muß aktiv sein.
– In dem Alias-Katalog-Eintrag in der SYSACF-Datei muß der Dateiname $.SYSLNK.EDT.165 (wenn anders installiert) durch den Installationsnamen ersetzt werden, und dann muß SYSACF in den system-globalen Alias-Katalog gemischt werden.

8.1 Produktbestandteile

Dateiname	Funktion	BS2000-Version
EDT	Phase für EDT	ab V10
SYSLNK.EDT.165	EDT-Modulbibliothek	ab V10
SYSLIB.EDT.165	Benutzer-Makrobibliothek	ab V10
SYSMES.EDT.165	Systemmeldungsdatei (MSGMAKER)	ab OSD-BC V2.0
SYSMSA.EDT.165	Systemmeldungsdatei (Meldungen)	V10, OSD-BC 1.0
SYSMSR.EDT.165	Systemmeldungsdatei (Help-Texte)	V10, OSD-BC 1.0
SYSMSV.EDT.165	Systemmeldungsdatei (Source)	V10, OSD-BC 1.0
SYSSSD.EDT.165	Subsystemdeklarationen für DSSM	V10
SYSSSC.EDT.165.110	Subsystemdeklarationen für SSCM	OSD-BC V1.0
SYSSSC.EDT.165.112	Subsystemdeklarationen für SSCM	ab OSD-BC V2.0
SYSSII.EDT.165	Struktur- und Installationsinformation	ab OSD-BC V2.0
SYSRMS.EDT.165	Korrekturdepot für RMS	ab V10
SYSREP.EDT.165	REP-Datei	ab V10
SYSFGM.EDT.165.D	Freigabe-Mitteilung (deutsch)	ab V10
SYSFGM.EDT.165.E	Freigabe-Mitteilung (englisch)	ab V10
SYSSDF.EDT.165	Syntaxdatei für SDF (START-EDT-Kommando)	ab OSD-BC V2.0
SYSACF.EDT.165	ALIAS-Katalog	ab OSD-BC V1.0
SINPRC.EDT.165	Prozedur zum Installieren einer Privatversion	ab V10

SYSLNK.EDT.165

Die Modulbibliothek SYSLNK.EDT.165 enthält die Module:

Modul	Funktion
EDTSTRT	Treiber für START-EDT
EDT	Großmodul für EDT
EDXCODIR	Nachlademodul für CODE-Anweisung
EDXPKEY	Nachlademodul für PKEY-Anweisung
EDXUFS	Nachlademodul für POSIX-Zugriffe
IEDTGLE	Bindemodul für EDT als Unterprogramm
CODTAB	Codiertabelle für CODE-Anweisung
EDTSSLNK	DSSM-Nachlademodul
EDTCON	Anschlußmodul für EDT
EDCNATA	Nachlademodul für nationale Anwendung

Das Modul EDTSSLNK ist in ladbarer Form in SYSLNK.EDT.165 enthalten und hat den folgenden Aufbau:

```
EDTSSLNK    START   0,PUBLIC,READ
            DC      V(EDT)
            DC      V(EDTF)
            DC      V(EDTC)
            DC      V(EDTXS)
            DC      V(EDTFXS)
            DC      V(EDTCXS)
            DC      V(EDTGLE)
            END
```

SYSSII.EDT.165

Diese Datei enthält die Struktur und Installationsinformation. Den Produktbestandteilen werden logische Namen (Logical ID) zugeordnet, mit deren Hilfe über IMON der aktuelle Installationsort bestimmt werden kann.

IMON wird vom EDT erst ab BS2000/OSD-BC V2.0 unterstützt.

