EDT V17.0A Unicode-Modus

Anweisungen

Ausgabe Dezember 2007

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@fujitsu-siemens.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2000

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2000 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Der EDT ist der Dateieditor des BS2000, mit dem BS2000-Dateien der Formate SAM und ISAM sowie textartige Bibliothekselemente und POSIX-Dateien auf komfortable Weise erstellt und bearbeitet werden können.

Die Durchführung der beim Editieren häufig vorkommenden Arbeiten, wie z.B. Löschen, Ändern, Einfügen und Kopieren von Sätzen und Zeichen, Suchen nach Sätzen mit bestimmten Zeichenfolgen, Ausgeben von Sätzen usw. werden durch leistungsfähige und dennoch leicht erlernbare Anweisungen unterstützt.

Der EDT V17.0A kann in einem erweiterten Unicode-Modus und einem V16.6-kompatiblen Kompatibilitäts-Modus betrieben werden.

- Im Unicode-Modus kann der EDT V17.0A in Unicode und anderen Zeichensätzen codierte Dateien bearbeiten. Dem Anwender wird dabei eine komfortable Unterstützung geboten, dazu zählen u.a. die Möglichkeit, unterschiedlich codierte Dateien in verschiedenen Arbeitsdateien des EDT gleichzeitig zu bearbeiten sowie die Aufhebung der Begrenzung der Zeilenlänge (bisher 256 Zeichen). Der EDT kann beim Lesen aus und Schreiben in Dateien alle von DVS und LMS angebotenen Satzlängen verarbeiten. Bei POSIX-Dateien kann er Zeilen mit einer Maximallänge von 32768 Zeichen verarbeiten.

  Die interne Verwendung einer Unicode-Darstellung im EDT hat zur Konsequenz, dass alle Schnittstellen, an denen der Anwender bisher direkten Zugriff auf die internen Daten des EDT hatte, nicht kompatibel bleiben können. Dies trifft auf die alte L-Modus-Unterprogramm-Schnittstelle, auf die bisherige @RUN-Schnittstelle und auf den Locate-Mode der IEDTGLE-Schnittstelle zu. Diese Schnittstellen können daher im Unicode-Modus nicht mehr verwendet werden.

- Der Kompatibilitäts-Modus bietet die volle Funktionalität des EDT V16.6B mit nur geringfügigen Erweiterungen.

Obwohl der EDT als Dialogprogramm konzipiert ist, kann er auch im Stapelbetrieb Dateien und Bibliothekselemente bearbeiten.

Dateibearbeitungen, die häufig in gleicher oder ähnlicher Form auszuführen sind, lassen sich mit EDT-Prozeduren programmieren.

Der EDT kann andere Programme als Unterprogramm aufrufen und kann selbst von einem Benutzerprogramm als Unterprogramm aufgerufen werden.

1.1 Konzept der EDT-Dokumentation

In den Handbüchern

- EDT V17.0A Unicode-Modus Anweisungen
- EDT V17.0A Unicode-Modus Unterprogrammschnittstelle

wird der Unicode-Modus des EDT beschrieben. Der Kompatibilitäts-Modus wird in den Handbüchern

- EDT V16.6B Anweisungen
- EDT V16.6A Unterprogrammschnittstelle

beschrieben.

Zusätzlich enthält das Handbuch EDT V17.0A Anweisungen noch einen Abschnitt, in dem die Erweiterungen des Kompatibilitäts-Modus gegenüber dem EDT V16.6B beschrieben sind.

Die Handbücher zu den EDT-Anweisungen beschreiben grundlegende Konzepte des EDT im jeweiligen Modus und dienen als Nachschlagwerk für die zahlreichen Anweisungen des EDT.

Die Handbücher zu den Unterprogrammschnittstellen beschreiben, wie Benutzerprogramme programmiert werden können, die vom EDT aufgerufen werden können bzw. die den EDT als Unterprogramm aufrufen wollen. Sie können nur in Verbindung mit den Handbüchern zu den EDT-Anweisungen sinnvoll genutzt werden.

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 das Handbuch zu den EDT-Unterprogrammschnittstellen an den Kreis der erfahrenen EDT-Anwender und Programmierer, die den EDT in eigene 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 9 „Anweisungen des EDT“ mit den Beschreibungen aller EDT-Anweisungen ein notwendiges Nachschlagewerk darstellt. Der EDT-Anwender, der selber EDT-Prozeduren schreiben oder bestehende EDT-Prozeduren warten will, findet im Abschnitt „EDT-Prozeduren“ einen wertvollen Einstieg in das Prozedurenschreiben 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.

In einem gesonderten Kapitel werden die Erweiterungen des Kompatibilitäts-Modus und sein Zusammenspiel mit dem Unicode-Modus dargestellt.

Dieses Handbuch enthält im Einzelnen folgende Themen:

– **Geänderte und neue Funktionalität im EDT V17.0A**

  Wesentliche Änderungen und Neuerungen im EDT V17.0A.
  Einführung in die neue Funktionalität.

– **Konzepte des EDT**

  Grundlegende Konzepte und Mechanismen, die im EDT verwendet werden.
  Unter anderem die Handhabung von Arbeitsdateien, der Umgang mit Zeilen, Zeichensätzen und EDT-Variablen sowie die Anwendung von EDT-Prozeduren.

– **Ablauf des EDT**

  Startkommando des EDT. Starten, Unterbrechen und Beenden des EDT-Laufs.
  Überwachen des EDT-Laufs, Ein-Ausgabe, Auftragsschalter sowie Zugriffsschutz.

– **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: Zeilenorientiertes Arbeiten.

– **Dateibearbeitung**

  Bearbeiten aller vom EDT unterstützten Dateitypen: ISAM-, SAM-, POSIX-Dateien und Bibliotheken.

– **Anweisungen des EDT**

  Übersicht der Anweisungen thematisch geordnet. Darstellung der Meta-, Anweisungs- und Operandensyntax.
  Anweisungen des EDT in alphabetischer Reihenfolge mit zahlreichen Beispielen.

  Bei vielen Anweisungen wurde , im Vergleich zum EDT V16.6B-Handbuch, die Einteilung und Benennung der Formate sinnvoller gestaltet. Dies sind aber nur Darstellungsänderungen, die keine technischen Unterschiede beinhalten.

– **Kompatibilitäts-Modus**

  Beschreibung der Erweiterungen des Kompatibilitäts-Modus und sein Zusammenspiel mit dem Unicode-Modus. Neue Anweisungen des Kompatibilitäts-Modus.

– **Meldungen des EDT**

  Auflistung aller Meldungen des EDT mit ihren Bedeutungen und Maßnahmen.

– **Logistik**

  Voraussetzungen und Verfahren, um den EDT V17.0A zu installieren und in Betrieb zu nehmen.

2 Geänderte und neue Funktionalität im EDT V17.0A

Der EDT wird mit V17.0A um wesentliche Funktionen erweitert.
So wird die Verarbeitung von in Unicode codierten Dateien unterstützt und die Begrenzung der Satzlänge auf 256 Zeichen aufgehoben.
Im folgenden wird eine kurze Einführung in die geänderte und neue Funktionalität gegeben.

2.1 Einführung zu den Betriebsmodi des EDT

Die wesentliche Neuerung des EDT V17.0A gegenüber den Vorgängerversionen ist die Unterstützung der Bearbeitung von in Unicode codierten Dateien. Die beiden folgenden Ziele waren dabei gleichberechtigt zu berücksichtigen:

- Dem Anwender, der in Unicode codierte Dateien bearbeiten will, soll eine komfortable Unterstützung geboten werden. Dazu zählt u.a. die Möglichkeit unterschiedlich codierte Dateien in verschiedenen Arbeitsdateien des EDT gleichzeitig zu bearbeiten sowie die Aufhebung der Begrenzung der Satzlänge (bisher 256 Zeichen).
- Dem Anwender, der seine alten, in 7-Bit- oder 8-Bit-Zeichensätzen codierten Dateien wie bisher bearbeiten will, sollen zu EDT V16.6B kompatible Funktionen und Schnittstellen angeboten werden. Das betrifft insbesondere den Ablauf von EDT-Prozeduren und die Unterprogramm-Schnittstellen.

Die Vergrößerung der erlaubten Satzlänge und die interne Verwendung einer Unicode-Darstellung in den Arbeitsdateien hat zur Konsequenz, dass alle Schnittstellen, an denen der Anwender bisher direkten Zugriff auf die internen Daten des EDT hatte, nicht kompatibel bleiben können. Dies trifft auf die alte L-Modus Unterprogramm-Schnittstelle, auf die bisherige @RUN-Schnittstelle und auf den Locate-Mode der IEDTGLE-Schnittstelle zu. Diese Schnittstellen können daher nicht mehr verwendet werden, wenn man die neuen Funktionen nutzen will.

Trotzdem sollen Programme, die die Unterprogramm-Schnittstellen des EDT lediglich dazu benutzen, ihren Anwendern ein Editieren von Dateien ohne Verlassen des Programms zu erlauben, möglichst ohne Änderungen auch mit in Unicode codierten Dateien arbeiten können.

Dies wird dadurch erreicht, dass der EDT V17.0A in zwei Modi betrieben werden kann:

- Im Unicode-Modus, der für die Bearbeitung von Unicode-Dateien erweitert wurde, aber einige wenige Inkompatibilitäten, insbesondere an der Unterprogramm-Schnittstelle aufweist.
- Im Kompatibilitäts-Modus, der die volle Funktionalität des EDT V16.6B bietet, aber weder die Bearbeitung von Unicode-Dateien erlaubt, noch die vergrößerte Satzlänge oder das lokale Einstellen von Zeichensätzen unterstütz

Weitere Informationen zu den **Betriebsmodi** und über ihr **Zusammenspiel** finden Sie in **Kapitel 11**.

2.2 Unicode-Modus

2.2.1 Zusätzliche Funktionen - Übersicht

Im Unicode-Modus werden folgende zusätzliche Funktionen geboten:

- Bearbeitung von Dateien unterschiedlicher Zeichensätze in den Arbeitsdateien des EDT, insbesondere von Dateien in Unicode-Codierung (`UTF16`, `UTFE` oder `UTF8`) und auch in ISO-Codierung (die Sonderbehandlung für ISO-codierte POSIX-Dateien geht in dem neuen Konzept auf). Interpretation einer Ersatzdarstellung von Unicode-Zeichen.
- Bearbeitung von Dateien mit Satzlängen bis 32768 Bytes (Obergrenze des DVS).
- Konsistente Behandlung von leeren Sätzen.
- Verfügbarkeit aller 23 Arbeitsdateien auch im F-Modus.
- Konsistente Erweiterung vieler Anweisungen.

2.2.2 Zusätzliche Funktionen - Erläuterungen

Lokale Zeichensätze

Im Unicode-Modus des EDT V17.0A kann für jede Arbeitsdatei ein anderer Zeichensatz eingestellt werden. Dabei sind die schon bisher unterstützten 7-Bit- und 8-Bit-Zeichensätze sowie zusätzlich die Unicode-Zeichensätze `UTF8`, `UTF16` und `UTFE`, die von XHCS unterstützten ISO-Zeichensätze sowie in XHCS deklarierte benutzerdefinierte Zeichensätze erlaubt.

In Anweisungen innerhalb von Literalen aber auch in Daten können Unicode-Zeichen über eine Ersatzdarstellung durch die Angabe des Hex-Wertes ihres `UTF16`-Codes angegeben werden.

Die Zeichensätze für die einzelnen Arbeitsdateien werden entweder implizit durch das Einlesen einer entsprechend codierten Datei eingestellt oder explizit über die @CODENAME-Anweisung.

Das Arbeiten mit lokalen Zeichensätzen, insbesondere beim Datentransfer zwischen Arbeitsdateien oder zwischen EDT-Variablen und Arbeitsdateien wird ausführlich im Abschnitt „Zeichensätze“ auf Seite 48 beschrieben.

Lange Sätze

Im Unicode-Modus des EDT V17.0A wird die Begrenzung der Satzlänge auf 256 Zeichen aufgehoben. Der EDT kann beim Lesen aus und Schreiben in eine DVS-Datei (je nach Dateiformat) maximal 32768 Byte lange Sätze verarbeiten. Diese Grenze ist durch das DVS vorgegeben und bezieht sich auf die *Byteanzahl*. Bei Bibliothekselementen liegt die durch LMS vorgegebene Grenze bei 32763 Bytes. Da bei Verwendung von Unicode-Zeichensätzen Zeichen auch durch mehrere Bytes codiert sein können, ist die Anzahl erlaubter *Zeichen* pro Satz möglicherweise geringer. Intern arbeitet der EDT mit längeren Puffern, so dass diese Begrenzung erst wirksam wird, wenn die Sätze zum Schreiben in den Zeichensatz der Ausgabedatei umcodiert werden. Der EDT bietet eine Anweisung an (@CHECK, Format 2), mit der sich überprüfen lässt, ob Sätze beim Schreiben gekürzt werden müssen.

Für POSIX-Dateien ist die Zeilenlänge nur durch die maximale Zeilenlänge im EDT auf 32768 Zeichen (nicht Bytes) begrenzt.

Die Aufhebung der Satzlängenbegrenzung betrifft Datensätze in den Arbeitsdateien, Zeichenfolgevariablen und die Anweisungslänge gleichermaßen und findet ihren Niederschlag auch in der Syntax der Anweisungen (z.B. bei Spaltenangaben), im Layout der Zustandsanzeige, in der Aufteilung des Bildschirms beim @EDIT LONG sowie in der Unterprogramm-Schnittstelle.

Konsistente Behandlung von Leerzeilen

Die vom EDT zu verarbeitenden Dateien können Sätze der Länge 0 enthalten.
Bei POSIX-Dateien oder SAM-Dateien handelt es sich dabei real um Sätze der Länge 0, bei ISAM-Dateien mit Standardeigenschaften um Sätze der Länge 8 bzw. der Länge 16 (bei in `UTF16` codierten Dateien).

Um Sätze der Länge 0 im Datenfenster darstellen zu können, kennzeichnet der EDT im Unicode-Modus das Satzende mit dem datenstationsspezifischen Zeichen LZE (Logisches Zeilenende).
Der Rest der Bildschirmzeile rechts vom LZE wird von der Datensichtstation (DSS) mit geschützten `NIL`-Zeichen (`X'00'`) gefüllt. Wenn sich das Satzende außerhalb des Datenfensters befindet, wird LZE nicht dargestellt. Die Bildschirmzeile endet dann mit dem letzten noch sichtbaren Zeichen des Satzes bzw. besteht nur aus geschützten `NIL`-Zeichen. Ein Satz der Länge Null wird also durch eine Bildschirmzeile im Datenfenster dargestellt, die nur aus dem Zeichen LZE in Spalte 1 und geschützten `NIL`-Zeichen besteht (Leerzeile). Wird in einer Zeile in Spalte 1 LZE eingegeben, dann wird für diese Zeile ein Satz der Länge 0 in der Arbeitsdatei angelegt.

Leerzeilen sind zu unterscheiden von neuen Zeilen, die der EDT im F-Modus nach dem letzten Satz der Datei bzw. bei der Bearbeitung der Kurzanweisungen `1..9` oder `I` anbietet. Diese Zeilen entsprechen (noch) keinem Satz in der Arbeitsdatei und bestehen nur aus (überschreibbaren) `NIL`-Zeichen (`X'00'`) ohne LZE.

Bei der Eingabe kann das Zeichen LZE normalerweise weggelassen werden.
Es muss lediglich dann eingegeben werden, wenn der Satz mit `NIL`-Zeichen enden soll. Der EDT ignoriert bei der Eingabe im F-Modus alle NIL-Zeichen am Ende der eingegebenen Zeile, d.h. alle `NIL`-Zeichen bis zu dem ersten Zeichen ungleich `NIL` (dies kann ein LZE oder ein anderes Zeichen sein) werden von rechts abgeschnitten. Das LZE selbst wird ebenfalls bei der Eingabe ignoriert. Da neue Zeilen nur aus `NIL`-Zeichen bestehen, werden sie bei der Eingabe als Ganzes ignoriert und nicht in die Arbeitsdatei eingefügt. Im Gegensatz dazu würde die Eingabe eines LZE in Spalte `n` einer neuen Zeile dazu führen, dass nach der Datenübertragung standardmäßig ein Satz mit `n-1` Leerzeichen in die Arbeitsdatei eingefügt wird (oder mit `n-1` `NIL`-Zeichen, je nach Festlegung durch @SYMBOLS FILLER). Insbesondere würde für `n=1` ein Satz der Länge 0 eingefügt.

Die DSS gestattet keine Eingaben in einer Zeile rechts vom Zeichen LZE. Beim Anfügen von Zeichen an eine Zeile ist also entweder der Einfügemodus der DSS einzuschalten oder das Zeichen LZE zu überschreiben. Diese Inkompatibilität zu EDT V16.6B wird zu Gunsten der konsistenten Behandlung von Leerzeilen im Unicode-Modus in Kauf genommen.

Eine genauere Beschreibung des Verhaltens bei der Eingabe von Zeilen, die `NIL`-Zeichen oder Füllzeichen enthalten, finden Sie im Abschnitt „F-Modus“ auf Seite 105.

Verfügbarkeit aller Arbeitsdateien

Im F-Modus kann jetzt auch mittels Eingabe der entsprechenden Nummer in die Anweisungszeile in eine der Arbeitsdateien 10 bis 22 gewechselt werden. Bisher war dies nur im L-Modus mit der @PROC-Anweisung möglich (siehe dazu den Abschnitt „F-Modus“ auf Seite 105).

Vereinheitlichung von Anweisungs-Schnittstellen

Im EDT V17.0A kann jetzt auf Dateien aller unterstützten Dateitypen mit einem einheitlichen Satz von Anweisungen (@OPEN, @CLOSE, @COPY, @WRITE) zugegriffen werden (siehe dazu Kapitel „Dateibearbeitung“ auf Seite 137 sowie die Beschreibung der einzelnen Anweisungen).

Die Anweisungen wurden dafür um entsprechende Operanden erweitert. Die Verwendung der alten Anweisungen ist zwar weiterhin möglich, es wird jedoch empfohlen, nur noch die neuen Anweisungen zu verwenden.

Auch bei anderen Anweisungen wurden zum Zwecke der Vollständigkeit und Vereinheitlichung neue Operanden eingeführt. Einzelheiten können Sie den Anweisungsbeschreibungen entnehmen.

2.2.3 Nicht mehr unterstützte Funktionen

Im Unicode-Modus stehen folgende Funktionen des EDT V16.6B nicht mehr zur Verfügung:

- Ausgaben in SDF-Listenvariablen bei @LOG
- V15-kompatible Syntaxkontrolle des L-Modus
- Unterstützung der alten L-Modus-Unterprogramm-Schnittstelle.
- Unterstützung des Locate-Mode der `IEDTGLE`-Schnittstelle.
- Unterstützung der bisherigen @RUN-Schnittstelle. Eine neue @RUN Schnittstelle wird angeboten.
- Unterstützung von Datensichtstationen mit arabischem oder Farsi-Zeichensatz.
- Unterstützung der Datensichtstation 3270 (IBM) und von Schreibstationen.
- Die @CODE-Anweisung, deren Verwendung im Zusammenhang mit XHCS schon im Handbuch EDT V16 als nicht sinnvoll bezeichnet wurde.
- Die Anweisungen @UPDATE und @ZERO-RECORDS.

2.3 Kompatibilitäts-Modus

Der Kompatibilitäts-Modus bietet die volle Funktionalität des EDT V16.6B inklusive der alten L-Modus-Unterprogramm-Schnittstelle. Die erweiterten Funktionen des Unicode-Modus sind im Kompatibilitäts-Modus nicht verfügbar.

Der Kompatibilitäts-Modus des EDT V17.0A wurde nur um wenige, notwendige Funktionen erweitert.