Release Item	Logical ID
EDT	SYSPRG
SYSLNK.EDT.165	SYSLNK
SYSLIB.EDT.165	SYSLIB
SYSMES.EDT.165	SYSMES
SYSMSA.EDT.165	SYSMSA
SYSMSR.EDT.165	SYSMSR
SYSMSV.EDT.165	SYSMSV
SYSSSD.EDT.165	SYSSSD
SYSSSC.EDT.165.110	SYSSSC.110
SYSSSC.EDT.165.112	SYSSSC.112
SYSSII.EDT.165	SYSSII
SYSRMS.EDT.165	SYSRMS
SYSREP.EDT.165	SYSREP
SYSFGM.EDT.165.D	SYSFGM.D
SYSFGM.EDT.165.E	SYSFGM.E
SYSPPS.EDT.165	SYSPPS
SYSSDF.EDT.165	SYSSDF
SINPRC.EDT.165	SINPRC
*DUMMY.EDTSTART	SYSDAT.EDTSTART

SINPRC.EDT.165

SINPRC.EDT.165 enthält eine Prozedur, mit deren Hilfe auf einer beliebigen Benutzerkennung eine Privatversion installiert werden kann. Eine Privatversion sollte nur zu Testzwekken installiert werden und nicht zur Koexistenz von zwei EDT-Versionen.

Nähere Informationen zur Prozedur SINPRC.EDT.165 siehe Freigabemitteilung.

8.2 Start-Prozedur EDTSTART

Ab BS2000/OSD-BC V2.0 kann für jede koexistenzfähige EDT-Version über IMON eine eigene, auf allen Kennungen geltende Start-Prozedur installiert werden. Der Installationsort der Prozedurdatei kann frei gewählt werden.

Für diese Datei wird in der SYSSII-Datei die logische Identifikation SYSDAT.EDTSTART definiert.

Der Installationsdateiname kann durch das Kommando SET-INSTALLATION-PATH dem Installations-Monitor bekanntgegeben werden. Der EDT holt diese Information mit der IMON-Funktion GETINSP, und der definierte Pfad wird anstelle von $EDTSTART eingesetzt.

Falls der logischen Identifikation SYSDAT.EDTSTART keine Datei zugewiesen ist, verwendet der EDT die Startup-Prozedur $EDTSTART.

8.3 EDT als Subsystem

Der EDT besteht aus 2 Subsystemen:

– EDT und
– EDTCON.

Das Subsytem EDT (bestehend aus dem Modul EDT) kann im oberen Adressraum geladen werden.

Das Subsystem EDTCON (bestehend aus den Modulen EDTCON, IEDTGLE und EDTSSLNK) wird im unteren Adressraum geladen.

Soll der EDT als Haupt- oder Unterprogramm im 24-Bit-Adressierungsmodus ablaufen, wird vom Modul EDTCON entweder die Verbindung zu einem unten geladenen EDT hergestellt oder der EDT privat im unteren Adressraum nachgeladen.

Ab BS2000/OSD-BC V2.0 muß EDTCON zuerst mit START-SUBSYSTEM SUBSYSTEM-NAME=EDTCON,SYNCHRONOUS=*YES gestartet werden. Dann kann EDT mit START-SUBSYSTEM SUBSYSTEM-NAME=EDT gestartet werden.

In niedrigeren Betriebssystemversionen ist die Startreihenfolge egal.

8.4 Installationshinweise für das Modul CODTAB

Das Modul CODTAB enthält folgende Codetabelle:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0					␣	&	-									0
1							/		a	j			A	J		1
2									b	k	s		B	K	S	2
3									c	l	t		C	L	T	3
4									d	m	u		D	M	U	4
5									e	n	v		E	N	V	5
6									f	o	w		F	O	W	6
7								~	g	p	x		G	P	X	7
8									h	q	y		H	Q	Y	8
9									i	r	z		I	R	Z	9
A					‘	!	^	:								
B					.	$	,	#	[		{					
C					<	*	%	@	\		|					
D					(	)	_	’	]		}					
E					+	;	>	=								
F							?	„								

Diese Codetabelle setzt bei der Ausgabe der Nachricht auf die Datensichtstation die Druckverschlüsselung von ä, ö, ü, Ä, Ö, Ü und ß in Codes um, die an den häufigsten Datensichtstations-Tastaturen für die entsprechenden Zeichen gelten. Aus dieser Ausgabetabelle wird die Eingabetabelle abgeleitet, um beim Einlesen der Nachricht diese Zeichen wieder in die Druckverschlüsselung von ä, ö, ü, Ä, Ö, Ü, ß umsetzen zu können.