Eine ausführliche Beschreibung des **Kompatibilitäts-Modus** finden Sie in **Kapitel 11**.

2.3.1 @CODENAME-Anweisung

Die @CODENAME-Anweisung wird auch im Kompatibilitätsmodus erweitert, so dass für Migrationszwecke ein gezieltes Ändern des Zeichensatzes auch bei nicht leeren Arbeitsdateien möglich wird.

2.3.2 @IF-Anweisung

Der Kompatibilitäts-Modus des EDT V17.0A wird um das Format 5 der @IF-Anweisung erweitert, die es erlaubt, den aktuellen Betriebs-Modus abzufragen und gegebenenfalls zu reagieren.

2.3.3 @MODE-Anweisung

Die neue @MODE-Anweisung wird auch im Kompatibilitätsmodus eingeführt und ermöglicht das Umschalten in den Unicode-Modus.

2.3.4 Meldungen

Die Meldung `EDT4983` kann im Kompatibilitäts-Modus des EDT V17.0A neu auftreten.

3 Konzepte des EDT

In diesem Abschnitt werden die grundlegenden Konzepte und übergreifenden Mechanismen beschrieben, die im EDT verwendet werden.

3.1 Arbeitsdateien

Dem Benutzer stehen 23 Arbeitsdateien im virtuellen Speicher zur Dateibearbeitung zur Verfügung, die Arbeitsdateien 0 bis 22. Die Arbeitsdateien können Sätze bis zu einer Länge von 32768 Zeichen aufnehmen. Damit können DVS-Dateien und Bibliothekselemente mit den maximal erlaubten Satzlängen bearbeitet werden. Die maximale Satzanzahl in einer Arbeitsdatei ist 99999999.
In Arbeitsdateien können Daten neu eingegeben werden, existierende Dateien zur Bearbeitung eingelesen werden, Daten durch EDT-Anweisungen erzeugt und bearbeitet werden oder aus anderen Arbeitsdateien kopiert werden.

Die Arbeitsdatei 9 wird im F-Modus von einigen EDT-Anweisungen zur Ablage von Ergebnissen verwendet (@COMPARE, @FSTAT, @SHIH, @SHOW, @STAJV, @STATUS). Dabei wird evtl. vorhandener Inhalt ohne Warnung gelöscht.

Ist jedoch in der Arbeitsdatei 9 eine Datei geöffnet, wird die Meldung `EDT5189` ausgegeben und die jeweilige Anweisung nicht ausgeführt. Die Arbeitsdatei 9 sollte deshalb nur als temporäre Hilfsdatei verwendet werden.

3.1.1 Eigenschaften von Arbeitsdateien

Jede Arbeitsdatei hat bestimmte Eigenschaften, die durch EDT-Anweisungen veränderbar sind und die die Wirkungsweise von EDT-Anweisungen oder die Darstellung der Arbeitsdatei beeinflussen. In der folgenden Tabelle sind die Eigenschaften einer Arbeitsdatei zusammengestellt.

Eigenschaft	Initialwert	Wert veränderbar durch
Allgemeines		
aktueller Zeichensatz der Arbeitsdatei	*NONE	@CODENAME implizit durch Dateneingabe (Einlesen einer Datei, Eingabe am Bildschirm, Anweisungen)
Arbeitsdatei belegt (nur für Arbeitsdateien 1 bis 22)	nicht belegt	@PROC, @DELETE, @DROP, Verwendung im F-Modus
Arbeitsdatei leer	ja	Einlesen einer Datei, Eingabe am Bildschirm, diverse Anweisungen
Arbeitsdatei verändert	nein	Veränderung der Arbeitsdatei, @DELETE
Sicherungsdatei vorhanden	nein	@AUTOSAVE und Veränderung der Arbeitsdatei, @DELETE
Zeilennummern		
aktuelle Zeilennummer (symbolische Zeilennummer *)	1.	@SET (Format 6), @+, @-, implizit durch Dateneingabe (Einlesen einer Datei, Eingabe am Bildschirm, Anweisungen)
aktuelle Schrittweite	1.	@SET (Format 6), @PAR INCREMENT
aktuelle Umnummerierung	ein	@PAR RENUMBER
niedrigste vergebene Zeilennummer (angezeigt durch @LIMITS)	0.0000	implizit durch Einlesen einer Datei, Eingabe am Bildschirm, diverse Anweisungen
symbolische Zeilennumer %	= *	implizit durch Einlesen einer Datei, Eingabe am Bildschirm, diverse Anweisungen
höchste vergebene Zeilennummer (angezeigt durch @LIMITS)	0.0000	implizit durch Einlesen einer Datei, Eingabe am Bildschirm, diverse Anweisungen
symbolische Zeilennummer $	= *	implizit durch Einlesen einer Datei, Eingabe am Bildschirm, diverse Anweisungen
symbolische Zeilennummer ?	0.0000	@ON
Speicherbereich für @SET (Format 6)	leer	@SET (Format 6)

Eigenschaft	Initialwert	Wert veränderbar durch
Dateibearbeitung		
Verknüpfung mit Dateinamen (lokaler @FILE-Eintrag)	Keine Verknüpfung	@FILE, @READ, @GET, @DELETE
Verknüpfung mit geöffneter Datei	Keine Verknüpfung	@OPEN, @CLOSE
Voreinstellung des Bibliotheksnamens	*NONE	@PAR LIBRARY
Voreinstellung des Element-Typs	S	@PAR ELEMENT-TYPE
Voreinstellung des Zeichensatzes einer POSIX-Datei	EDF041	@PAR CODE
Eingabe		
Unterscheidung zwischen Groß- und Kleinbuchstaben	ein	@PAR LOWER
Maximale Satzlänge bei Eingabe im F-Modus	32768	@PAR LIMIT
Fluchtsymbol für Unicode-Ersatzdarstellung	*NONE	@PAR ESCAPE-CHARACTER
Unicode-Ersatzdarstellung auch für Daten	aus	@PAR DATA-REPLACEMENT
Darstellung der Arbeitsdatei		
Vollständige Darstellung von Sätzen im F-Modus	aus	@PAR EDIT-LONG
Hexadezimalmodus	aus	@PAR HEX
Datenfenster und Kurzanweisungsspalte gleichzeitig überschreibbar	aus	@PAR EDIT-FULL
Zeilenlineal im Datenfenster	aus	@PAR SCALE
Informationszeile im Datenfenster	aus	@PAR INFORMATION

Eigenschaft	Initialwert	Wert veränderbar durch
Datenfensterspezifische Darstellung		
erste angezeigte Zeile in Datenfenster 1	0.0000	@SETF, +, −, ++, −−, Kurzanweisungen +, −, B, I, S, 1..9 in Datenfenster 1
erste angezeigte Spalte in Datenfenster 1	1	@SETF, >, <, << in Datenfenster 1
Zeilennummernanzeige in Datenfenster 1	ein	@PAR INDEX in Datenfenster 1
erste angezeigte Zeile in Datenfenster 2	0.0000	@SETF, +, −, ++, −−, Kurzanweisungen +, −, B, I, S, 1..9 in Datenfenster 2
erste angezeigte Spalte in Datenfenster 2	1	@SETF, >, <, << in Datenfenster 2
Zeilennummernanzeige in Datenfenster 2	ein	@PAR INDEX in Datenfenster 2
Sonstige		
Programmname für die SDF-Syntax-Prüfung	*NONE	@PAR SDF-PROGRAM
Typ des Programmnamens für die SDF-Syntax-Prüfung	INTERNAL	@PAR SDF-NAME-TYPE
Zeichen für das Auftrennen von Datensätzen	*NONE	@PAR SEPARATOR
Zeichen für das Strukturblättern	@	@PAR STRUCTURE
Schreibschutz auf Satzebene	aus	@PAR PROTECTION
Indikator für Treffer bei letztem @ON für @IF (Format 3)	aus	@ON
Spalte des Treffers bei letztem @ON für @IF (Format 3)	0	@ON

Die mit der Anweisung @PAR definierten Eigenschaften einer Arbeitsdatei können mit @PAR $0..$22 ohne weitere Operanden wieder auf ihre Initialwerte zurückgesetzt werden.

3.1.2 Aktuelle Arbeitsdatei

Zu jedem Zeitpunkt gibt es genau eine Arbeitsdatei, die als aktuelle Arbeitsdatei bezeichnet wird. In der aktuellen Arbeitsdatei werden Daten eingegeben und EDT-Anweisungen wirksam (falls nicht in der Anweisung explizit eine andere Arbeitsdatei angegeben ist).

Im F-Modus wird normalerweise ein Ausschnitt der aktuellen Arbeitsdatei am Bildschirm angezeigt. Die Nummer der aktuellen Arbeitsdatei wird in der Statusanzeige angezeigt. Man kann den angezeigten Ausschnitt der Arbeitsdatei verschieben (siehe Anweisungen @SETF und +, -, ++, --, >, <, <<) oder die aktuelle Arbeitsdatei wechseln (siehe Anweisungen @SETF, $0..$22 und 0..22).

Man kann auch das Arbeitsfenster teilen und Ausschnitte zweier Arbeitsdateien gleichzeitig anzeigen (siehe Abschnitt F-Modus). In diesem Fall wechselt die aktuelle Arbeitsdatei zwischen den beiden angezeigten Arbeitsdateien. Bei der Abarbeitung einer Anweisung oder Kurzanweisung (siehe Abschnitt „Abarbeitungsreihenfolge“ auf Seite 120) ist jeweils die Arbeitsdatei die aktuelle, aus deren Anweisungszeile bzw. Kurzanweisungsspalte die Anweisung bzw. Kurzanweisung stammt. Bei der Verarbeitung von Eingaben im Datenfenster ist diejenige Arbeitsdatei die aktuelle, in deren Datenfenster die Eingaben erfolgten.

Im L-Modus wird die Nummer der aktuellen Arbeitsdatei durch die Anweisung @PROC angezeigt. Das Wechseln der aktuellen Arbeitsdatei erfolgt mit den Anweisungen @SETF, @PROC und @END. Jedoch kann eine aktive Arbeitsdatei (die eine laufende @DO-Prozedur enthält) niemals zur aktuellen Arbeitsdatei gemacht werden.

Beim Start des EDT ist die Arbeitsdatei 0 die aktuelle Arbeitsdatei.

3.1.3 Leere Arbeitsdatei

Eine leere Arbeitsdatei ist eine Arbeitsdatei, die keine Sätze enthält. Sie entsteht

- beim Start des EDT oder
- beim vollständigen Löschen der Arbeitsdatei mit @DELETE (Format 2), @DROP oder durch andere Anweisungen, die implizit ein @DELETE (Format 2) ausführen oder
- beim Löschen aller Zeilen im F-Modus mit der Kurzanweisung `D` oder `M` sowie beim Löschen aller Zeilen mit @DELETE (Format 1), @MOVE oder @ON (Format 8 oder 10).

Im Dialogbetrieb erkennt man eine leere Arbeitsdatei unmittelbar am Nicht-Vorhandensein von Sätzen. In EDT-Prozeduren kann man mit der Anweisung @IF (Format 3) prüfen, ob eine Arbeitsdatei leer ist.

Das Schreiben einer leeren Arbeitsdatei ist möglich. Dabei wird abhängig von den Operanden der jeweiligen Anweisung ggf. auch der für die Arbeitsdatei eingestellte Zeichensatz als Zeichensatz der Datei eingetragen.

Die Eigenschaften einer Arbeitsdatei beim Start des EDT können der Tabelle im vorigen Abschnitt entnommen werden. Alle mit der Anweisung @PAR definierten Eigenschaften einer Arbeitsdatei mit Ausnahme der nachfolgend explizit aufgeführten bleiben beim Löschen erhalten.

In der folgenden Tabelle sind die weiteren Eigenschaften einer Arbeitsdatei nach dem Löschen zusammengestellt.

Eigenschaft	**Wert nach dem Löschen mit @DELETE (Format 2) oder @DROP**	**Wert nach dem Löschen aller vorhandenen Zeilen**
Allgemeines		
aktueller Zeichensatz der Abeitsdatei	*NONE	nicht verändert
Arbeitsdatei belegt (nur für Abeitsdateien 1 bis 22)	nein (außer aktuelle)	ja
Arbeitsdatei leer	ja	ja
Arbeitsdatei verändert	nein	ja
Sicherungsdatei vorhanden	Sicherungsdatei gelöscht	Sicherungsdatei gelöscht
Zeilennummern		
aktuelle Zeilennummer (symbolische Zeilennummer *)	1.	0 + aktuelle Schrittweite
aktuelle Schrittweite	1.	nicht verändert
niedrigste vergebene Zeilennummer (angezeigt durch @LIMITS)	0.0000	0.0000
symbolische Zeilennumer %	1.	==*
höchste vergebene Zeilennummer (angezeigt durch @LIMITS)	0.0000	0.0000
symbolische zeilennummer $	1.	==*
symbolische zeilennummer ?	0.0000	nicht verändert
Speicherbereich für @SET (Format 6)	leer	nicht verändert
Dateibearbeitung		
Verknüpfung mit Dateinamen	keine Verknüpfung	nicht verändert
Verknüpfung mit geöffneter Datei	keine Verknüpfung (implizites @CLOSE wird ausgeführt)	nicht verändert (aufhebbar durch @CLOSE)
Eingabe		
Unicode-Ersatzdarstellung auch für Daten	aus	nicht verändert

Eigenschaft	Wert nach dem Löschen mit @DELETE (Format 2) oder @DROP	Wert nach dem Löschen aller vorhandenen Zeilen
Datenfensterspezifische Darstellung		
erste angezeigte Zeile in Datenfenster 1	0.0000	0.0000
erste angezeigte Spalte in Datenfenster 1	1	nicht verändert
erste angezeigte Zeile in Datenfenster 2	0.0000	0.0000
erste angezeigte Spalte in Datenfenster 2	1	nicht verändert
Sonstige		
Indikator für Treffer bei letztem @ON für @IF (Format 3)	aus	nicht verändert
Spalte des Treffers bei letztem @ON für @IF (Format 3)	0	nicht verändert

3.2 Zeilennummern

Jede Zeile einer Arbeitsdatei hat eine 8-stellige Zeilennummer (Wertebereich 0000.0001 bis 9999.9999), aber es muss nicht zu jeder Zeilennummer eine Zeile existieren.
Die Zeilennummern dienen zur eindeutigen Identifikation von Zeilen innerhalb einer Arbeitsdatei und durch ihren numerischen Wert wird außerdem für alle Zeilen einer Arbeitsdatei eine Reihenfolge festgelegt (hat eine Zeile eine kleinere Zeilennummer als eine andere Zeile, so liegt sie vor der anderen Zeile).
Dadurch ist es auch möglich, Bereiche von Zeilen anzugeben und zwar durch die kleinste und die größte Zeilennummer eines Zeilenbereichs.
Bei vielen EDT-Anweisungen, die in irgendeiner Form Zeilen bearbeiten, werden Zeilennummern verwendet, um die von der Anweisung zu bearbeitenden Zeilen oder Zeilenbereiche festzulegen.

3.2.1 Aktuelle Zeilennummer und aktuelle Schrittweite

Jeder Arbeitsdatei sind eine *aktuelle Zeilennummer* und eine *aktuelle Schrittweite* zugeordnet. Im L-Modus erfolgt die Dateneingabe in der Zeile mit der aktuellen Zeilennummer. Die neue aktuelle Zeilennummer ergibt sich danach aus der bisherigen aktuellen Zeilennummer plus der aktuellen Schrittweite. Wenn im L-Modus mit `WRTRD` gelesen wird, dient die aktuelle Zeilennummer als Prompt für die Dateneingabe. Eine Zeile mit der aktuellen Zeilennummer kann, muss aber nicht existieren.

Bei einigen EDT-Anweisungen lässt sich die Stelle der Arbeitsdatei, an der Zeilen eingefügt werden sollen, durch eine temporäre aktuelle Zeilennummer und eine temporäre aktuelle Schrittweite festlegen, die über die Operanden der betreffenden EDT-Anweisung eingegeben werden und nur für diese eine Anweisung gültig sind, aber ansonsten in dieser EDT-Anweisung die Rolle der aktuellen Zeilennummer und der aktuellen Schrittweite übernehmen. Die aktuelle Zeilennummer kann sich allerdings durch die Ausführung einer solchen EDT-Anweisung ändern.

Nach dem Start des EDT oder nachdem eine Arbeitsdatei vollständig gelöscht wurde (explizit oder implizit), ist die aktuelle Zeilennummer `1.0000` und die aktuelle Schrittweite `1.0000`. Die aktuelle Zeilennummer und die aktuelle Schrittweite können mit der Anweisung @SET (Format 6) neu festgelegt werden, die aktuelle Schrittweite allein auch mit der Anweisung @PAR INCREMENT.

Mit den Anweisungen @+ und @- wird die aktuelle Zeilennummer neu festgelegt, indem die aktuelle Schrittweite zu der bisherigen aktuellen Zeilennummer addiert bzw. von ihr subtrahiert wird. Wenn zuvor die Anweisung @EDIT mit dem Operand `SEQUENTIAL` abgesetzt worden ist, so wird die aktuelle Zeilennummer nur dann auf diese Weise gebildet, wenn zwischen der vorherigen aktuellen Zeilennummer und der neuen aktuellen Zeilennummer keine Zeilen existieren. Im anderen Fall wird die erste dazwischen liegende Zeilennummer zur aktuellen Zeilennummer. Die aktuelle Schrittweite bleibt in jedem Fall unverändert.

Durch die Anweisung @RENUMBER wird sowohl die aktuelle Zeilennummer als auch die aktuelle Schrittweite verändert. Die neue aktuelle Zeilennummer ergibt sich aus der Zeilennummer der nach der Neunummerierung letzten Zeile der Arbeitsdatei plus der durch die Anweisung @RENUMBER neu festgelegten aktuellen Schrittweite. Wird bei der Anweisung @RENUMBER nur eine Zeilennummer angegeben, so wird die neue aktuelle Schrittweite implizit in gleicher Weise durch diese Zeilennummer festgelegt wie z.B. bei der Anweisung @SET, Format 6 (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

Die aktuelle Zeilennummer und die aktuelle Schrittweite werden auch durch alle Anweisungen verändert, die die Arbeitsdatei explizit oder implizit vollständig löschen. Nach dem expliziten oder impliziten Löschen der gesamten Arbeitsdatei ist die aktuelle Schrittweite `1.0000` und die aktuelle Zeilennummer ist `1.0000`. Allerdings wird im Fall des impliziten Löschens der gesamten Arbeitsdatei die aktuelle Zeilennummer danach in der Regel durch die löschende Anweisung wieder verändert.

Wird durch die Anweisung @SET (Format 6) die aktuelle Zeilennummer und die aktuelle Schrittweite verändert, so wird das Wertepaar bestehend aus der bisherigen aktuellen Zeilennummer und der bisherigen aktuellen Schrittweite in einem Speicherbereich gesichert, der maximal drei solcher Wertepaare aufnehmen kann (siehe Anweisung @SET, Format 6).

Durch die Angabe der Anweisung @SET (Format 6) ohne Parameter wird das zuletzt im Speicherbereich gesicherte Wertepaar wieder zur aktuellen Zeilennummer und zur aktuellen Schrittweite. Durch alle Anweisungen, die die Arbeitsdatei explizit oder implizit löschen, wird auch dieser Speicherbereich vollständig gelöscht.

3.2.2 Symbolische Zeilennummern

Neben den Zeilennummervariablen (#L00..#L20), die beliebige Zeilennummern aufnehmen können, gibt es vier Sonderzeichen (*, %, $, ?), die symbolische Zeilennummern darstellen.
Die aktuelle Zeilennummer kann über die symbolische Zeilennummer * angesprochen werden, die niedrigste Zeilennummer einer Arbeitsdatei über die symbolische Zeilennummer % und die höchste Zeilennummer einer Arbeitsdatei über die symbolische Zeilennummer $.

Nach dem Start des EDT haben diese drei symbolischen Zeilennummern den Wert 1.0000. Die symbolische Zeilennummer ? enthält die Zeilennummer der ersten Trefferzeile einer vorangegangenen @ON-Anweisung. Sie hat beim Start des EDT den Wert 0.0000 und wird nur durch @ON-Anweisungen verändert, die einen Treffer liefern. Die vier symbolischen Zeilennummern *, %, $ und ? sind arbeitsdateispezifisch.
Darüber hinaus gibt es bei der Anweisung @DO (Format 1) die Möglichkeit, ein Sonderzeichen als Schleifensymbol für eine @DO-Prozedur festzulegen. Dieses Schleifensymbol nimmt der Reihe nach Werte an, die durch einen Startwert, einen Endwert und eine Schrittweite (ebenfalls in der @DO-Anweisung anzugeben) festgelegt werden.
In der Prozedur kann dann dieses Sonderzeichen wie eine symbolische Zeilennummer in EDT-Anweisungen verwendet werden, die den jeweils aktuellen Wert des Schleifensymbols repräsentiert. Welche Sonderzeichen für den Einsatz als Schleifensymbol in Frage kommen, ist der Beschreibung der Anweisung @DO (Format 1) zu entnehmen.

3.2.3 Implizite Schrittweitenvergabe

Wird bei Anweisungen, bei denen sowohl eine Zeilennummer als auch eine Schrittweite angegeben werden kann (z.B. @SET, Format 6), nur eine Zeilennummer angegeben, so wird die Schrittweite implizit durch die angegebene Zeilennummer festgelegt (bei der Zeilennummer und der Schrittweite kann es sich sowohl um die aktuelle Zeilennummer und die aktuelle Schrittweite handeln als auch um Angaben, die nur für die betreffende Anweisung wirksam sind).

Treten dabei in dem Ausdruck, durch den die Zeilennummer festgelegt wird, nur Zeilennummervariablen (#L0..#L20) oder symbolische Zeilennummern (%,*,$,?) oder Ganzzahlvariablen (#I0..#I20) oder relative Zeilennummernangaben (nL) oder eine Kombination davon auf, so ist die neue aktuelle Schrittweite 0.0001.
Ist eine numerische Zeilennummernangabe Bestandteil des Ausdrucks, so bestimmt die Anzahl der maximal vier Nachkommastellen dieser numerischen Zeilennummernangabe die aktuelle Schrittweite. Ist keine Nachkommastelle vorhanden, ist die aktuelle Schrittweite 1, bei einer Nachkommastelle ist die aktuelle Schrittweite 0.1 usw.
Bei vier Nachkommastellen schließlich ist die aktuelle Schrittweite 0.0001.

3.2.4 Zeilennummernvergabe

Das Einbringen neuer Zeilen in eine Arbeitsdatei durch EDT-Anweisungen (z.B. durch Einlesen einer Datei oder Kopieren von Zeilen etc.) aber auch das Löschen von Zeilen, wenn dabei die bisher letzte Zeile der Arbeitsdatei zu den gelöschten Zeilen gehört, hat in den meisten Fällen eine Veränderung der aktuellen Zeilennummer zu Folge (es gibt einige Ausnahmen, wo die aktuelle Zeilennummer unverändert bleibt). In der Regel ergibt sich die neue aktuelle Zeilennummer aus der Zeilennummer der letzten Zeile der Arbeitsdatei plus der aktuellen Schrittweite. Aber in einigen Fällen ergibt sich die aktuelle Zeilennummer aus der Zeilennummer der letzten durch die EDT-Anweisung eingefügten Zeile plus der aktuellen Schrittweite.

Die Nummerierung von neu in eine Arbeitsdatei eingebrachten Zeilen wird nach einem der fünf nachfolgend beschriebenen Verfahren durchgeführt.

3.2.4.1 Verwendung der Zeilennummern der Quelle

Es werden die Zeilennummern der Quelle verwendet. Beim Kopieren sind dies die Zeilennummern der zu kopierenden Zeilen aus einer anderen Arbeitsdatei.
Beim Einlesen aus einer Datei werden die Zeilennummern im Fall von ISAM-Dateien aus dem ISAM-Schlüssel gebildet (falls nichts anderes angegeben), im Fall von SAM-Dateien mit Angabe des Operanden KEY werden die ersten acht Zeichen eines jeden Satzes als Zeilennummer interpretiert.
Bereits in der Zielarbeitsdatei existierende Zeilen mit gleichen Zeilennummern werden überschrieben. Die neue aktuelle Zeilennummer ergibt sich aus der Zeilennummer der letzten Zeile plus der aktuellen Schrittweite, falls sich die Zeilennummer der letzten Zeile der Arbeitsdatei geändert hat, ansonsten bleibt sie unverändert.

Der folgenden Tabelle kann entnommen werden, bei welcher EDT-Anweisung dieses Verfahren angewendet wird (bei einigen dieser EDT-Anweisungen kommen bei einem anderen Format oder anderen Operanden auch noch andere Verfahren zur Anwendung).

Anweisung	Bemerkungen zu den Operanden	Allgemeine Bemerkungen
@COPY Format 1	Einlesen von ISAM-Dateien mit Angabe des Operanden KEY=LINENUMBER	
@COPY Format 2	ohne Zielangabe	
@GET	mit Angabe des Operanden NORESEQ	
@MOVE	ohne Zielangabe	
@ON Format 4 + 5	mit Angabe des Operanden KEEP	Wird der Operand OLD nicht angegeben, so wird die Zielarbeitsdatei vor dem Einfügen gelöscht.
@OPEN Format 1	Öffnen von ISAM-Dateien mit Angabe des Operanden KEY=LINENUMBER	Vor Ausführung der Anweisung muss die Arbeitsdatei leer sein.
@OPEN Format 2	Öffnen einer Kopie von SAM-Dateien mit Angabe des Operanden KEY und von ISAM-Dateien	Vor Ausführung der Anweisung muss die Arbeitsdatei leer sein. Ist bereits eine Datei mit der Anweisung @OPEN (Format 2) geöffnet und wird erneut eine @OPEN-Anweisung (Format 2) abgesetzt, so wird vor der zweiten @OPEN-Anweisung implizit eine @CLOSE-Anweisung abgesetzt, d.h., die Arbeitsdatei wird implizit gelöscht.
@READ	mit Angabe des Operanden KEY	

3.2.4.2 Einfügen bei der aktuellen Zeilennummer

Das Einfügen von Zeilen beginnt bei der aktuellen Zeilennummer. Die Zeilennummern der weiteren einzufügenden Zeilen ergeben sich durch Addieren der aktuellen Schrittweite zu der Zeilennummer der jeweils zuletzt eingefügten Zeile. Bereits in der Zielarbeitsdatei existierende Zeilen mit gleichen Zeilennummern werden überschrieben. Die neue aktuelle Zeilennummer ergibt sich aus der Zeilennummer der letzten neu eingefügten Zeile plus der aktuellen Schrittweite.

Der folgenden Tabelle kann entnommen werden, bei welcher EDT-Anweisung dieses Verfahren angewendet wird (bei einigen dieser EDT-Anweisungen kommen bei einem anderen Format oder anderen Operanden auch noch andere Verfahren zur Anwendung).

Anweisung	**Bemerkungen zu den Operanden**	**Allgemeine Bemerkungen**
L-Modus-Dateneingabe		Das betrifft nicht nur Eingaben am L-Modus-Prompt, sondern auch Dateneingaben in EDT-Prozeduren und über den `text`-Operanden einiger L-Modus-Anweisungen.
@GET	ohne Angabe des Operanden `NORESEQ`	
@GETLIST		
@ON Format 4 + 5	ohne Angabe des Operanden `KEEP` aber mit Angabe des Operanden `OLD`	
@OPEN Format 2	Öffnen einer Kopie von SAM-Dateien ohne Angabe des Operanden `KEY`	Vor Ausführung der Anweisung muss die Arbeitsdatei leer sein. Dies ist der Fall nach dem Start des EDT oder nach dem expliziten Löschen. In beiden Fällen ist die aktuelle Zeilennummer und die aktuelle Schrittweite jeweils `1.0000`. Man kann aber vor Ausführung der Anweisung eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite einstellen.
@READ	ohne Angabe des Operanden `KEY`	

3.2.4.3 Einfügen nach implizitem Löschen

Die Arbeitsdatei, in die Zeilen eingefügt werden sollen, wird vor dem Einfügen implizit vollständig gelöscht, d.h., die aktuelle Schrittweite und die aktuelle Zeilennummer sind unmittelbar vor dem Einfügen jeweils `1.0000`. Das Einfügen von Zeilen beginnt danach bei der aktuellen Zeilennummer (`= 1.0000`).

Die Zeilennummern der weiteren einzufügenden Zeilen ergeben sich durch Addieren der aktuellen Schrittweite (`= 1.0000`) zu der Zeilennummer der jeweils zuletzt eingefügten Zeile. Nach Beendigung des Einfügevorgangs ergibt sich die neue aktuelle Zeilennummer aus der Zeilennummer der letzten neu eingefügten Zeile plus der aktuellen Schrittweite.

Der folgenden Tabelle kann entnommen werden, bei welcher EDT-Anweisung dieses Verfahren angewendet wird (bei einigen dieser EDT-Anweisungen kommen bei einem anderen Format oder anderen Operanden auch noch andere Verfahren zur Anwendung).

Anweisung	**Bemerkungen zu den Operanden**	**Allgemeine Bemerkungen**
@ON Format 4 + 5	ohne Angabe der Operanden `KEEP` und `OLD`	
@OPEN Format 2	Öffnen einer Kopie von SAM-Dateien ohne Angabe des Operanden `KEY` in einer Arbeitsdatei, in der bereits eine Datei mit @OPEN (Format 2) geöffnet ist.	Vor der zweiten @OPEN-Anweisung wird implizit eine @CLOSE-Anweisung abgesetzt, d.h., die Arbeitsdatei wird implizit gelöscht.
@SHIH		Die Ausgabe erfolgt in die Arbeitsdatei 9.
@STATUS	ohne Zielangabe bei Ausgabe im F-Modus	Die Ausgabe erfolgt in die Arbeitsdatei 9.

3.2.4.4 Einfügen bei vorgegebener Zeilennummer

Das Einfügen von Zeilen beginnt bei der in der Anweisung als Operand angegebenen Zeilennummer. Die Zeilennummern der weiteren einzufügenden Zeilen ergeben sich durch Addieren der in der Anweisung evtl. als Operand angegebenen Schrittweite zu der Zeilennummer der jeweils zuletzt eingefügten Zeile.

Ist in der Anweisung keine Schrittweite angegeben, so wird diese implizit durch die angegebene Zeilennummer (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36) festgelegt, wie bei der impliziten Festlegung der aktuellen Schrittweite durch die aktuelle Zeilennummer. Bereits in der Zielarbeitsdatei existierende Zeilen mit gleichen Zeilennummern werden überschrieben.
Die neue aktuelle Zeilennummer ergibt sich aus der Zeilennummer der letzten Zeile plus der aktuellen Schrittweite, falls sich die Zeilennummer der letzten Zeile der Arbeitsdatei geändert hat, ansonsten bleibt sie unverändert.

Der folgenden Tabelle kann entnommen werden, bei welcher EDT-Anweisung dieses Verfahren angewendet wird (bei einigen dieser EDT-Anweisungen kommen bei einem anderen Format oder anderen Operanden auch noch andere Verfahren zur Anwendung).

Anweisung	**Bemerkungen zu den Operanden**	**Allgemeine Bemerkungen**
@COMPARE Format 1	ohne Angabe des Operanden `LIST` erfolgt die Ausgabe auf den Bildschirm, bei Angabe des Operanden `LIST` ohne Zeilennummer auf `SYSLST`	
@COPY Format 2	mit Zielangabe	
@FSTAT	mit Zielangabe	
@GETJV	mit Ausgabe in eine Zeile	Es wird nur eine Zeile erzeugt.
@GETVAR	mit Ausgabe in eine Zeile	Es wird nur eine Zeile erzeugt.
@MOVE	mit Zielangabe	
@SHOW Format 1 + 2	mit Zielangabe	
@STAJV	mit Zielangabe	
@STATUS	mit Zielangabe	
@SYSTEM	mit Zielangabe	

3.2.4.5 Einfügen zwischen zwei Zeilen

Bei diesem Verfahren werden die neu einzufügenden Zeilen zwischen zwei bereits existierende Zeilen eingefügt, ohne bereits existierende Zeilen zu überschreiben. Zuerst wird versucht, die neu einzufügenden Zeilen unter Verwendung der aktuellen Schrittweite einzufügen. Können die neuen Zeilen nicht auf diese Weise eingefügt werden, wird die Schrittweite zum Einfügen solange durch 10 dividiert, bis das Einfügen der neuen Zeilen möglich ist oder die kleinstmögliche Schrittweite von `0.0001` erreicht wurde.

Konnten auch mit der kleinstmöglichen Schrittweite von `0.0001` die neuen Zeilen nicht komplett eingefügt werden, wird der Einfügevorgang mit einer Fehlermeldung abgebrochen, wenn zuvor durch die Anweisung @PAR RENUMBER=OFF die automatische Umnummerierung von Zeilen ausgeschaltet wurde.

Ist dagegen die automatische Umnummerierung von Zeilen eingeschaltet (@PAR RENUMBER=ON), wird versucht, die größere der beiden Zeilennummern, zwischen die eingefügt werden soll und eventuell weitere nachfolgende Zeilennummern so umzunummerieren, dass der Einfügevorgang erfolgreich abgeschlossen werden kann.

Nur wenn nicht mehr genügend Zeilen umnummeriert werden können, wird der Einfügevorgang auch in diesem Fall mit einer Fehlermeldung abgebrochen. Beim Start des EDT gilt immer die Einstellung @PAR RENUMBER=ON für alle Arbeitsdateien, die aber für jede Arbeitsdatei separat geändert werden kann.
Nachfolgend wird der Einfügevorgang noch etwas genauer beschrieben.

Bei diesem Einfügeverfahren werden immer zwei Zeilennummern benötigt, zwischen die die neuen Zeilen eingefügt werden sollen. Eine der beiden Zeilennummern ist durch die betreffende Anweisung (ggf. implizit) vorgegeben. Bei den Kurzanweisungen des F-Modus ist dies z.B. die Zeilennummer der Zeile, in der die Kurzanweisung angegeben wurde, bei der Anweisung @XCOPY z.B. die Zeilennummer der bisher letzten Zeile der Arbeitsdatei.

In den meisten Fällen wird nach dieser explizit oder implizit gegebenen Zeilennummer eingefügt. Als zweite Zeilennummer wird dann die Zeilennummer der nächsten Zeile genommen. Wird nach der letzten Zeile der Arbeitsdatei eingefügt, so wird als fiktive Zeilennummer für die zweite Zeile `10000.0000` angenommen.

Nur bei den Kurzanweisungen `B,I,1..9` und der Anweisung @COPY (Format 1) mit Angabe des Parameters `BEFORE` wird vor der vorgegebenen Zeile eingefügt. In diesem Fall wird als zweite Zeilennummer die Zeilennummer der vorangehenden Zeile genommen. Wird vor der ersten Zeile eingefügt, so wird als Zeilennummer für die zweite Zeile `0.0000` angenommen. Ist die Arbeitsdatei leer, so wird als erste Zeilennummer `0.0000` und als zweite Zeilennummer `10000.0000` angenommen.
Die fiktive Zeilennummer `0.0000` dient dabei nur zur Bestimmung der Zeilennummer der ersten einzufügenden Zeile (= `0.0000` + Schrittweite) und kann selbst nicht belegt werden.

Einfügen ohne automatische Umnummerierung (@PAR RENUMBER=OFF)

Als Schrittweite für den ersten Einfügeversuch wird die aktuelle Schrittweite genommen. Die Zeilennummer der ersten einzufügenden Zeile erhält man, indem man zu der kleineren der beiden Zeilennummern, zwischen die eingefügt werden soll, die Schrittweite addiert. Die Zeilennummern der weiteren einzufügenden Zeilen ergeben sich durch sukzessives Addieren der Schrittweite. Passen die auf diese Weise nummerierten Zeilen zwischen die beiden Ausgangszeilen hinein, wird diese Nummerierung verwendet.
Passen sie dagegen nicht hinein, hängt das weitere Vorgehen davon ab, ob die Schrittweite gleich der kleinstmöglichen Schrittweite von `0.0001` ist oder nicht.
Ist sie gleich `0.0001`, so wird das Einfügen der Zeilen mit der Meldung `EDT5365` abgewiesen.
Ist die Schrittweite größer als `0.0001`, so wird daraus eine neue Schrittweite ermittelt, indem die vorhergehende Schrittweite durch 10 geteilt wird (die aktuelle Schrittweite wird dabei nicht verändert!). Ist diese neue Schrittweite kleiner als `0.0001`, so wird `0.0001` als neue Schrittweite genommen.
Mit dieser neuen Schrittweite wird nun ein neuer Einfügeversuch unternommen. Können die einzufügenden Zeilen mit dieser Nummerierung eingefügt werden, ist man fertig.
Ist andernfalls die Schrittweite bereits gleich `0.0001`, so wird der Einfügevorgang mit der Meldung `EDT5365` abgewiesen. Ist die Schrittweite noch größer als `0.0001`, so wird erneut durch 10 geteilt und mit dieser neuen Schrittweite ein weiterer Versuch unternommen.
Die aktuelle Zeilennummer ändert sich nur, wenn mindestens eine neue Zeile entsteht, deren Zeilennummer größer ist, als die bisher höchste Zeilennummer und sie ist dann gleich der Zeilennummer der letzten Zeile der Arbeitsdatei plus der aktuellen Schrittweite (dies kann nur beim Einfügen nach der bisher letzten Zeile entstehen).

Einfügen mit automatischer Umnummerierung (@PAR RENUMBER=ON)

In diesem Fall wird ein zweistufiges Verfahren angewendet, um möglichst kompatibel zum EDT V16.6B zu bleiben.
Es wird zunächst genauso vorgegangen, wie im Fall @PAR RENUMBER=OFF.

Ist man bei der Schrittweite `0.01` angekommen und können die Zeilen noch nicht eingefügt werden, wird versucht, alle bereits existierenden Zeilen, die durch die neu eingefügten Zeilen überschrieben worden wären, so umzunummerieren, dass sie an die bisher zuletzt eingefügte Zeile unter Verwendung der Schrittweite `0.01` angefügt werden können. Dies ist das Verfahren des EDT V16.6B bzw. des Kompatibilitäts-Modus. Können dabei nicht genügend Zeilen umnummeriert werden, bricht der EDT V16.6B bzw. der Kompatibilitäts-Modus den Einfügevorgang mit Fehlermeldung ab.
Der Unicode-Modus dagegen versucht nun durch weiteres Verkleinern der Schrittweite (d.h. Teilen durch 10) bis zur kleinstmöglichen Schrittweite von `0.0001`, die Zeilen einzufügen.

Ist man schließlich bei der kleinstmöglichen Schrittweite von `0.0001` angekommen, so werden die Zeilen auf jeden Fall mit der Schrittweite `0.0001` eingefügt. Alle bereits existierenden Zeilen, die durch die neu eingefügten Zeilen überschrieben worden wären, werden nun so umnummeriert, dass sie an die bisher zuletzt eingefügte Zeile unter Verwendung der Schrittweite `0.0001` angefügt werden.
Falls durch diese Umnummerierung noch mal bereits existierende Zeilen überschrieben werden, wiederholt sich der Vorgang, d.h., auch diese Zeilen werden umnummeriert. Wenn durch die Umnummerierung keine bereits existierenden Zeilen mehr überschrieben werden, ist der Vorgang des Einfügens und Umnummerierens beendet.
Auch in diesem Fall ändert sich die aktuelle Zeilennummer nur, wenn mindestens eine neue Zeile entsteht, deren Zeilennummer größer ist, als die bisher höchste Zeilennummer und sie ist dann gleich der Zeilennummer der letzten Zeile der Arbeitsdatei plus der aktuellen Schrittweite.