Es wird davon ausgegangen, daß für die Umlaute und das Zeichen ß folgende Verschlüsselungen für die Tastatur bzw. den Drucker gelten:

Tastatur	Zeichen		Datei
X'FB'	ä	{	X'AB'
X'4F'	ö	|	X'AC'
X'FD'	ü	}	X'AD'
X'BB'	Ä	[	X'8B'
X'BC'	Ö	\	X'8C'
X'BD'	Ü	]	X'8D'
X'FF'	ß	–	X'67'

Einzelne Zeichen dieser Tabelle können folgendermaßen erweitert bzw. geändert werden:

Z.B. sollen runde Klammern in eckige Klammern auf dem Drucker umgesetzt werden. Dabei wird angenommen, daß der „runden Klammer auf“ auf der Tastatur die Verschlüsselung X’4D’ und der „runden Klammer zu“ die Verschlüsselung X’5D’ entspricht. Dem Zeichen „eckige Klammer auf“ auf der Druckerstation entspricht X’63’ und der „eckigen Klammer zu“ X’64’.

Mit der REP-Anweisung des LMR kann nun geändert werden:

```
REP 63, X'4D' OBJMOD=CODTAB
REP 64, X'5D' OBJMOD=CODTAB
REP 4D, X'07' OBJMOD=CODTAB
REP 5D, X'07' OBJMOD=CODTAB
```

Um Mehrfachbelegungen zu verhindern, muß der Benutzer die Positionen X’4D’ und X’5D’ in der Codetabelle mit Schmierzeichen (X’07’) belegen. Eine Überprüfung auf Mehrfachbelegung findet nicht statt.

Sind die Änderungen umfangreicher, so ist es u.U. sinnvoll, eine eigene Codetabelle zu erstellen und das ausgelieferte Modul CODTAB in der $EDTLIB durch ein eigenes Modul mit gleichem Namen zu ersetzen. Es ist also ein Source zu erstellen, der mit dem ASSEMBLER zu übersetzen ist. Das so erzeugte Bindemodul ist mit dem Programm LMS in die $EDTLIB aufzunehmen.

Beispiel für das Quellprogramm

```
CODTAB  CSECT
CODTAB  AMODE ANY
CODTAB  RMODE ANY
        TITLE 'EDTF.V2 *** CODTAB  *** '
        SPACE
*MODUL ENTHAELT LEDIGLICH UMSETZ-TABELLEN
*
*TASTATUR * ZEICHEN * DATEI
*****************************
* X'FB'   *  (ä)   { *  X'AB'
* X'4F'   *  (ö)    *  X'AC'
* X'FD'   *  (ü)   } *  X'AD'
* X'BB'   *  (Ä)   [ *  X'8B'
* X'BC'   *  (Ö)   \ *  X'8C'
* X'BD'   *  (Ü)   ] *  X'8D'
* X'FF'   *  (ß)   ~ *  X'67'
*
*****************************
        EJECT
        SPACE
*   UMSETZTABELLE FUER "CODIER-MODUS"  - EINGABEBEARBEITUNG
*   UMSETZTABELLE FUER "LOWER ON " MODUS
*   DIE UMSETZ-TABELLE FUER LOWER OFF WIRD AUS DIESER TABELLE
*   ABGELEITET.
CODTAB1 DC    X'000707070707070707070707070707'
        DC    X'070707070707070707070707070707'
        DC    X'070707070707070707070707070707'
        DC    X'070707070707070707070707070707'
        DC    X'400707070707070707074A4B4C4D4E07'
        DC    X'500707070707070707075A5B5C5D5E5F'
        DC    X'606107070707070FF07076A6B6C6D6E6F'
        DC    X'070707070707070707077A7B7C7D7E7F'
        DC    X'078182838485868788907BBBCBD07'
        DC    X'079192939495969798990707070707'
        DC    X'0707A2A3A4A5A6A7A8A907FB4FFD07'
        DC    X'070707070707070707070707070707'
        DC    X'07C1C2C3C4C5C6C7C8C9070707070707'
        DC    X'07D1D2D3D4D5D6D7D8D9070707070707'
        DC    X'0707E2E3E4E5E6E7E8E9070707070707'
        DC    X'F0F1F2F3F4F5F6F7F8F9070707070707'

        SPACE 3
        END   CODTAB
```