Eine Ausnahme bildet die Kurzanweisung `0`.
Bei dieser Kurzanweisung werden zunächst, beginnend bei der mit `0` markierten Zeile, nur bereits existierende Zeilen überschrieben. Sind alle nach der mit `0` markierten Zeile bereits überschrieben und sind noch einzufügende Zeilen vorhanden, so werden diese Zeilen nach der letzten überschriebenen Zeile gemäß dem oben beschriebenen Verfahren eingefügt.

Der folgenden Tabelle kann entnommen werden, bei welcher EDT-Anweisung dieses Verfahren angewendet wird (bei einigen dieser EDT-Anweisungen kommen bei einem anderen Format oder anderen Operanden auch noch andere Einfügeverfahren zur Anwendung). Anweisungen, die nicht mit einem EDT-Anweisungssymbol beginnen, sind Kurzanweisungen.

Anweisung	**Bemerkungen zu den Operanden**	**Allgemeine Bemerkungen**
A		
B		
I		
1..9		
O		Wegen der Abweichung vom Verfahren, siehe oben.
T		
@COMPARE Format 2		Die Arbeitsdatei wird vor der Verwendung gelöscht, deshalb Einfügen nach der Zeilennummer `0.0000`.

Anweisung	Bemerkungen zu den Operanden	Allgemeine Bemerkungen
@COPY Format 1	Einlesen von SAM-Dateien, POSIX-Dateien und Bibliothekselementen sowie von ISAM-Dateien mit einer Angabe des Operanden KEY ungleich LINENUMBER	Wird keiner der Operanden AFTER oder BEFORE angegeben, so wird nach der bisher letzten Zeile eingefügt, sonst vor oder nach der angegebenen Zeile.
@FSTAT	ohne Zielangabe bei Ausgabe im F-Modus	Die Arbeitsdatei 9 wird vor der Verwendung gelöscht, deshalb Einfügen nach der Zeilennummer 0.0000.
@OPEN Format 1	Öffnen von SAM-Dateien, POSIX-Dateien und Bibliothekselementen sowie von ISAM-Dateien mit einer Angabe des Operanden KEY ungleich LINENUMBER	Die Arbeitsdatei muss vor Ausführung der Anweisung leer sein, deshalb Einfügen nach der Zeilennummer 0.0000.
@SDFTEST		
@SEPARATE		
@SHOW Format 1 + 2	ohne Zielangabe bei Ausgabe im F-Modus	Die Arbeitsdatei 9 wird vor der Verwendung gelöscht, deshalb Einfügen nach der Zeilennummer 0.0000.
@STAJV	ohne Zielangabe bei Ausgabe im F-Modus	Die Arbeitsdatei 9 wird vor der Verwendung gelöscht, deshalb Einfügen nach der Zeilennummer 0.0000.
@XCOPY		Wenn die Arbeitsdatei nicht leer ist, wird nach der bisher letzten Zeile eingefügt, bei leerer Arbeitsdatei nach der Zeilennummer 0.0000.
@XOPEN		Die Arbeitsdatei muss vor Ausführung der Anweisung leer sein, deshalb Einfügen nach der Zeilennummer 0.0000.

3.3 Satzmarkierungen

Jeder Zeile einer Arbeitsdatei des EDT kann mit einer oder mehreren Satzmarkierungen gekennzeichnet werden. Die Satzmarkierungen werden im Datenbereich des EDT vermerkt und sind für den Benutzer nicht sichtbar. Sie werden beim Schreiben in Dateien nicht übernommen.
Wird der EDT nicht als Unterprogramm aufgerufen, stehen die Satzmarkierungen 1 bis 9 zur Verfügung. Beim Aufruf des EDT als Unterprogramm können zusätzlich für Sonderfunktionen die Satzmarkierungen 13, 14 und 15 verwendet werden.
Die Satzmarkierungen 0, 10, 11 und 12 sind für interne Sonderfunktionen reserviert und stehen dem Anwender nicht zur Verfügung, unabhängig davon, ob der EDT als Unterprogramm aufgerufen wird oder nicht.

Die Satzmarkierungen 1 bis 9 können sowohl mit EDT-Anweisungen bzw. Kurzanweisungen als auch mit den Funktionen `IEDTPUT` und `IEDTPTM` der EDT-Unterprogramm-Schnittstelle gesetzt und gelöscht werden, die Satzmarkierungen 13, 14 und 15 jedoch ausschließlich mit den Funktionen `IEDTPUT` und `IEDTPTM`.

Zeilen in durch @OPEN (Format 2) real geöffneten ISAM-Dateien können nicht markiert werden.

Satzmarkierungen können gesetzt werden durch

- die Kurzanweisung `1..9` F3 in der Kurzanweisungsspalte
- die Anweisung @ON (Format 3) oder
- die Funktionen `IEDTPUT` und `IEDTPTM` bei Aufruf des EDT als Unterprogramm

Satzmarkierungen können gelöscht werden durch

- die Kurzanweisung `D` F3 in der Kurzanweisungsspalte
- die Anweisung @DELETE (Format 4) oder
- die Funktionen `IEDTPUT` und `IEDTPTM` bei Aufruf des EDT als Unterprogramm

Die Satzmarkierungen 1 bis 9 können im F-Modus als Ziel für das Positionieren in der Arbeitsdatei benutzt werden, entweder mit einer der Anweisungen `+`, `++`, `-` bzw. `-` gefolgt von F3 oder mit der Anweisung @SETF.

Bei den Anweisungen @ON (Format 4) und @SETLIST mit dem Parameter `MARK` dienen die Satzmarkierungen 1 bis 9 als ein Mittel zur Auswahl der zu verarbeitenden Zeilen. Mit der Anweisung @ON (Format 4) können markierte Zeilen in eine andere Arbeitsdatei kopiert werden während bei @SETLIST die markierten Zeilen in eine SDF-P-Listenvariable übernommen werden.

Die Satzmarkierungen werden gelöscht, wenn ein Satz neu erstellt wird und dabei einen existierenden Satz ersetzt, sie werden dagegen nicht gelöscht, wenn ein Satz nur modifiziert wird. Neu erstellt werden Sätze bei den Anweisungen @CREATE, @COPY (Format 2), @MOVE, @READ, @GET, @GETJV, @GETVAR, @GETLIST @ON (Format 4, 5) und den Kurzanweisungen `A`, `B` und `O`. Als Modifikation gelten die Eingabe

neuer Inhalte am Terminal (Eintippen), Änderungen durch die Kurzanweisungen L und U und durch die Anweisungen @PREFIX, @SUFFIX, @CONVERT, @COLUMN, @SEQUENCE (Format 1, 2) und @ON (Format 6, 7, 8, 9). Bei der Anweisung @SEPARATE behält die Ursprungszeile die Markierung, aber die neu erzeugten Zeilen erhalten keine Markierung. Bei Verknüpfung mit J behält die obere Zeile ggf. die Markierung, von der unteren Zeile wird die Markierung nicht übernommen.

Bei Umnummerierungen durch @RENUMBER und @SORT, oder beim impliziten Umnummerieren durch Einfügen anderer Sätze bleibt die Markierung eines Satzes erhalten.

Wird mit dem EDT als Unterprogramm gearbeitet, können Satzmarkierungen mit den Funktionen IEDTPUT und IEDTPTM gesetzt oder gelöscht werden.

Mit der Funktion IEDTGTM können Zeilen mit bestimmten Satzmarkierungen gelesen werden. Beim Lesen mit IEDTGET bekommt man ebenfalls die Satzmarkierungen einer Zeile zurück geliefert.

Die Satzmarkierung 13 hat die Sonderfunktion eines Ignorier-Indikators. Derart markierte Zeilen werden

- bei der Rückkehr zum Hauptprogramm nach Aufruf von @DIALOG über die Unterprogrammschnittstelle automatisch gelöscht
- beim Schreiben in eine Datei bzw. ein Bibliothekselement nicht übernommen
- beim Kopieren von Zeilen nicht kopiert
- durch die Funktionen IEDTGET und IEDTPUT nur dann berücksichtigt, wenn im Kontrollblock EDTAMCB im Feld EAMFLAG das Kennzeichen EAMIGN13 gesetzt ist. Bei den Funktionen IEDTGTM und IEDTPTM wird die Satzmarkierung 13 immer berücksichtigt, unabhängig davon, ob im Feld EAMFLAG das Kennzeichen EAMIGN13 gesetzt ist oder nicht.

Bei der Anweisung @SDFTEST bzw. der Kurzanweisung T sowie bei den Kurzanweisungen J, C, M, R, A, B und O werden Zeilen mit Satzmarkierung 13 ignoriert.

Die Satzmarkierung 14 hat die Sonderfunktion eines Update-Indikators. Derart markierte Zeilen sind im F-Modus überschreibbar dargestellt. Sie bleiben überschreibbar, wenn nur DUE abgeschickt wird oder wenn eine solche Zeile durch eine Anweisung (z.B. der Anweisung @ON, Formate 6 bis 9) geändert wird.
Nur wenn direkt in der Zeile mindestens ein Zeichen eingegeben und mit DUE abgeschickt wird, wird auch die Satzmarkierung 14 gelöscht und die Zeile ist dann auch nicht mehr als überschreibbar dargestellt.

Die Satzmarkierung 15 hat die Sonderfunktion eines Schreibschutz-Indikators. Zeilen mit Satzmarkierung 15 können mit den Kurzanweisungen X, H oder mit F2 im F-Modus-Bildschirmdialog nicht auf überschreibbar gestellt werden.

Die Zeile kann dennoch mit Anweisungen (z.B. der Anweisung @ON, Formate 6 bis 9) geändert werden oder es können an einer mit der Satzmarkierung 15 versehenen Zeile eine oder mehrere nachfolgende Zeilen mit der Kurzanweisung J angehängt werden.
In beiden Fällen bleibt die Satzmarkierung 15 weiter bestehen.

Voraussetzung für die Auswertung der Satzmarkierungen 14 und 15 durch den EDT ist die Einstellung von PROTECTION=ON mit der Anweisung @PAR.

Wird ein Satz mit einer der Markierungen 13, 14 oder 15 am Terminal durch Eingabe im Datenfenster (Eintippen) modifiziert, so werden diese Sondermarkierungen gelöscht.

3.4 Zeichensätze

Mit dem EDT können Texte bearbeitet werden, die in unterschiedlichen Zeichensätzen vorliegen. Im Unicode-Modus können und müssen Daten von einem Zeichensatz in einen anderen konvertiert werden. Damit wird die Funktionalität, im Unterschied zum Kompatibilitätsmodus (siehe Kapitel „Kompatibilitäts-Modus“ auf Seite 633), erheblich erweitert.

In jeder Arbeitsdatei kann ein anderer Zeichensatz eingestellt sein. Es können also in verschiedenen Arbeitsdateien Daten in unterschiedlichen Zeichensätzen bearbeitet werden. Diese Zeichensätze können jederzeit durch die @CODENAME-Anweisung geändert werden. Darüber hinaus hat der EDT einen eigenen Zeichensatz, den Kommunikations-Zeichensatz, in dem er mit der Datensichtstation kommuniziert. Dies kann ein anderer Zeichensatz sein, als der Zeichensatz irgendeiner Arbeitsdatei, in der Daten abgelegt werden.

3.4.1 Zeichensätze im BS2000

Im BS2000 werden durch das Softwareprodukt XHCS Zeichensätze bereitgestellt. Standardmäßig gehören dazu:

- 7-Bit-Zeichensätze, wie z.B. ISO646 (Internationaler 7-Bit-Zeichensatz, ASCII), EDF03IRV (Internationale Referenz Version, EBCDIC), EDF03DRV (Deutsche Referenz Version, EBCDIC).
- 8-Bit-Zeichensätze, wie z.B. ISO88591 (Lateinisches Alphabet Nr.1, ASCII), EDF041 (Lateinisches Alphabet Nr.1, EBCDIC), EDF04DRV (Erweiterung von EDF03DRV) usw.
- Die 3 Unicode-Zeichensätze UTF16, UTF8 und UTFE.

XHCS bietet auch die Möglichkeit, benutzerdefinierte Zeichensätze bereitzustellen. Diese Zeichensätze müssen mit allen Attributen versehen werden, die auch die standardmäßig definierten Zeichensätze besitzen, d.h. alle Eigenschaftstabellen müssen vorhanden sein. Ist das nicht der Fall, kann der Zeichensatz im EDT nicht verwendet werden.
Darüber hinaus benötigt der EDT noch eine besondere Eigenschaft der Konversion. Es müssen nämlich alle vorkommenden Zeichen nach UTF16 konvertierbar sein.

Hinweis

Es ist nicht garantiert, dass die Glyphen aller Unicode-Zeichen unterstützt werden, z.B. enthalten die Terminalemulation MT9750 V7 und Spool in OSD V6 nicht den vollen Unicode-Umfang, sondern nur die Zeichen aus den unterstützten ISO 8859 Varianten 1,2,3,4,5,7,9,15.

Für die Namen der Zeichensätze werden im Text die XHCS-Namen verwendet und nicht die vollständigen Namen, also `EDF041` statt EBCDIC.DF.04-1 oder `UTF16` statt UTF-16.

Der Zeichensatz `UTFE` ist ein BS2000-proprietärer Unicode-Zeichensatz, in dem analog `UTF8` die Zeichen in variabel langen Bytefolgen codiert sind. Das besondere daran ist, dass nicht nur alle Zeichen aus `EDF03IRV` sondern auch alle relevanten BS2000-Steuerzeichen mit dem gleichen Code wie bisher in einem Byte codiert sind. Damit ist dieser Zeichensatz nicht nur abwärtskompatibel zu `EDF03IRV`, sondern auch zu den Transportsequenzen bei der Kommunikation mit der Datensichtstation.

Dateien und Bibliothekselemente können im BS2000 in ihrem Katalogeintrag mit einer Angabe über den Zeichensatz versehen werden. Diese Angabe wird von verschiedenen Produkten im BS2000 ausgewertet wie OpenFT, SHOW-FILE und EDT.

Die Kommunikation mit einer Datensichtstation erfolgt immer in einem Zeichensatz. Zuständig für diese Kommunikation ist VTSU. Dabei lässt VTSU zu, dass bei jedem Dialogschritt ein Zeichensatz angegeben werden kann. Allerdings erlässt VTSU dabei einige Restriktionen in Abhängigkeit vom DSS-Modus. Findet der Dialogschritt im 7-Bit-Modus statt, kann ausschließlich `EDF03IRV` verwendet werden. Im 8-Bit-Modus kann nur ein EBCDIC-Zeichensatz angegeben werden, der zu einer ISO-Zeichensatz-Variante kompatibel ist, die von der Datensichtstation unterstützt wird. Unterstützt die Datensichtstation Unicode, kann die Kommunikation in `UTFE` erfolgen.

Beispiel

Die Datensichtstation kann ausschließlich ISO-Zeichensatz-Variante 1 darstellen. Dann kann im `VTSUCB` als Zeichensatz `EDF041` oder `EDF04DRV` angegeben werden. Der Zeichensatz `ISO88591` kann nicht angegeben werden, da es sich um einen ISO-Zeichensatz handelt. Der Zeichensatz `EDF042` kann nicht angegeben werden, da er nicht kompatibel zur ISO-Zeichensatz-Variante 1 ist.

Weitere Informationen zu XHCS und Zeichensätzen siehe [8].

3.4.2 Unterstützte Zeichensätze

Der EDT lässt grundsätzlich nur Zeichensätze zu, die von der aktuellen XHCS-Installation unterstützt werden.
Dies gilt sowohl für den Stapelbetrieb als auch für den Dialogbetrieb.
Der Kommunikationszeichensatz muss mit der Datensichtstation verträglich sein, allerdings kann es ein anderer sein, als der Zeichensatz der Daten.
Die notwendigen Konvertierungen werden über XHCS abgewickelt und auch die Eigenschaften der Zeichen (Groß-/Kleinbuchstaben, Sonderzeichen) werden von XHCS bereitgestellt.
Standardmäßig werden neben den Unicode-Zeichensätzen und allen EBCDIC-Zeichensätzen auch die ISO-Zeichensätze zugelassen.

Behandlung ungültiger Zeichen

In Unicode-Zeichensätzen ist es möglich, dass Bytesequenzen illegal sind. So können z.B. in `UTFE` oder `UTF8` mehrere Mehrbyte-Einleiter hintereinander stehen. Der EDT lehnt die Eingabe solcher, illegaler Bytesequenzen grundsätzlich ab, sei es beim Einlesen von Dateien oder Variablen oder bei Eingaben im Hex-Modus.

In `UTF16` sind dabei nur die Zeichen aus dem Surrogat-Bereich (`xD800-xDFFF`) illegal.

Alle anderen Zeichen d.h. 2-Byte-Sequenzen werden akzeptiert, auch wenn sie nicht am Terminal darstellbar sind. Auch werden besondere Semantiken vom EDT nicht berücksichtigt, z.B., dass ein Zeichen keinen Vorschub auslösen soll. Insbesondere hat auch ein *Byte Order Mark* (BOM) keine Wirkung, sondern wird einfach als Zeichen übernommen.

Bei 7-Bit-Zeichensätzen oder nicht vollständig definierten 8-Bit-Zeichensätzen werden aus Kompatibilitätsgründen alle Bytes akzeptiert und unverändert übernommen. Solche undefinierten Zeichen können aber niemals in einen anderen Zeichensatz konvertiert werden.

Behandlung nationaler 7-Bit-Zeichensätze

Im Unicode-Modus sind alle 7-Bit-Zeichensätze zulässig, die in XHCS definiert sind (zur Behandlung nationaler 7-Bit-Zeichensätze siehe Abschnitt „Der Zeichensatz EDF03DRV“ auf Seite 59).

Welche Zeichensätze aktuell unterstützt werden, kann mit der Anweisung @SHOW CCS abgefragt werden.

3.4.3 Zeichenfolgen

Die Interpretation und Verarbeitung aller Zeichenfolgen erfolgt immer in einem Zeichensatz.

Dies können unterschiedliche Zeichensätze sein. Daten können z.B. in einer Datei in einem Zeichensatz vorliegen, in einem anderen Zeichensatz in der Arbeitsdatei abgelegt werden und in einem weiteren angezeigt werden.
Bei Bedarf wird dann die Zeichenfolge aus dem Quell- in den Ziel-Zeichensatz konvertiert. So kann z.B. eine Datei, die im Zeichensatz EDF03IRV vorliegt, in eine Arbeitsdatei mit Zeichensatz UTF16 eingelesen werden und an einer Unicode-Datensichtstation in UTFE angezeigt werden. Es können dann beliebige (aus der Menge der unterstützten) Zeichen eingefügt werden.
Für ein Protokoll kann die Datei auch im Zeichensatz EDF041 nach SYSLST ausgegeben werden.

In der folgenden Tabelle wird beschrieben wie der EDT den Zeichensatz bestimmt, der jeweils zur Anwendung kommt.

Quelle / Ziel	**Zeichensatz der Zeichenfolge**
Ein-Ausgabe an einer Datensichtstation	Kommunikations-Zeichensatz
Lesen von SYSDTA	Zeichensatz, der SYSDTA zugeordnet ist (wird vom BS2000-Makro GCCSN bereitgestellt). Falls *NONE, wird EDF03IRV verwendet. Ist SYSDTA einer Datensichtstation zugeordnet, wird der Kommunikationszeichensatz verwendet.
Lesen aus Arbeitsdatei	Zeichensatz der Arbeitsdatei
Schreiben in Arbeitsdatei	Zeichensatz der Arbeitsdatei. Falls *NONE, Zeichensatz der Eingabedaten.
Lesen von Zeichenfolgevariablen	Zeichensatz der Zeichenfolgevariablen
Schreiben in Zeichenfolgevariablen	Beim Neuanlegen einer Zeichenfolgevariablen, Zeichensatz des CODE-Operanden. Falls nicht angegeben, Zeichensatz der Zeichenfolge. Falls nicht neu angelegt, Zeichensatz der Zeichenfolgevariablen.
S-Variablen bzw. Jobvariablen	Zeichensatz aus dem CODE-Operanden der Lese-/Schreib-Anweisung. EDF041, falls nicht angegeben.
Ausführen einer @INPUT-Prozedur	Zeichensatz der Datei, die die @INPUT-Prozedur enthält. Falls *NONE, wird EDF03IRV verwendet.
Ausführen einer @DO-Prozedur	Zeichensatz der Arbeitsdatei, die die @DO-Prozedur enthält.

Quelle / Ziel	Zeichensatz der Zeichenfolge
Einfügen aus einer DVS-Datei oder einem Bibliothekselement	Zeichensatz aus dem Katalogeintrag der Datei. Falls *NONE, wird EDF03IRV verwendet.
Schreiben in eine DVS-Datei oder in ein Bibliothekselement	Bei Neuanlegen der Datei, Zeichensatz des CODE-Operanden oder der Arbeitsdatei. Beim Zurückschreiben, Zeichensatz des CODE-Operanden, der Datei oder der Arbeitsdatei
Lesen und Schreiben aus / in POSIX-Datei	Zeichensatz aus dem CODE-Operanden der Lese/Schreib-Anweisung oder der mit @PAR CODE eingestellte Zeichensatz. Standardmäßig EDF041.
Schreiben nach SYSOUT	Zeichensatz, der SYSOUT zugeordnet ist (wird vom BS2000-Makro GCCSN bereitgestellt). Falls *NONE, wird EDF03IRV verwendet. Ist SYSOUT einer Datensichtstation zugeordnet, wird der Kommunikationszeichensatz verwendet.
Schreiben nach SYSLST	Zeichensatz, der SYSLST zugeordnet ist (wird vom BS2000-Makro GCCSN bereitgestellt). Falls *NONE, wird EDF03IRV verwendet.

Bei einigen Anweisungen (z.B. @CREATE, @SETJV) können mehrere Zeichenfolgen angegeben werden, die zunächst in ein Zwischenergebnis verkettet werden.
Haben alle beteiligten Zeichenfolgen den gleichen Zeichensatz, ist dies auch der Zeichensatz des Zwischenergebnisses. Sind Zeichenfolgen mit unterschiedlichen Zeichensätzen beteiligt, ist der Zeichensatz des Zwischenergebnisses UTFE.
Dieses Zwischenergebnis wird anschließend in den Ziel-Zeichensatz konvertiert.

3.4.4 Konvertierung und Ersatzzeichen

Der EDT konvertiert falls notwendig Daten aus jedem zulässigen Zeichensatz in jeden anderen zulässigen Zeichensatz. Sind dabei Zeichen aus den Quell-Daten im Ziel-Zeichensatz nicht vorhanden, wird ein Ersatzzeichen eingesetzt, falls der Benutzer ein solches mit der Anweisung @PAR SUBSTITUTION-CHARACTER definiert hat. Standardmäßig ist kein Ersatzzeichen definiert (SUBSTITUTION-CHARACTER=*NONE).

Bei Ausgaben auf die Datensichtstation im Dialog wird anders als bei anderen Ausgabe-Zielen immer das gerätespezifische Schmierzeichen eingesetzt. Ist kein Ersatzzeichen definiert, wird bei Ausgabe nach SYSOUT/SYSLST für alle Zeichen, die im Ziel-Zeichensatz nicht vorhanden sind, ein Leerzeichen eingesetzt. In allen anderen Fällen wird die Konvertierung abgelehnt, d.h. die jeweilige Anweisung nicht ausgeführt.

Mit der Anweisung @CHECK (Format 2) kann überprüft werden, ob ein Zeilenbereich verlustfrei in einen Zielzeichensatz konvertiert werden kann. Dabei wird nicht nur überprüft, ob alle Zeichen im Zielzeichensatz vorhanden sind, sondern auch, ob eine Längenbeschränkung nicht überschritten wird. Zeichenfolgen können bei der Konvertierung in einen Unicode-Zeichensatz deutlich länger werden.

Anmerkung

XHCS konvertiert nur kompatible Zeichensätze, so dass ggf. der Umweg über einen Unicode-Zeichensatz (diese sind immer kompatibel) genommen wird.

3.4.5 Unicode-Ersatzdarstellung

Im Unicode-Modus erlaubt der EDT die Eingabe von Zeichen, die z.B. im Zeichensatz der Eingabequelle nicht definiert sind, in Form einer Ersatzdarstellung, in der direkt der UTF16-Code des Zeichens spezifiziert wird. Dazu ist mit der Anweisung @PAR ESCAPE-CHARACTER global oder arbeitsdateispezifisch ein Fluchtsymbol zu vereinbaren, das die Ersatzdarstellung einleitet. Standardmäßig erfolgt keine Ersetzung (ESCAPE-CHARACTER=*NONE). Mit dem Operanden DATA-REPLACEMENT der Anweisung @PAR kann der Kontext eingestellt werden, in dem die Ersetzung stattfindet.

Die Ersatzdarstellung wird standardmäßig nur innerhalb von Anweisungen, und dort nur in Literalen ausgewertet (DATA-REPLACEMENT=OFF). Mit DATA-REPLACEMENT=ON kann eingestellt werden, dass sie auch in Dateneingaben erfolgt.

Die Ersatzdarstellung hat die Form specUxxxx, d.h. das Fluchtsymbol wird gefolgt von einem U oder u (für Unicode) und genau 4 Hexadezimalziffern, die den Code des Zeichens spezifizieren. Wenn beispielsweise mit @PAR ESCAPE-CHARACTER='%' das Fluchtsymbol % spezifiziert wurde, kann das griechische Ω in der Form %U03A9 oder auch %u03a9 (Groß- und Kleinschreibung wird nicht unterschieden) eingegeben werden.

Wenn global oder für die aktuelle Arbeitsdatei @PAR ESCAPE-CHARACTER=*NONE vereinbart ist (Voreinstellung), wenn die Ersatzdarstellung formal falsch ist oder dem eingegebenen Code kein gültiges UTF16 Zeichen entspricht, wird die Ersatzdarstellung wie eine normale Zeichenfolge behandelt. Eine Umwandlung der Ersatzdarstellung bei der Dateneingabe im F-Modus erfolgt ebenfalls nicht, wenn die Zeichenfolge der Ersatzdarstellung eine Spaltenposition überschreitet, für die ein Hardware-Tabulator definiert ist.

Wenn zwar ein gültiges UTF16 Zeichen spezifiziert wird, dieses aber nicht in den Ziel-Zeichensatz umcodiert werden kann, wird so verfahren, als ob man das ungültige Zeichen direkt (z.B. über eine entsprechende Tastatur) eingegeben hätte.

Die Interpretation erfolgt unabhängig davon, ob das mittels der Ersatzdarstellung eingegebene Zeichen am Bildschirm darstellbar ist oder nicht. Nicht darstellbare Zeichen werden, wie im vorigen Abschnitt beschrieben, als Schmierzeichen dargestellt.

3.4.6 Kommunikationszeichensatz

Der Kommunikations-Zeichensatz ist der Zeichensatz, den der EDT zum Datenaustausch mit einer Datensichtstation verwendet.

Im Unicode-Modus kann der Kommunikations-Zeichensatz ein anderer Zeichensatz sein, als der Zeichensatz der aktuellen Arbeitsdatei. Eingaben und Ausgaben werden dann bei Bedarf entsprechend konvertiert.

Da VTSU die Schnittstelle zur Datensichtstation ist, kann im Dialogbetrieb der Kommunikations-Zeichensatz nur ein Zeichensatz sein, der von VTSU akzeptiert wird, mit Ausnahme von EDF03DRV an einem 7-Bit-Terminal (siehe Abschnitt „Der Zeichensatz EDF03DRV“ auf Seite 59). Daraus ergeben sich Einschränkungen bei der Wahl des Kommunikations-Zeichensatzes.

Beim Start stellt der EDT den Zeichensatz aus der Terminal-Option `CODED-CHARACTER-SET` als Kommunikations-Zeichensatz ein. Mit der Anweisung @CODENAME (Format 2) kann der Kommunikations-Zeichensatz explizit eingestellt werden. Diese Einstellung bleibt gültig, bis sie mit einer erneuten @CODENAME-Anweisung geändert wird.

Mit @CODENAME *AUTO, TERMINAL kann ein Mechanismus aktiviert werden, mit dem der EDT automatisch versucht, einen möglichst geeigneten Zeichensatz zu wählen. Kommuniziert er mit einer Unicode-fähigen Emulation, stellt er dann `UTFE` als Kommunikations-Zeichensatz ein.

An einer Datensichtstation im 8-Bit-Betrieb, wenn also keine Kommunikation in `UTFE` möglich ist, wird der Kommunikations-Zeichensatz implizit durch den Zeichensatz der im (oberen) Arbeitsfenster angezeigten Arbeitsdatei festgelegt.
Ändert sich dieser Zeichensatz, z.B. wenn der EDT eine andere Arbeitsdatei anzeigt, kann sich auch der Kommunikations-Zeichensatz ändern. Wenn die Arbeitsdatei leer ist und den Zeichensatz `*NONE` hat, nimmt der EDT standardmäßig den Zeichensatz aus der Terminal-Option `CODED-CHARACTER-SET`. Ist dort 7-BIT angegeben, nimmt er `EDF03IRV`, sonst den dort angegebenen Zeichensatz. Hat die im (oberen) Arbeitsfenster angezeigte Arbeitsdatei einen Zeichensatz, der mit der Datensichtstation verträglich ist, wird dieser eingestellt.
Handelt es sich um einen ISO-Zeichensatz, der kompatibel zu einem Zeichensatz ist, der mit der Datensichtstation verträglich ist, wird dieser eingestellt. Ist das nicht der Fall wird `EDF041` eingestellt.

Bei Kommunikation mit einer Datensichtstation im 7-Bit-Betrieb, wird immer der Kommunikations-Zeichensatz `EDF03IRV` eingestellt.

Im L-Modus gelten die gleichen Regeln. Dies gilt auch, wenn von `SYSDTA` gelesen wird (@EDIT ONLY), und `SYSDTA` einer Datensichtstation zugeordnet ist.

3.4.7 Zeichensätze in Arbeitsdateien

Im Unicode-Modus kann jede Arbeitsdatei einen eigenen Zeichensatz besitzen.

Der Zeichensatz einer leeren und nur einer leeren Arbeitsdatei kann auch *NONE sein. Dies ist die Einstellung nach dem Start des EDT und nach dem vollständigen Löschen einer Arbeitsdatei mit @DROP oder @DELETE (Format 2).

Ist die Einstellung *NONE, kann der Zeichensatz entweder implizit festgelegt werden dadurch, dass Daten in die Arbeitsdatei eingefügt werden oder explizit durch eine @CODENAME-Anweisung. Ist der Zeichensatz für eine Arbeitsdatei einmal eingestellt, werden alle Daten, die in diese Arbeitsdatei gelangen, auch in diesen Zeichensatz konvertiert.

Werden Daten in eine leere Arbeitsdatei mit Zeichensatz *NONE eingefügt, so erhält die Arbeitsdatei implizit den Zeichensatz dieser Daten entsprechend der Quelle aus der sie stammen. Wie der EDT den Zeichensatz bestimmt, kann der Tabelle im Abschnitt Zeichenfolgen entnommen werden.

Hat die Arbeitsdatei bereits einen festgelegten Zeichensatz (also nicht *NONE), wird beim Einfügen von Daten in diese Arbeitsdatei eine Konvertierung dieser Daten aus ihrem Quell-Zeichensatz durchgeführt. Unabhängig davon, ob die Arbeitsdatei leer ist oder nicht, werden die Daten beim Einfügen in den Zeichensatz der Arbeitsdatei konvertiert.

Wenn sich Daten in der Arbeitsdatei befinden, bewirkt ein Umschalten des Zeichensatzes mit der @CODENAME-Anweisung, dass diese Daten in den neuen Zeichensatz konvertiert werden.

Mit @CODENAME …,GLOBAL kann der Zeichensatz aller Arbeitsdateien mit einer Anweisung auf den gleichen Wert eingestellt werden.

Für Migrationszwecke bietet die @CODENAME-Anweisung noch eine zusätzliche Leistung. Mit @CODENAME …,FORCE=YES wird der Zeichensatz der Arbeitsdatei umetikettiert. Die Arbeitsdatei erhält dann einen neuen Zeichensatz, die in ihr enthaltenen Daten werden aber nicht konvertiert, sondern bleiben unverändert. Dies kann verwendet werden, um fehlerhaft etikettierte Dateien zu korrigieren. Der FORCE-Operand wirkt nur bei 7-Bit- und 8-Bit-Zeichensätzen.

3.4.8 Einlesen von Dateien

Im Folgenden wird statt Datei **oder** Bibliothekselement nur von **Datei** gesprochen.

Der EDT wertet beim Einlesen einer Datei den Zeichensatz des Katalogeintrags aus.

Wenn die Arbeitsdatei, in die die Datei eingelesen werden soll, noch keinen Zeichensatz hat (*NONE), wird für diese Arbeitsdatei der Zeichensatz der Datei eingestellt und die Datei eingelesen. Hat die Datei den Zeichensatz *NONE, wird EDF03IRV eingestellt.

Hat die Arbeitsdatei schon einen Zeichensatz, wird der Inhalt der Datei beim Einlesen aus dem Zeichensatz des Katalogeintrags in den Zeichensatz der Arbeitsdatei konvertiert. Die Arbeitsdatei muss nicht leer sein. Es können also Dateien mit unterschiedlichen Zeichensätzen in eine Arbeitsdatei eingelesen werden und die Daten liegen anschließend im Zeichensatz der Arbeitsdatei vor. Enthält die Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht unterstützt werden, wird die Datei nicht eingelesen, wenn nicht ein Ersatzzeichen definiert ist, das dann eingesetzt wird. Enthält die Datei eine ungültige Bytefolge (in Unicode-Zeichensätzen möglich), wird die Datei nicht eingelesen.

Wird der im Katalog eingetragene Zeichensatz nicht unterstützt, wird die Datei ebenfalls nicht eingelesen.

Wird die Datei zur realen Bearbeitung geöffnet (siehe Abschnitt „Dateibearbeitung“ auf Seite 137), muss die Arbeitsdatei leer sein und ihr Zeichensatz muss, damit er eingestellt wird, gleich dem Zeichensatz der Datei oder *NONE sein.

Beispiel

```
   1.00 ·····························································
   2.00 ·····························································
   3.00 ·····························································
   4.00 ·····························································
   5.00 ·····························································
   6.00 ·····························································




CODENAME UTFE;COPY FILE=ADRESSEN···························0000.00:00001(00)
```

Einstellen des Zeichensatzes UTFE für die Arbeitsdatei 0, Einlesen der Datei ADRESSEN mit CCS=*NONE.

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN<·····················
   2.00 DUCK     DONALD    WALTSTR.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 ···············································************************


CREATE 0.01 'ÅNGSTRØM ANDERS    STERNWARTE     STOCKHOLM ';--··0001.00:00001(00)
```

Da die Arbeitsdatei den Zeichensatz UTFE hat, können beliebige Zeichen aus dem Unicode-Zeichenvorrat eingegeben werden, falls die Datensichtstation Unicode unterstützt. Dies kann wie hier z.B. durch Anweisungen geschehen.

```
   0.01 ÅNGSTRØM ANDERS    STERNWARTE     STOCKHOLM<·························
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN<·····················
   2.00 DUCK     DONALD    WALTSTR.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 ···············································************************


·····························************************0000.01:00001(00)
```

Es kann auch durch direkte Eingabe der Daten geschehen. Hier wurde **UE** durch **Ü** ersetzt und Zeile 6 **πλάτων** usw. eingefügt.

```
   0.01 ÅNGSTRØM ANDERS    STERNWARTE     STOCKHOLM<·························
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MÜNCHEN<·····················
   2.00 DUCK     DONALD    WALTSTR.8      DISNEYLAND<·························
   3.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG<·····················
   4.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MÜNCHEN<·····················
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MÜNCHEN<·····················
   6.00 πλάτων             Akademeia      Athen<·····························