Literatur

[1] **SDF** (BS2000/OSD)
Einführung in die Dialogschnittstelle SDF
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an BS2000/OSD-Anwender.
Inhalt
Das Handbuch beschreibt die Dialog-Eingabe von Kommandos und Anweisungen im SDF-Format. Ein Schnelleinstieg mit leicht nachvollziehbaren Beispielen und weitere umfangreiche Beispiele erleichtern die Anwendung. SDF-Syntaxdateien werden erklärt. Ab SDF V4.0A kann der Anwender das Kommandogedächtnis zur erneuten Eingabe nutzen. SHOW-SDF-OPTIONS und SHOW-SYNTAX-VERSIONS bieten die Ausgabe in S-Variablen. Ein eigener Abschnitt erläutert die Anwendung.

[2] **BS2000/OSD-BC**
Kommandos
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.
Inhalt
Das Handbuch enthält die BS2000/OSD-Kommandos (Grundausbau und ausgewählte Produkte). Die Einleitung gibt Hinweise zur Kommandoeingabe. Die Kommandobeschreibung zeigt u.a. die Privilegien zur Nutzung der Kommandos. Mit SDF-P können Anwender bei SHOW-Kommandos die Ausgabe in strukturierte S-Variablen nutzen.

[3] **Assembler** (BS2000)
Beschreibung

Zielgruppe
Assembler-Anwender im BS2000
Inhalt
– Assembler-Charakteristik
– Assemblersprache
– Makrosprache
– Handhabung des Assemblers
– Meldungen bzw. Fehlermeldungen
– Flags
– Beschreibung des Assembler-Diagnoseprogramms ADIAG

[4] **ASSEMBH** (BS2000)
Benutzerhandbuch

Zielgruppe
Assembler-Anwender im BS2000
Inhalt
– Aufruf und Steuerung des ASSEMBH
– Übersetzen, Binden, Laden und Starten
– Eingabequellen und Ausgaben des ASSEMBH
– Laufzeitsystem, Listenausgabe der strukturierten Programmierung
– Sprachverknüpfungen
– Assembler-Diagnoseprogramm ASSDIAG
– Dialogtesthilfe AID
– Meldungen des ASSEMBH
– Format der Assemblerbefehle

[5] **BS2000/OSD-BC**
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an alle BS2000/OSD-Assembler-Programmierer.
Inhalt
Das Handbuch enthält eine Zusammenstellung der Makroaufrufe an den Ablaufteil, die ausführliche Beschreibung jedes Makroaufrufs mit Hinweisen und Beispielen - einschließlich der Jobvariablen-Makros - sowie einen ausführlichen allgemeinen Lernteil.
Im Lernteil werden u.a. Serilisation, Ereignissteuerung, Intertaskkommunikation und Contingencies beschrieben.

[6] **XHCS** (BS2000/OSD)
8-bit-Code-Verarbeitung im BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie Systemverwalter, Anwender, die von EHCS auf XHCS umstellen.
Inhalt
XHCS (Extended Host Code Support) ist ein Softwareprodukt des BS2000/OSD. Es ermöglicht Ihnen erweiterte Zeichensätze bei 8-bit-Datenstationen zu nutzen. XHCS ist die zentrale Informationsquelle über die codierten Zeichensätze im BS2000/SD. XHCS löst EHCS ab.