·····························************************0000.01:00001(00)
```

3.4.9 Schreiben von Dateien

Beim Schreiben von neuen Dateien wird der Zeichensatz der Arbeitsdatei in den Katalog eingetragen.

Beim Schreiben der Arbeitsdatei in eine neue Datei oder in eine existierende Datei mit Zeichensatz *NONE wird der mit dem neuen CODE-Operanden angegebene Zeichensatz oder der der Arbeitsdatei zum Zeichensatz der Datei. Ist dieser Zeichensatz EDF03IRV und hatte die Datei zuvor den Zeichensatz *NONE, bleibt *NONE erhalten.

Sind beim Schreiben in eine existierende Datei mit Zeichensatz ungleich *NONE die Zeichensätze der Datei und der Arbeitsdatei verschieden (dieser Fall tritt ein beim Zurückschreiben einer eingelesenen Datei nach der expliziten Umschaltung des Zeichensatzes mit @CODENAME oder beim Überschreiben einer existierenden Datei mit neuem Inhalt), so kann über einen Operanden in den Anweisungen @CLOSE, @WRITE (Format 1) und @XWRITE ausgewählt werden, mit welchem Zeichensatz geschrieben werden soll. Ist der Operand nicht angegeben, so wird im Stapelbetrieb das Schreiben abgewiesen. Im Dialogbetrieb wird der Anwender gefragt, welcher Zeichensatz verwendet werden soll.

Wird der Datei-Zeichensatz ausgewählt, konvertiert der EDT die Arbeitsdatei vor dem Schreiben. Gibt es dabei Zeichen, die im Datei-Zeichensatz nicht darstellbar sind, und ist kein Ersatzzeichen definiert (siehe @PAR SUBSTITUTION-CHARACTER), dann wird das Schreiben mit einer Fehlermeldung abgelehnt. Der Benutzer kann dann ein Ersatzzeichen definieren oder den Zeichensatz für das Schreiben verändern und die Datei erneut schreiben.

3.4.10 Kopieren zwischen Arbeitsdateien

Mit Hilfe der @COPY-Anweisung, der @MOVE-Anweisung, der @ON-Anweisung bzw. mit Kurzkommandos können Daten von einer Arbeitsdatei in eine andere kopiert werden.

Die Quell-Arbeitsdatei und die Ziel-Arbeitsdatei können unterschiedliche Zeichensätze haben. Dann werden die zu kopierenden Daten aus dem Quell-Zeichensatz in den Ziel-Zeichensatz konvertiert. Sind dabei Zeichen aus den Quell-Daten im Ziel-Zeichensatz nicht vorhanden, wird die Übertragung abgelehnt, falls der Benutzer nicht ein Ersatzzeichen definiert hat.

3.4.11 Zeichensatz einer Anweisung

Die Anweisungs-Analyse wird immer in UTFE durchgeführt.
Die Anweisungen werden aus dem Zeichensatz, in dem sie eingelesen wurden, nach UTFE konvertiert. Dies ist immer mögich. Damit ist z.B. gewährleistet, dass in allen Zeichensätzen das Zeichen '@' als EDT-Anweisungssymbol verwendet werden kann.
Neben dem EDT-Anweisungssymbol können im EDT auch andere syntaktisch relevante Zeichen umdefiniert werden (Hochkomma und Anführungszeichen mit @QUOTE, das Zeilenbereichssymbol mit @RANGE, Musterzeichen und Füllzeichen mit @SYMBOLS). Um auch hier eine möglichst konsistente Behandlung zu erreichen, wird die Menge der dafür zulässigen Zeichen eingeschränkt. Es können nur die folgenden Zeichen verwendet werden:

!	"	#	$	%	&	'	(	)	*	+
,	-	.	/	:	;	<	=	>	?	@
[	\	]	^	_	`	{	|	}	~	

Damit sind nur die Sonderzeichen aus EDF03IRV zugelassen. Zeichen, die nur in anderen Zeichensätzen Sonderzeichen sind, werden abgelehnt. Für EDF041 z.B. sind dies neben den, an einer normalen Tastatur nicht eingebbaren Zeichen wie ¼, ², ¶ , auch § und ¤.

3.4.12 Der Zeichensatz EDF03DRV

Der Zeichensatz EDF03DRV ist der einzige nationale 7-Bit Zeichensatz, der in XHCS bekannt ist. Er wird aus Kompatibilitätsgründen besonders behandelt, wenn das Terminal als 7-Bit-Terminal erkannt wird.

In diesem Fall erlaubt der EDT auch die Einstellung von EDF03DRV als Kommunikationszeichensatz. Die Zeichen werden aber bei der Übertragung zum und vom Terminal wie EDF03IRV-Zeichen (ohne Konvertierung) behandelt. Die korrekte Darstellung der Zeichen erfolgt durch das Terminal bzw. die Emulation durch entsprechende Einstellungen. Dass diese Einstellung konsistent vorgenommen wird, kann der EDT aber nicht überprüfen. Dies liegt in der Verantwortung des Anwenders.

3.4.13 Zeichenfolgevariablen

Zeichenfolgevariablen können beliebige Texte aufnehmen, wie eine Zeile in einer Arbeitsdatei. Sie sind über Arbeitsdateien hinweg global zugreifbar. Jede Zeichenfolgevariable hat zu jedem Zeitpunkt einen Inhalt, da ihr bei der Initialisierung ein Zeichen '␣' (`X'40'`) zugewiesen wird.

Zeichenfolgevariablen haben immer einen Zeichensatz, da sie auch zu jedem Zeitpunkt einen Inhalt haben. Jede Zeichenfolgevariable kann einen anderen Zeichensatz haben.

Zeichenfolgevariablen können beim Start des EDT durch die S-Variablen `SYSEDT-S00`..`SYSEDT-S20` initialisiert werden. Falls eine oder mehrere dieser S-Variablen vorhanden sind, wird ihr Wert in die korrespondierende Zeichenfolgevariable übernommen. Zu diesem Zeitpunkt kann kein Zeichensatz angegeben werden. Die Zeichenfolgevariablen erhalten daher unabhängig davon, ob sie mit '␣' oder durch S-Variablen initialisiert wurden, als initialen Zeichensatz `EDF041`.

Mit der Anweisung @CREATE kann Zeichenfolgevariablen ein neuer Wert zugewiesen werden. Mit dem Operanden `CODE=name` kann dabei der resultierende Zeichensatz explizit angegeben werden. Ist `CODE` nicht angegeben, ist der resultierende Zeichensatz der Zeichensatz des zuzuweisenden Wertes. Entsteht dieser Wert durch Verkettung von Zeichenfolgen mit unterschiedlichen Zeichensätzen, wird der umfassende Zeichensatz `UTFE` genommen.

Der Zeichensatz von Zeichenfolgevariablen kann auch mit der @CODENAME-Anweisung eingestellt werden. Mit @CODENAME name,#S0 wird der Inhalt von `#S0` in den angegebenen Zeichensatz konvertiert. Der Zeichensatz wird der Zeichenfolgevariablen zugewiesen. Mit @CODENAME name #S0,FORCE=YES kann der Zeichensatz umetikettiert werden (nur bei 7- Bit- und 8-Bit-Zeichensätzen erlaubt).

Mit der Anweisung @SET kann ebenfalls Zeichenfolgevariablen ein Wert zugewiesen werden. Wird dieser Wert als Zeichenfolge spezifiziert, wird immer die Zeichenfolgevariable neu angelegt. Sie erhält dann den Zeichensatz dieser Zeichenfolge.
Entsteht der Wert aus dem Inhalt einer Ganzzahlvariablen, dem Inhalt einer Zeilennummervariablen bzw. einer Zeilennummer oder dem Namen einer Zeichenfolgevariablen in abdruckbarer Form, erhält die Zeichenfolgevariable den Zeichensatz `EDF041`, falls sie neu angelegt wird. Wird nur ein Teil des Inhalts überschrieben, wird das Zwischenergebnis in den Zeichensatz der Zeichenfolgevariablen konvertiert und eingefügt.
Wird ein Wert binär eingefügt, wird die Zeichenfolgevariable immer neu angelegt und sie erhält den Zeichensatz `EDF041`.

3.4.14 S-Variablen und Job-Variablen

Mit den Anweisungen @GETVAR, @GETLIST und @GETJV können die Inhalte von S-Variablen bzw. von Job-Variablen in Zeichenfolgevariablen oder Arbeitsdateien übernommen werden.

Im Unicode-Modus benötigt jede Zeichenfolge eine Angabe über den Zeichensatz. Da S-Variablen bzw. Job-Variablen keine Information über ihren Zeichensatz mitführen, muss der Zeichensatz bei der Übernahme festgelegt werden. Dies kann mit dem neuen Operanden `CODE=name` geschehen. Ist der Operand nicht angegeben, wird der Zeichensatz `EDF041` genommen. Ist das Ziel der Anweisung eine Arbeitsdateizeile, wird der Inhalt der Variablen ggf. aus diesem Zeichensatz in den Zeichensatz der Arbeitsdatei konvertiert. Ist das Ziel eine Zeichenfolgevariable, wird der Wert übernommen und der Zeichenfolgevariablen der Zeichensatz zugewiesen.

Hinweis

Mit @GETVAR SYSEDT,CODE=name kann der Wert der `SYSEDT`-Variablen unter Berücksichtigung eines Zeichensatzes neu in die Zeichenfolgevariablen übernommen werden.

Mit den Anweisungen @SETVAR, @SETLIST und @SETJV können S-Variablen bzw. Job-Variablen erzeugt und ihnen ein Wert zugewiesen werden.

Bei diesen Anweisungen kann mit dem Operanden `CODE=name` ein Zeichensatz angegeben werden, in den die Werte vor der Zuweisung konvertiert werden sollen. Ist dieser Operand nicht angegeben, wird `EDF041` verwendet.

3.4.15 POSIX-Dateien

POSIX-Dateien führen ebenfalls keine Information über den Zeichensatz mit sich.

Bei den Anweisungen, mit denen POSIX-Dateien gelesen oder geschrieben werden, kann mit dem Operanden `CODE=name` angegeben werden, in welchem Zeichensatz die Daten dieser Datei vorliegen bzw. in welchem Zeichensatz sie in die Datei geschrieben werden sollen. Für `name` sind alle unterstützten Zeichensätze zugelassen.

Die Angaben `CODE=ISO`, entspricht `CODE=ISO88591` und `CODE=EBCDIC`, entspricht `CODE=EDF041`, sind aus Kompatibilitätsgründen auch weiterhin möglich.

3.4.16 Ausgaben nach SYSOUT und SYSLST

Ist SYSOUT der Datensichtstation zugeordnet, erfolgt die Ausgabe im Kommunikationszeichensatz und Zeichen, die in diesem nicht darstellbar sind, werden durch das gerätespezifische Schmierzeichen ersetzt.

Ansonsten erfolgen Ausgaben nach SYSOUT bzw. SYSLST im jeweils zugeordneten Zeichensatz, der mit dem BS2000-Makro GCCSN ermittelt wird. Ist dieser *NONE, wird EDF03IRV verwendet. Falls die Ausgabe Zeichen enthält, die im Ziel-Zeichensatz nicht darstellbar sind, und kein Ersatzzeichen definiert wurde, wird ein Leerzeichen eingesetzt.

3.5 EDT-Variablen

Die EDT-Variablen dienen zum Ablegen von Werten. Diese Werte können ganze Zahlen, Zeichenfolgen oder Zeilennummern sein. Innerhalb von EDT-Prozeduren dienen Variablen z.B. zum Zwischenspeichern von Werten, als Schleifenzähler, zur Eingabe von Zeichenfolgen (Dateinamen, Suchbegriffe, u.ä.) oder um einfache Berechnungen durchzuführen.

EDT-Variablen sind nur während des EDT-Laufs gültig, solange der Betriebsmodus (siehe Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21) nicht gewechselt wird. Sie sind global sichtbar, können also von allen Arbeitsdateien aus belegt, benutzt oder abgefragt werden. 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). Eine weitere Möglichkeit besteht darin, EDT-Variablen mit dem Inhalt von Jobvariablen (@GETJV) oder von S-Variablen (@GETVAR, @GETLIST) zu versorgen (siehe unten).

Die Zeilennummervariable #L0 und die Ganzzahlvariablen #I0 bis #I3 sollten nicht verwendet werden, da sie im Fall eines Treffers bei der @ON-Anweisung evtl. mit Werten überschrieben werden.

Jobvariablen und S-Variablen gehören zwar nicht zu den EDT-Variablen, sie werden jedoch in der folgenden Übersicht trotzdem behandelt, weil sie oft verwendet werden, um den Inhalt von EDT-Variablen über die Grenzen eines EDT-Laufs hinweg oder auch permanent zu sichern.

3.5.1 Ganzzahlvariablen

In den Ganzzahlvariablen (#I0..#I20) können positive bzw. negative Ganzzahlen abgelegt werden. Die größtmögliche Zahl ist 2147483647 (2^{31} - 1).

Die Ganzzahlvariablen können über die @SET-Anweisung (Format 1) und mit @GETVAR mit Werten versorgt werden. Mit @STATUS kann der Inhalt der Ganzzahlvariablen am Bildschirm ausgegeben werden. Mit @IF (Format 2) können Werte von Ganzzahlvariablen innerhalb einer EDT-Prozedur ausgewertet werden.

Beim Starten des EDT werden alle Ganzzahlvariablen mit dem Wert 0 vorbelegt.

3.5.2 Zeichenfolgevariablen

In den Zeichenfolgevariablen (#S0..#S20) können Zeichenfolgen in allen vom EDT unterstützten Zeichensätzen abgelegt werden.
Eine Zeichenfolgevariable ähnelt einem Datensatz in einer Arbeitsdatei.
Sie kann ebenso wie ein Datensatz maximal 32768 Zeichen aufnehmen, die mittels Spaltenangaben auch einzeln oder in Abschnitten ansprechbar sind, in Zeichenfolgevariablen kann gesucht werden, Zeichenfolgen können durch andere Zeichenfolgen ersetzt werden usw.

Die Zeichenfolgevariablen können über die @SET-Anweisung, Format 2, 4 und 5, über @CREATE oder mit @GETJV, @GETVAR bzw. @GETLIST mit Werten versorgt werden. Mit @PRINT kann der Inhalt der Zeichenfolgevariablen am Bildschirm ausgegeben werden. Mit @IF (Format 2) können Werte von Zeichenfolgevariablen innerhalb einer EDT-Prozedur ausgewertet werden, dabei können über Spaltenangaben auch Teilbereiche oder einzelne Zeichen angesprochen werden.

Zeichenfolgevariablen werden beim Starten des EDT mit einem Leerzeichen im Zeichensatz EDF041 vorbelegt, wenn sie nicht die Werte der eventuell vorhandenen S-Variablen SYSEDT-S00..SYSEDT-S20 übernommen haben (siehe Abschnitt „Starten des EDT“ auf Seite 91).

Jeder Zeichenfolgevariablen ist ein Zeichensatz zugeordnet, durch den festgelegt wird, wie der Inhalt der Variablen zu interpretieren ist.
Bei den Anweisungen @CREATE und @SET sowie bei @GETJV und @GETVAR kann dieser Zeichensatz explizit für die neu zu füllende Zeichenfolgevariable angegeben werden. Wird kein Zeichensatz spezifiziert, oder werden die Zeichenfolgevariablen schon beim Starten des EDT mit Werten belegt (siehe oben), werden in Abhängigkeit von der Datenquelle und der Systemumgebung Standardwerte für den Zeichensatz eingestellt.

In Anweisungen, in denen die Bezeichnung einer Zeichenfolgevariablen mit einem Dateinamen oder einem Bibliothekselement verwechselt werden kann, muss ein Punkt vor dem Variablennamen angegeben werden, wenn eine Zeichenfolgevariable gemeint ist, z.B. wird mit @OPEN FILE=.#S1 die Datei geöffnet, deren Namen in der Zeichenfolgevariablen `#S1` hinterlegt ist, mit @OPEN FILE=#S1 wird die Datei mit dem Namen '`#S1`' geöffnet.

3.5.3 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. Mit @IF (Format 2) können Werte von Zeilennummervariablen innerhalb einer EDT-Prozedur ausgewertet werden.

Beim Starten des EDT werden alle Zeilennummervariablen mit dem (ungültigen) Wert `0.0000` vorbelegt.

3.5.4 Jobvariablen

In Systemen, in denen das Subsystem JV (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. es kann im EDT auf existierende Jobvariablen zugegriffen werden.

Im EDT kann man Einträge von Jobvariablen löschen (@ERAJV), Werte von Jobvariablen ausgeben, in eine Arbeitsdatei oder in eine Zeichenfolgevariable übertragen (@GETJV), Jobvariablen erzeugen und ihnen Werte zuweisen (@SETJV) und Informationen über Jobvariablen ausgeben bzw. in eine Arbeitsdatei schreiben (@STAJV). Innerhalb von EDT-Prozeduren lässt sich der Inhalt von Jobvariablen nicht direkt auswerten sondern nur nach Übertragung in eine Arbeitsdatei oder in eine Zeichenfolgevariable.

Jobvariablen ist seitens des BS2000 kein Zeichensatz zugeordnet. Beim Lesen einer Jobvariablen mit @GETJV kann daher explizit der Zeichensatz festgelegt werden, in dem der EDT den Inhalt der Jobvariablen interpretieren soll. Wird nichts angegeben, greifen Standardmechanismen, die im Abschnitt Zeichensätze beschrieben sind.

3.5.5 S-Variablen

Im EDT kann auf S-Variablen zugegriffen werden (für S-Listenvariablen muss SDF-P installiert sein). Im Gegensatz zu den Ganzzahl-, Zeichenfolge- und Zeilennummervariablen bleiben S-Variablen auch nach Beenden des EDT bestehen, bzw. es kann im EDT auf existierende S-Variablen zugegriffen werden.

Im EDT kann man Inhalte von S-Variablen vom Typ `STRING` und `INTEGER` ausgeben, in eine Arbeitsdatei, eine Zeichenfolgevariable oder eine Ganzzahlvariable übertragen (@GETVAR), S-Variablen vom Typ `STRING` und `INTEGER` erzeugen und ihnen Werte zuweisen (@SETVAR) sowie S-Listenvariablen (Typ `LIST` mit Elementtyp `STRING`) löschen, erweitern, ihnen Werte zuweisen (@SETLIST) und ihre Werte in eine Arbeitsdatei übertragen (@GETLIST). Innerhalb von EDT-Prozeduren lässt sich der Inhalt von S-Variablen nicht direkt auswerten sondern nur nach Übertragung in eine Arbeitsdatei, eine Zeichenfolgevariable oder eine Ganzzahlvariable.

S-Variablen ist seitens des BS2000 kein Zeichensatz zugeordnet. Beim Lesen einer S-Variablen mit @GETVAR kann daher explizit der Zeichensatz festgelegt werden, in dem der EDT den Inhalt der S-Variablen interpretieren soll. Wird nichts angegeben, greifen Standardmechanismen, die im Abschnitt Zeichensätze beschrieben sind.

3.6 EDT-Prozeduren

Der EDT bietet die Möglichkeit, Folgen von Anweisungen in Arbeitsdateien, katalogisierten Dateien oder Bibliothekselementen abzulegen und bei Bedarf zur Ausführung zu bringen. Neben Anweisungen dürfen auch Datensätze enthalten sein, die dann unter der jeweils aktuellen Zeilennummer in die aktuelle Arbeitsdatei eingefügt werden. Eine derartige Folge von Anweisungen und Datensätzen wird mit dem Oberbegriff EDT-Prozedur bezeichnet.

Abhängig vom Ablageort und der zum Starten benutzten Anweisung unterscheidet man bei den EDT-Prozeduren zwischen @DO-Prozeduren und @INPUT-Prozeduren.

@DO-Prozeduren

– sind in einer Arbeitsdatei (1..22) des EDT abgelegt,
– können nur insgesamt ausgeführt werden,
– erlauben Parameterübergabe, Schachtelung, (bedingte) Sprünge und Schleifen und
– werden über die EDT-Anweisung @DO gestartet.

@INPUT-Prozeduren

– sind in einer Datei abgelegt,
– können insgesamt oder in Teilen ausgeführt werden,
– erlauben keine Parameterübergabe, Schachtelung, Sprünge und Schleifen,
– können bereits beim Starten des EDT als EDT-Startprozedur oder
– über die EDT-Anweisung @INPUT gestartet werden.

In den folgenden Abschnitten werden zunächst die Konzepte erläutert, die für alle Prozedurtypen gültig sind. Dabei geht es um grundlegende Regeln beim Erstellen und Ausführen von EDT-Prozeduren. Dann werden die Möglichkeiten bei @INPUT-Prozeduren und bei der Integration von EDT-Prozeduren in BS2000-Systemprozeduren besprochen. Als spezieller Typ einer @INPUT-Prozedur wird die EDT-Startprozedur behandelt. Schließlich werden die Sprachmittel vorgestellt, die nur im Zusammenhang mit @DO-Prozeduren Anwendung finden können, nämlich Sprünge, Schleifen und Parameter.

3.6.1 Erstellen und Ausführen von EDT-Prozeduren

Beim Erstellen und Ausführen von EDT-Prozeduren sind nachfolgende Regeln zu beachten.

Lesen von Anweisungen und Datenzeilen

Der EDT liest beim Ausführen einer EDT-Prozedur Anweisungen und Datenzeilen im L-Modus. Wird die EDT-Prozedur im F-Modus angestartet, wechselt der EDT in den L-Modus, führt die Prozedur aus und wechselt anschließend wieder in den F-Modus.

Für die Unterscheidung zwischen Anweisungen und Datenzeilen gelten daher die gleichen Regeln wie für die Eingaben im L-Modus (siehe Abschnitt „L-Modus“ auf Seite 131). Insbesondere werden Zeilen mit zwei aufeinander folgenden (nur evtl. durch Leerzeichen getrennten) EDT-Anweisungssymbolen bzw. Benutzeranweisungssymbolen als Datenzeilen interpretiert. Dies erlaubt, innerhalb von Prozeduren dynamisch andere Prozeduren aufzubauen und diese gleich auszuführen (siehe Beispiele weiter unten).

Erlaubte Anweisungen

Mit Ausnahme der Anweisungen @DIALOG, @DROP und @INPUT sind innerhalb von EDT-Prozeduren alle Anweisungen erlaubt, die auch im L-Modus erlaubt sind (siehe Abschnitt „L-Modus“ auf Seite 131).
In @DO-Prozeduren sind zusätzlich die Anweisungen @GOTO, @DO (Format 2) und @PARAMS erlaubt. In @INPUT-Prozeduren wird die Anweisung @IF ... GOTO stillschweigend ignoriert, wenn die Bedingung nicht erfüllt ist, und wird mit der Fehlermeldung `EDT4942` abgewiesen, wenn die Bedingung erfüllt ist.

Aktuelle und aktive Arbeitsdatei, spezielle Arbeitsdateien

Die aktuelle Arbeitsdatei wird im L-Modus mit der @PROC- oder der @SETF-Anweisung, im F-Modus mit der @SETF-Anweisung oder durch Eingabe in der Anweisungszeile des entsprechenden Arbeitsfensters festgelegt. Beim Ausführen von EDT-Prozeduren wirken die gelesenen Anweisungen und Datensätze stets auf die aktuelle Arbeitsdatei. In einer @DO-Anweisung darf daher die aktuelle Arbeitsdatei nicht spezifiziert werden.
Der Versuch wird mit der Meldung `EDT4906` abgewiesen.

Als *aktive* Arbeitsdateien werden diejenigen Arbeitsdateien bezeichnet, die eine @DO-Prozedur enthalten, die gerade ausgeführt wird. Wenn @DO-Prozeduren geschachtelt werden, können mehrere (bis zu 22) Arbeitsdateien aktiv sein, nämlich diejenigen, die mit @DO aktiviert und noch nicht mit @RETURN wieder verlassen wurden.

Bei der Schachtelung von @DO-Prozeduren darf auch eine aktive Arbeitsdatei in @DO-Anweisungen angegeben werden.

Rekursive Aufrufe sind also möglich, jedoch ebenfalls nur bis zu einer maximalen Schachtelungstiefe von 23. Eine *aktive* Arbeitsdatei darf nicht zur *aktuellen* Arbeitsdatei gemacht werden. Eine @PROC- oder @SETF-Anweisung für eine aktive Arbeitsdatei wird daher mit der Meldung `EDT4959` zurückgewiesen.

Grundsätzlich können @DO-Prozeduren in jeder Arbeitsdatei, mit Ausnahme von Arbeitsdatei 0, also in den Arbeitsdateien 1 bis 22, abgelegt und zum Ablauf gebracht werden. Ein @DO 0 wird mit der Meldung `EDT3209` abgewiesen.
Da etliche EDT-Anweisungen Ausgaben standardmäßig in die Arbeitsdatei 9 schreiben, empfiehlt es sich, diese Arbeitsdatei nicht für @DO-Prozeduren zu verwenden.

EDT-Prozeduren und Zeichensätze

Beim Ausführen einer EDT-Prozedur werden Anweisungen und Datensätze in dem Zeichensatz der jeweiligen EDT-Prozedur gelesen.
Das ist bei @INPUT-Prozeduren der Zeichensatz der katalogisierten Datei bzw. des Bibliothekselements, bei @DO-Prozeduren der Zeichensatz derjenigen aktiven Arbeitsdatei, aus der gerade gelesen wird. Die gelesenen Datensätze und die in den Anweisungen enthaltenen Literale oder textartigen Ausdrücke müssen evtl. noch umcodiert werden, falls sie sich auf Objekte (z.B. die aktuelle Arbeitsdatei oder eine Zeichenfolgevariable) beziehen, denen ein anderer Zeichensatz zugeordnet ist. Die genauen Regeln dafür siehe Abschnitt „Zeichensätze“ auf Seite 48.

Beispiel: Erstellen und Aufruf einer @DO-Prozedur im F-Modus

```
  1.00 ·····························································
  2.00 ·····························································
  3.00 ·····························································
  4.00 ·····························································
 23.00 ·····························································
22·····························································0000.00:00001(00)
```

Es wird in die Arbeitsdatei 22 gewechselt. In ihr wird die Prozedur erstellt.

```
  1.00 @COPY FILE=TESTDATEI<·····································
  2.00 @PAR LOWER=ON<·····························································
  3.00 ···························································
  4.00 ···························································
 23.00 ···························································
0·····························································0001.00:00001(22)
```

Die Anweisungen werden in der Arbeitsdatei 22 erstellt. Anschließend wird wieder in die Arbeitsdatei 0 zurückgekehrt.

```
  1.00 ···························································
  2.00 ···························································
 23.00 ···························································
@do 22·····························································0000.00:00001(00)
```

Aus der Arbeitsdatei 0 wird die Prozedur in Arbeitsdatei 22 mit @DO aufgerufen.

```
  1.00 Dies ist eine Testdatei,<··································
  2.00 die mit einer Prozedur<····································
  3.00 in die Arbeitsdatei 0 eingelesen wurde.<···················
  4.00 ···························································
  5.00 ···························································
 23.00 ···························································
·····························································0001.00:00001(00)
```

Das Ergebnis des Prozedurlaufes ist in Arbeitsdatei 0 zu sehen.

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) EDT-Anweisungen werden in die Arbeitsdatei `1` geschrieben (@DO-Prozedur). Dabei wird der Mechanismus der L-Modus-Eingabe ausgenutzt, dass Zeilen mit zwei EDT-Anweisungssymbolen als Datenzeilen in die Arbeitsdatei eingefügt und vor dem zweiten Anweisungssymbol abgeschnitten werden (siehe Abschnitt „L-Modus“ auf Seite 131).

(03) Es wird in die Arbeitsdatei `0` zurückgekehrt.

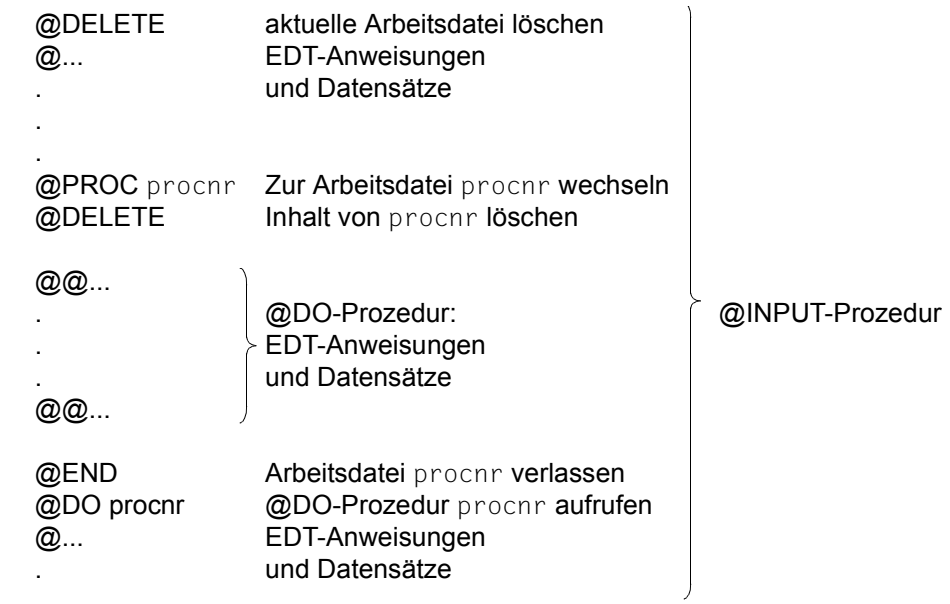
(04) Aufruf der @DO-Prozedur. Die Anweisungen in der Arbeitsdatei werden ausgeführt.

3.6.2 @INPUT-Prozeduren

EDT-Anweisungen und Datensätze können als @INPUT-Prozeduren in eine SAM-, ISAM- bzw. POSIX-Datei oder in ein Bibliothekselement zulässigen Typs (siehe Abschnitt „Dateibearbeitung“ auf Seite 137) geschrieben werden.

Die Vorteile einer @INPUT-Prozedur (permanente Verfügbarkeit, sprechende Namen der Prozedur) lassen sich mit den Vorteilen der @DO-Prozedur (Schachtelung, Sprünge und Schleifen, Parametrisierung) dadurch kombinieren, dass innerhalb der @INPUT-Prozedur eine oder mehrere @DO-Prozeduren dynamisch aufgebaut und zum Ablauf gebracht werden. Dabei wird die Möglichkeit der L-Modus-Eingabe mit verdoppeltem EDT-Anweisungssymbol genutzt.

Aufbau einer @DO-Prozedur innerhalb einer @INPUT-Prozedur



Beispiel

```
  1.00          @NOTE---------------------------------------DATEINAME EINLESEN
  2.00 @CREATE #S00 READ 'DATEINAME: '<·····································
  3.00          @NOTE-------------------------------ARBEITSDATEI 22 AUFKLAPPEN
  4.00 @PROC 22<·····························································
  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 22 ZUKLAPPEN
 17.00 @END<·····
 18.00 @DO 22<·····
 19.00 ·····
 20.00 ·····
 21.00 ·····
 22.00 ·····
 23.00 ·····
@write file=sicherungskopie.input···························0001.00:00001(00)
```

Die Prozedur wird im F-Modus in der Arbeitsdatei 0 erstellt und als SAM-Datei unter dem Namen SICHERUNGSKOPIE.INPUT gespeichert.

Die @DO-Prozedur besteht aus allen Anweisungen mit zwei aufeinander folgenden, nur durch Leerzeichen getrennten Anweisungssymbolen (@ @ - siehe weiter unten).

```
 20.00 ·····························································
 21.00 ·····························································
 22.00 ·····························································
% EDT0172 FILE 'SICHERUNGSKOPIE.INPUT' CREATED AND WRITTEN
@input file=sicherungskopie.input;22·····························0001.00:00001(00)
```

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 22 abgelegt (siehe weiter unten) und gleich ausgeführt. Anschließend wird in die Arbeitsdatei 22 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 ·····························································
·····························································0001.00:00001(22)
```

Die Anweisungen der @INPUT-Prozedur mit mehr als einem Anweisungssymbol wurden in der Arbeitsdatei 22 als @DO-Prozedur abgelegt, dabei wurde der Inhalt vor dem zweiten Anweisungssymbol gelöscht.

3.6.3 Aufruf einer EDT-Prozedur in einer BS2000-Systemprozedur

Eine EDT-Prozedur kann auch innerhalb einer BS2000-Systemprozedur dynamisch aufgebaut und aufgerufen werden.

Beispiel für eine EDT-Prozedur in einer BS2000-Systemprozedur

```
/SET-PROCEDURE-OPTIONS DATA-ESCAPE-CHAR=*STD
/BEGIN-PARAMETER-DECLARATION
/DECLARE-PARAMETER DATEI,TYPE=STRING,INITIAL-VALUE=*PROMPT
/END-PARAMETER-DECLARATION
/SHOW-FILE-ATTRIBUT &DATEI ------------------------------------------------(01)
/MODIFY-JOB-SWITCHES ON=(4,5) ---------------------------------------------(02)
/START-EDT ----------------------------------------------------------------(03)
@PROC 20 ------------------------------------------------------------------(04)
@ @READ '&DATEI'
@ @PAR LOWER=ON
@ @PAR SCALE=ON -----------------------------------------------------------(05)
@ @PAR INFORMATION=ON
@ @PAR EDIT FULL=ON
@END ----------------------------------------------------------------------(06)
@DO 20 --------------------------------------------------------------------(07)
@DIALOG -------------------------------------------------------------------(08)
@HALT ---------------------------------------------------------------------(09)
/SET-JOB-STEP
/MODIFY-JOB-SWITCHES OFF=(4,5) --------------------------------------------(10)
```

(01) Prüfen, ob die über den Prozedur-Parameter spezifizierte Datei vorhanden ist. Wenn nicht, wird zu `SET-JOB-STEP` verzweigt.

(02) Auftragsschalter 4 und 5 setzen (siehe Abschnitt „Auftragsschalter“ auf Seite 102).

(03) EDT aufrufen.

(04) Es wird in die Arbeitsdatei `20` gewechselt.

(05) Die EDT-Anweisungen der @DO-Prozedur werden in der Arbeitsdatei `20` abgelegt.

(06) Es wird in die Arbeitsdatei `0` zurückgekehrt.

(07) Aufruf der @DO-Prozedur, die in Arbeitsdatei `20` steht (Einlesen einer Datei, Unterscheidung zwischen Klein- und Großbuchstaben, Ausgeben eines Spaltenzählers, Ausgeben einer Informationszeile, Datenfenster und Markierungsspalte auf überschreibbar stellen).

(08) Umschalten in den F-Modus-Dialog. Nach Beenden des Dialogbetriebs mit @HALT oder @RETURN wird der Lauf der Systemprozedur an der Stelle fortgesetzt, wo er unterbrochen wurde.

(09) Beenden des EDT.

(10) Rücksetzen der Auftragsschalter.

3.6.4 EDT-Startprozedur

Die EDT-Startprozedur ist eine spezielle @INPUT-Prozedur, die bereits beim Starten des EDT ausgeführt wird (siehe Abschnitt „Starten des EDT“ auf Seite 91). Die EDT-Startprozedur wird gemäß folgender Suchhierarchie bestimmt:

1. Falls der Linkname `$EDTPAR` vergeben wurde, wird die mit ihm verknüpfte Datei als EDT-Startprozedur verwendet und die Suche beendet.
2. Falls unter der Kennung des Aufrufers des EDT eine Datei namens `EDTSTART` existiert, wird diese verwendet und die Suche beendet.
3. Falls bei der Installation des EDT vom Systemverwalter die logische Identifikation `SYSDAT.EDTSTART` einer existierenden und zugreifbaren Datei zugewiesen wurde, wird diese Datei verwendet und die Suche beendet.
4. Falls die Datei `$.EDTSTART` unter der Standard-Benutzerkennung existiert und zugreifbar ist, wird diese verwendet und die Suche beendet.
5. Falls mit 1 bis 4 keine Datei gefunden wurde, wird keine EDT-Startprozedur ausgeführt.

Bei jedem Aufruf des EDT kann also mittels `/SET-FILE-LINK` eine individuelle EDT-Startprozedur eingestellt werden. Insbesondere kann man mit
`/SET-FILE-LINK FILE-NAME=*DUMMY,LINK-NAME=$EDTPAR`
erreichen, dass gar keine EDT-Startprozedur, also auch nicht die vom Systemverwalter eingerichtete, ausgeführt wird.

3.6.5 Unbedingte und bedingte Sprünge

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.

Mit der @IF-Anweisung mit angegebener @GOTO-Anweisung kann in einer @DO-Prozedur in Abhängigkeit von einer Bedingung verzweigt werden. Ist die Bedingung erfüllt, so wird auf die Zeile verzweigt, die in der @GOTO-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 @INPUT-Prozedur sind Sprünge nicht erlaubt.

Beispiel für unbedingte und bedingte Sprünge

```
@PROC 2
@DELETE
@1.00 ------------------------------------------------------------------ (01)
@ @CONTINUE *** AB HIER ZEILENNUMMER 1.00 ***
@ @CREATE #S1 READ 'BITTE SUCHBEGRIFF EINGEBEN: ' ------------------- (02)
@ @ON & FIND #S1 MARK 5
@ @IF .TRUE. @GOTO 2 ---------------------------------------------- (03)
@ @CREATE #S2: 'KEIN TREFFER GEFUNDEN'
@ @PRINT #S2
@ @GOTO 3 --------------------------------------------------------- (04)
@2.00
@ @CONTINUE *** AB HIER ZEILENNUMMER 2.00 ***
@ @DELETE MARK 5
@ @ON & PRINT #S1 ------------------------------------------------- (05)
@3.00
@ @CONTINUE *** AB HIER ZEILENNUMMER 3.00 ***
@END
```

(01) Mit der Anweisung @1.00 wird in der Arbeitsdatei `2` die Zeilennummer `1.00` und implizit die Schrittweite `0.01` eingestellt. Das Gleiche gilt sinngemäß für die anderen @SET-Anweisungen.

(02) Mit @CREATE...READ wird die Eingabe eines Suchbegriffs vom Benutzer angefordert. In der Folgezeile werden alle Zeilen, die den Suchbegriff enthalten, mit der Satzmarkierung `5` gekennzeichnet.

(03) Mit @IF wird überprüft, ob es Treffer gibt, falls ja wird zu Zeile `2.00` verzweigt.

(04) Im Falle, dass es keine Treffer gab, wird die entsprechende Meldung ausgegeben und zum Ende der Prozedur verzweigt.

(05) Im Trefferfall wird die Satzmarkierung gelöscht und die Trefferzeilen werden ausgegeben.

Die Zeilen mit zwei aufeinander folgenden Anweisungssymbolen werden wieder in die Arbeitsdatei `2` in die über die @-Anweisung bestimmten Zeilen eingelesen. Die EDT-Prozedur lässt sich anschließend mit @DO 2 ausführen.

3.6.6 Äußere und innere Schleifen

Mit äußeren Schleifen können @DO-Prozeduren mehrmals vollständig durchlaufen werden. Sollen nur Teile einer Prozedur mehrmals durchlaufen werden, ist dies nur in Form von inneren Schleifen möglich.

Die äußere Schleife wird dadurch realisiert, dass in der @DO-Prozedur auf einen Schleifenzähler Bezug genommen wird, für den beim Aufruf der Prozedur in der @DO-Anweisung Startwert, Endwert und Schrittweite angegeben werden muss (siehe @DO-Anweisung Seite 296). Der Schleifenzähler muss ein Sonderzeichen sein und sollte nicht mit Sonderzeichen kollidieren, die im EDT eine feste Bedeutung haben (z.B. `%` oder `$`), geeignet sind hingegen `!` oder `|`. Innerhalb der @DO-Prozedur muss der Schleifenzähler wie eine Zeilennummer verwendet werden (nicht wie eine Zeilennummervariable, d.h. er kann innerhalb der Prozedur nicht verändert werden).

Der Schleifenzähler wird nach Abarbeitung der letzten Anweisung innerhalb der @DO-Prozedur um die angegebene Schrittweite erhöht bzw. erniedrigt und mit dem Endwert verglichen. Wenn der Endwert noch nicht über- bzw. unterschritten wurde, wird die Prozedur mit dem veränderten Wert des Schleifenzählers erneut durchlaufen. Wird die letzte Anweisung der @DO-Prozedur nicht ausgeführt, etwa weil die Prozedur mit @RETURN verlassen worden ist, wird die Prozedur nicht erneut durchlaufen sondern evtl. vor Erreichen des angegebenen Endwertes abgebrochen. Ggf. ist also eine künstliche (leere) letzte Anweisung zu schreiben (siehe @CONTINUE-Anweisung).

Das Sonderzeichen, das den Schleifenzähler repräsentiert, kann bei der Eingabe der @DO-Anweisung in einem anderen Zeichensatz vorliegen als innerhalb der aufgerufenen Prozedur. Es wird ggf. nach den gleichen Regeln umcodiert wie Literale, die in anderen EDT-Anweisungen vorkommen können (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Äußere Schleifen können durch innere Schleifen ersetzt werden. In einer äußeren Schleife kann neben Start- und Endwert nur eine feste positive oder negative Schrittweite angegeben werden. In einer inneren Schleife kann die Schrittweite variabel, z.B. über eine Zeilennummervariable angegeben werden.

Beispiel für eine äußere Schleife

```
@PROC 3
@DELETE
@ @COLUMN 10 ON ! INSERT !:27-36:
@END
```

Durch die obige Anweisungsfolge wird in Arbeitsdatei 3 eine @DO-Prozedur aufgebaut, die nur eine Anweisung @COLUMN... enthält.

Wird diese @DO-Prozedur mit @DO 3, !=11,15 gestartet, wird für den Schleifenzähler ! nacheinander der Wert 11,12,13,14 und 15 eingesetzt (implizite Schrittweite ist 1). In diesen Zeilen wird der Inhalt der jeweiligen Zeile von Spalte 27-36 an der Spalte 10 nochmals eingefügt.

Wenn diese Prozedur auf eine Arbeitsdatei mit kleinerer Schrittweite (etwa 0.1) angewendet wird, bleiben also evtl. Zeilen unberücksichtigt. Sollen unabhängig von der Schrittweite alle Zeilen berücksichtigt werden, ist eine innere Schleife zu verwenden (siehe folgendes Beispiel).