[7] **JV** (BS2000/OSD)
Jobvariablen
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich sowohl an den nichtprivilegierten als auch privilegierten BS2000/OSD-Anwender.
Inhalt
Es beschreibt die Anwendung des Software-Produkts JV (Jobvariablen). Es enthält die Beschreibungen der Kommandos und Makros zur Verwaltung der JVs und zur bedingungsabhängigen Auftragssteuerung.

[8] **SDF-P** (BS2000/OSD)
Programmieren in der Kommandosprache
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an BS2000-Anwender und Systemverwalter.
Inhalt
SDF-P ist eine strukturierte Prozedursprache im BS2000. Nach einer Einführung werden Kommandos, Funktionenen und Makros ausführlich beschrieben.

[9] **EDT-Unterprogrammschnittstellen** (BS2000/OSD)
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an erfahrene EDT-Anwender und Programmmierer.
Inhalt
Das Handbuch beschreibt die Unterprogrammschnittstelle des EDT, den Aufruf des EDT als Unterprogramm, das Einbinden der EDT-Funktionalität in eigene Programme.
Aufbau und Generierung der Kontrollblöcke, Erstellen von externen Anweisungsroutinen, Aufruf eines Benutzerprogramms aus dem EDT.

[10] **EDT-Anweisungsformate** (BS2000/OSD)
Tabellenheft

Zielgruppe
Das Tabellenheft wendet sich an EDT-Anwender.
Inhalt
Das Tabellenheft enthält alle Anweisungen des EDT geordnet nach Funktionsgruppen und alphabetisch geordnet mit den Anweisungsformaten.

[11] **EDT-ARA** (BS2000/OSD)
Additionel Information for Arabic
User Guide

Target group
The manual addresses EDT users wishing to edit Arabic texts.
Contens
The manual describes how to create, delete, update, insert and copy bilingual records or parts thereof.
With EDT-ARA it is possible to replace Latin strings with Arabic strings and vice versa, to prefix or suffix records in one script with strings in the other script etc.

[12] **EDT-FAR** (BS2000/OSD)
Additionel Information for Farsi
User Guide

Target group
The manual addresses EDT users wishing to edit Farsi texts.
Contens
The manual describes how to create, delete, update, insert and copy bilingual records or parts thereof.
With EDT-FAR it is possible to replace Latin strings with Farsi strings and vice versa, to prefix or suffix records in one script with strings in the other script etc.

[13] **LMS** (BS2000/OSD)
SDF-Format
Benutzerhandbuch

Zielgruppe
BS2000-Anwender
Inhalt
Beschreibung der Anweisungen zum Erstellen und Verwalten von PLAM-Bibliotheken und darin enthaltenen Elementen.
Häufige Anwendungsfälle werden an Hand von Beispielen erklärt.

[14] **POSIX** (BS2000/OSD)
Grundlagen für Anwender und Systemverwalter
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an BS2000-Systemverwalter, POSIX-Verwalter und POSIX-Benutzer.
Inhalt
Inhalt des Handbuchs: Einführung und Arbeiten mit POSIX; BS2000-Softwareprodukte im Umfeld von POSIX; POSIX-Subsystem installieren, starten, steuern und beenden; POSIX-Benutzer durch das BS2000 verwalten.

[15] **POSIX** (BS2000/OSD)
Kommandos
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an alle Benutzer der POSIX-Shell.
Inhalt
Das Handbuch ist ein Nachschlagewerk. Es beschreibt das Arbeiten mit der POSIX-Shell sowie die Kommandos der POSIX-Shell in alphabetischer Reihenfolge.

Bestellen von Handbüchern

Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriften-verzeichnis* der Siemens Nixdorf Informationssysteme AG. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an Ihre zuständige Geschäftsstelle. Dort können Sie auch die Handbücher bestellen.

Stichwörter

A

C

F

M

T

Inhalt

EDT V16.5A (BS2000/OSD)

Anweisungen

Zielgruppe
EDT-Einsteiger und EDT-Anwender
Inhalt
Bearbeiten von SAM- und ISAM-Dateien und Elementen aus Programm-Bibliotheken und POSIX-Dateien.

Ausgabe: November 1994

Datei: EDT.PDF