Beispiel für eine innere Schleife

```
@PROC 4 ------------------------------------------------------------- (01)
@DELETE
@RESET
@1.00
@ @IF ERRORS : @GOTO 2 ---------------------------------------------- (02)
@ @IF #L10 > 15 @GOTO 2 --------------------------------------------- (03)
@ @COLUMN 10 ON #L10 INSERT #L10:27-36: ----------------------------- (04)
@ @SET #L10 = #L10 + 1L --------------------------------------------- (05)
@ @GOTO 1 ----------------------------------------------------------- (06)
@2.00
@ @CONTINUE
@END....

@SET #L10 = 11 ------------------------------------------------------ (07)
@DO 4
```

(01) In Arbeitsdatei 4 wird eine @DO-Prozedur aufgebaut.

(02) Bei EDT-Fehlern soll die Prozedur abgebrochen werden.

(03) Wenn der Schleifenzähler #L10 den Wert 15 überschreitet, soll abgebrochen werden.

(04) In der über #L10 bestimmten Zeile wird der Inhalt der jeweiligen Zeile von Spalte 27-36 an der Spalte 10 nochmals eingefügt.

(05) Die Zeilennummer wird auf die nächste existierende Zeilennummer gesetzt. Würde man hier statt 1L den Wert 1 einsetzen, hätte man den gleichen Effekt wie bei der äußeren Schleife (siehe oben).

(06) Es wird zum Anfang der Schleife gesprungen.

(07) Der Anfangswert des Schleifenzählers wird außerhalb der @DO-Prozedur gesetzt, dann wird die Prozedur aufgerufen.

Hinweis

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.

3.6.7 Parameter

Bei der Erstellung von @DO-Prozeduren im EDT können über @PARAMS formale Parameter definiert werden, denen beim Aufruf der Prozedur mit @DO aktuelle Werte (Aktualparameter) zugewiesen werden.

Die @PARAMS-Anweisung muss als erste Anweisung in einer @DO-Prozedur stehen und darf nur einmal in der Prozedur vorkommen. Es sind Stellungs- und Schlüsselwortparameter erlaubt. Alle Stellungsparameter müssen vor den Schlüsselwortparametern definiert werden.

Ein formaler Parameter beginnt mit dem Zeichen `&`. Ihm folgt ein Buchstabe, dem bis zu 6 weitere Buchstaben oder Ziffern folgen können.

Beim Aufruf der Prozedur werden die Parameter in der @DO-Anweisung als Aktualparameter angegeben. Schlüsselwortparameter können auch innerhalb der @PARAMS-Anweisung mit einem Standardwert versorgt werden.

Der Standardwert wird verwendet, wenn der entsprechende Schlüsselwortparameter in der @DO-Anweisung nicht angegeben wird. Beim Ablauf der Prozedur werden die formalen Parameter innerhalb der Prozedur durch die Werte der Aktualparameter bzw. durch die Standardwerte ersetzt.

Die Verarbeitung der Parameter ist als zweistufiger Prozess zu sehen. Zunächst findet innerhalb der aufgerufenen Prozedur eine Textersetzung der formalen Parameter durch die Aktualparameter statt, dann werden die so geänderten Zeilen verarbeitet. Dabei sind evtl. mehrere unterschiedliche Zeichensätze zu berücksichtigen, der Zeichensatz der Anweisung (für die aktuellen Parameter), der Zeichensatz der als Prozedur auszuführenden Arbeitsdatei und der Zeichensatz der aktuellen Arbeitsdatei, auf die die Anweisungen der Prozedur angewendet bzw. in die die Datensätze eingefügt werden.

In der ersten Stufe der Textersetzung muss daher vom Zeichensatz der Anweisung in den Zeichensatz der auszuführenden Prozedur umcodiert werden.
Dies gilt sowohl für die angegebenen Aktualparameter wie für die Namen der formalen Schlüsselwortparameter. Enthalten die Aktualparameter eine Ersatzdarstellung für Unicode-Zeichen (siehe Abschnitt „Unicode-Ersatzdarstellung“ auf Seite 53), wird diese bei der Textersetzung noch nicht in das entsprechende Unicode-Zeichen umgewandelt, selbst wenn der Aktualparameter in Hochkommas eingeschlossen wurde.

In der zweiten Stufe der Verarbeitung (Ausführung) müssen Datenzeilen sowie Literale innerhalb von Anweisungen vom Zeichensatz der Prozedur in den der aktuellen Arbeitsdatei umcodiert werden, dabei erfolgt dann auch die Interpretation einer Ersatzdarstellung für Unicode-Zeichen, sofern die Ersatzdarstellung innerhalb eines Literals verwendet wurde.

Ein unverändertes Durchschleusen der Aktualparameter ist nicht möglich, weil die Textersetzung an beliebiger Stelle in einer Zeile der auszuführenden Prozedur (in Literalen, in anderen Operanden, sogar im Anweisungsnamen selbst) stattfinden kann.

Wenn in einer Unicode-Umgebung mit einer Prozedurdatei gearbeitet wird, die in einem 7-Bit- oder 8-Bit-Zeichensatz vorliegt, können bei der Umcodierung eines in Unicode eingegebenen Parameters in den Zeichensatz der Prozedurdatei also unerwünschte Zeichenersetzungen stattfinden. Dies lässt sich nur verhindern, wenn von der Ersatzdarstellung für Unicode-Zeichen Gebrauch gemacht wird (siehe oben). Die genauen Regeln für die Umcodierung findet man im Abschnitt „Zeichensätze“ auf Seite 48.

Hinweis

Wenn EDT-Prozeduren mit Parametern innerhalb von ebenfalls parametrisierten BS2000-Systemprozeduren verwendet werden sollen, empfiehlt es sich, zur Vermeidung von Konflikten das BS2000-Parametersymbol auf einen Wert ungleich `&` zu setzen (`/SET-PROC-OPT DATA-ESCAPE-CHAR=...`).

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: '~' * 50
 7.     @ @CREATE 0.02: 'SPEISEPLAN ','&SUCH'
 8.     @ @CREATE 0.03: '~' * 50
 9.     @ @RENUMBER
10.     @ @CREATE $+1: '~' * 50
11.     @ @PRINT
12.     @ @END
13.     @END
 1.     @DO 4 (SPEISEPLAN,KW 49) -------------------------------------- (02)
 1.0000 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 2.0000 SPEISEPLAN KW 49
 3.0000 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 4.0000 KW 49 - 05.12. BOHNENEINTOPF, SCHOKOLADENCREME
 5.0000 KW 49 - 06.12. HACKBRATEN, SALZKARTOFFELN, QUARKSPEISE
 6.0000 KW 49 - 07.12. SCHNITZEL WIENER ART, POMMES FRITES, ROTE GRUETZE
 7.0000 KW 49 - 08.12. MAULTASCHEN IN BRUEHE, GOETTERSPEISE
 8.0000 KW 49 - 09.12. ROTBARSCHFILET, KARTOFFELSALAT, VANILLEPUDDING
 9.0000 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
10.     @DO 4 (SPEISEPLAN,SCHNITZEL) ---------------------------------- (02)
 1.0000 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 2.0000 SPEISEPLAN SCHNITZEL
 3.0000 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 4.0000 KW 45 - 10.11. JAEGERSCHNITZEL, BRATKARTOFFELN, TUTTI FRUTTI
 5.0000 KW 49 - 07.12. SCHNITZEL WIENER ART, POMMES FRITES, ROTE GRUETZE
 6.0000 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

(01) Definieren der symbolischen Parameter (zwei Stellungsparameter).

(02) Aufruf der Prozedur mit den jeweiligen Aktualparametern. Die Formalparameter in der @READ-, @ON-Anweisung werden bei jedem @DO-Aufruf durch die aktuellen Werte ersetzt.

3.7 Suchen mit @ON

Es gib zehn Formate der @ON-Anweisung, in denen abhängig von einem Suchbegriff Aktionen ausgelöst werden. Der Suchbegriff definiert eine oder mehrere Zeichenfolgen, nach denen in einem Suchbereich gesucht wird.

Neben einfachen Zeichen kann der Suchbegriff Musterzeichen enthalten, die Platzhalter für Zeichengruppen sind. Die Musterzeichen stehen entweder für genau ein Zeichen oder für eine beliebig lange Zeichenfolge. Wenn die Musterzeichen interpretiert werden, findet beim Suchen eine Mustererkennung statt.

Der Suchbegriff ist sowohl links als auch rechts von einem Begrenzersymbol eingeschlossen. Es gibt zwei verschiedene Begrenzersymbole.

Abhängig vom jeweiligen Begrenzersymbol wird der Beginn bzw. das Ende der Trefferzeichenfolge entweder nur durch den Suchbegriff oder zusätzlich noch durch Textbegrenzerzeichen vor bzw. hinter dem Suchbegriff bestimmt. Die beiden möglichen Begrenzersymbole können bei einer direkten Eingabe des Suchbegriffes beliebig miteinander kombiniert werden. Welche Zeichenfolgen als Treffer gewertet werden, hängt auch von den verwendeten Begrenzersymbolen ab.

Falls Musterzeichen interpretiert werden oder falls im Suchobjekt nach Textbegrenzerzeichen gesucht wird, sind der Suchbegriff und die Trefferzeichenfolgen ggf. nicht identisch. Auch die einzelnen Trefferzeichenfolgen können sich in diesem Falle ggf. voneinander unterscheiden.

Der Suchbegriff wird nur in den Zeilen- bzw. Spaltenbereichen gesucht, die in der @ON-Anweisung angegeben werden. Im Folgenden werden die Details aufgeführt, die beim Suchen mit der @ON-Anweisung relevant sind.

3.7.1 Berücksichtigung von Groß-/Kleinschreibung

Mit der @SEARCH-OPTION-Anweisung kann festgelegt werden, ob bei der Suche nach Zeichenfolgen mit der @ON-Anweisung die Groß-/Kleinschreibung zu berücksichtigen ist. Standardmäßig wird bei der Suche nach Groß- und Kleinbuchstaben unterschieden. Ist die Groß-/Kleinschreibung nicht zu beachten, werden der Suchbegriff und der Suchbereich zur Anweisungsausführung temporär von Klein- nach Großbuchstaben umcodiert.

Die Umwandlung erfolgt mit der von XHCS angebotenen Schnittstelle. Beim Umcodieren von Klein- nach Großbuchstaben ändern sich unabhängig vom eingestellten Zeichensatz die Längen der Zeichenfolgen nicht.

Beispiel

@SEARCH-OPTION CASELESS-SEARCH=OFF	Die Groß-/Kleinschreibung eines Zeichens wird bei der Treffersuche beachtet. Dies entspricht der Voreinstellung der@SEARCH-OPTION-Anweisung.
@ON & C'suCH' TO 'SUCH'	Nur die Zeichenfolge 'suCH' wird in 'SUCH' umgewandelt. Zeichenfolgen der Form 'such' oder 'sucH' werden nicht ersetzt.
@SEARCH-OPTION CASELESS-SEARCH=ON	Bei der Treffersuche wird die Groß-/Kleinschreibung nicht beachtet.
@ON & C'suCH' TO 'SUCH'	Alle Zeichenfolgen 'such' in allen möglichen Groß/Kleinschreibungsvariationen werden in 'SUCH' umgewandelt.

Die Einstellung von CASELESS-SEARCH wirkt nur für die Formate 1 bis 3 und 5 bis 10. Beim Format 4 ist sie nicht relevant, da dort nur nach markierten Zeilen gesucht wird.

3.7.2 Verwendung von Musterzeichen im Suchbegriff

Neben einfachen Zeichen können auch Platzhalter für Zeichengruppen, so genannte Musterzeichen, angegeben werden. Es gibt zwei Musterzeichen.

asterisk (Standardwert `*`) steht für eine beliebig lange, auch leere Zeichenfolge. Wird es mehrmals nebeneinander angegeben, wird es wie ein einziger asterisk ausgeführt, z.B. ist '`ABC**F`' gleichwertig mit '`ABC*F`'.

slash (Standardwert /) steht für genau ein Zeichen.

Ist das Schlüsselwort `PATTERN` angegeben, werden die Musterzeichen interpretiert und es findet eine Mustererkennung (Pattern Matching) statt. Die Musterzeichen werden durch die kürzest mögliche Teilfolge des Suchbereiches befriedigt.

Ist das Schlüsselwort `PATTERN` nicht angegeben, werden die Musterzeichen 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 Musterzeichen verwendet werden. Ein nur aus Musterzeichen bestehender Suchbegriff ist ebenfalls zulässig. Mit der Anweisung @SYMBOLS können die Musterzeichen umdefiniert werden.

Innerhalb der Ersatzdarstellung von Unicode-Zeichen dürfen keine Musterzeichen angegeben werden.

3.7.3 Negatives Suchen

Ist das Schlüsselworte `NOT` in der @ON-Anweisung angegeben, werden die Sätze bzw. Zeichenfolgevariablen ermittelt, die den Suchbegriff nicht enthalten. Bei den einzelnen Formaten der @ON-Anweisung ist dies im Detail erläutert.

Beispiel

Die Arbeitsdatei enthalte die folgenden Zeilen:

```
1.ABCD
2.ABCE
3.ABDE
4.ACDE
```

Mit @ON & PRINT NOT 'AB' wird nur die 4. Zeile ausgegeben.

Mit @ON & PRINT NOT 'ABC' werden die 3. und die 4. Zeile ausgegeben.

3.7.4 Begrenzersymbole

Bei der Eingabe ist ein Suchbegriff sowohl links als auch rechts von einem Begrenzersymbol eingeschlossen. Es gibt zwei Begrenzersymbole.

apostrophe (Standardwert ') legt fest, dass im Suchbereich nicht nach Textbegrenzerzeichen vor bzw. hinter dem Suchbegriff gesucht wird. Der Beginn bzw. das Ende der Trefferzeichenfolge wird also nur durch den Suchbegriff bestimmt.

quotation mark
(Standardwert `"`) bestimmt, dass im Suchbereich die Trefferzeichenfolge durch ein Textbegrenzerzeihen vor bzw. hinter dem Suchbegriff begrenzt sein muss. Der Beginn bzw. das Ende der Trefferzeichenfolge werden also durch Textbegrenzerzeichen oder durch die erste bzw. die letzte Spalte des zu untersuchenden Spaltenbereiches festgelegt.

Die Einstellungen für das Begrenzersymbol *apostrophe* und für das Begrenzersymbol *quotation mark* können mit der Anweisung @QUOTE geändert werden.

Welche Zeichenfolgen als Treffer gewertet werden, hängt auch von der Art der verwendeten Begrenzersymbole ab. Wenn der Suchbegriff durch die Begrenzersymbole *apostrophe* eingeschlossen ist, dann wird das Auffinden des Suchbegriffes im Suchbereich als Treffer gewertet.
Das Musterzeichen *asterisk* hat keine Bedeutung, wenn es im Suchbegriff unmittelbar neben dem Begrenzersymbol *apostrophe* steht.

Ist das linke Begrenzersymbol des Suchbegriffes *quotation mark*, dann muss dieser im Trefferfalle entweder am Zeilenanfang stehen oder sich unmittelbar vor ihm ein Textbegrenzerzeichen befinden.

Steht das Musterzeichen *asterisk* im Suchbegriff unmittelbar hinter dem Begrenzersymbol *quotation mark*, dann reicht die Trefferzeichenfolge bis zum nächsten Textbegrenzerzeichen vor dem Suchbegriff. Gibt es kein Textbegrenzerzeichen, dann reicht die Trefferzeichenfolge bis zum Zeilenanfang.

Ist das rechte Begrenzersymbol des Suchbegriffes *quotation mark*, dann muss dieser im Trefferfalle entweder am Zeilenende stehen oder sich unmittelbar hinter ihm ein Textbegrenzerzeichen befinden. Steht das Musterzeichen *asterisk* im Suchbegriff unmittelbar vor dem Begrenzersymbol *quotation mark*, dann reicht die Trefferzeichenfolge bis zum nächsten Textbegrenzerzeichen hinter dem Suchbegriff.
Gibt es kein Textbegrenzerzeichen, dann reicht die Trefferzeichenfolge bis zum Zeilenende.

Der Satz der Textbegrenzerzeichen ist durch den EDT standardmäßig mit den Zeichen Leerzeichen (`X'40'`) sowie `+.!*();-/,?:'="` vorbelegt.
Diese Zeichenmenge kann über die Anweisung @DELIMIT umdefiniert werden.

Will man in einem Suchbegriff nach den Begrenzersymbolen *apostrophe* oder *quotation mark* suchen, so müssen diese verdoppelt werden.

Die erste bzw. die letzte Spalte eines angegebenen Spaltenbereiches wirken bei der Suche nach Textbegrenzerzeichen wie ein Zeilenanfang bzw. ein Zeilenende.

Die Begrenzersymbole eines Suchbegriffes können beliebig kombiniert werden.

Beispiel 1

Die Arbeitsdatei enthalte die folgenden Zeilen:

```
1.ABCD
2.A,BCD
3.ABC,D
4.A,BC,D
```

Mit `@ON & PRINT 'BC'` wird in allen 4 Zeilen 1 Treffer festgestellt.
Mit `@ON & PRINT "BC'` wird nur in der 2. und 4. Zeile 1 Treffer festgestellt.
Mit `@ON & PRINT 'BC"` wird in der 3. und 4. Zeile 1 Treffer festgestellt.
Mit `@ON & PRINT "BC"` wird nur in der 4. Zeile 1 Treffer festgestellt.

Beispiel 2

Die Arbeitsdatei enthalte die folgende Zeile:

```
1.XXX,ABCDEFGH-YYY
@ON 1 PRINT PATTERN 'EFG*"
```

Die Trefferzeichenfolge ist '`EFGH`', da sie nach rechts bis zum nächsten Textbegrenzerzeichen `'-'` reicht.

```
@ON 1 PRINT PATTERN "*BCD'
```

Die Trefferzeichenfolge ist 'ABCD', da sie nach links bis zum nächsten Textbegrenzerzeichen ',' reicht.

Beispiel 3

```
@ON & PRINT 'Diese ""Zeichenfolge"" hat kein ''X''.'
```

Gesucht wird in diesem Falle nach der Zeichenfolge

```
Diese "Zeichenfolge" hat kein 'X'.
```

3.7.5 Indirekte Angabe des Suchbegriffes

Der Suchbegriff kann in der @ON-Anweisung auch über eine Zeilennummer, eine Zeilennummervariable oder eine Zeichenfolgevariable (jeweils Spaltenangabe möglich) spezifiziert werden. Die Zeile mit der angegebenen Zeilennummer bzw. die Zeichenfolgevariable enthält den Suchbegriff.

Beispiele

```
@CREATE 6 : 'AB*C//D'
@ON 2-3 PRINT PATTERN 6:2-5:
```

Der Suchbegriff ist somit die Zeichenfolge 'B*C//'.

```
@CREATE 1 : 'ABCDEFG'
@SET #L3 = 1
@ON 2-3 PRINT PATTERN #L3:4-7:
```

Der Suchbegriff lautet also 'DEFG'.

```
@SET #SO = 'ABCD*E//F'
@ON & PRINT PATTERN #SO:3-8:
```

Der Suchbegriff ist somit die Zeichenfolge 'CD*E//'.

Der Suchbegriff wird bei indirekter Angabe so behandelt, als wäre er zwischen den Begrenzersymbolen *apostrophe* eingeschlossen. Das Suchen nach Textbegrenzerzeichen in Suchbereichen ist bei indirekten Eingabe also nicht möglich.

3.7.6 Suchbereich

Die @ON-Anweisung sucht nur im angegebenen Suchbereich.

Es können hierbei einer oder mehrere Zeilenbereiche spezifiziert werden. Ein Zeilenbereich kann auch aus nur einer einzelnen Zeile bestehen. Die Zeilenbereiche werden in der angegebenen Reihenfolge durchsucht. Bei einem Zeilenbereich kann auch ein Bereich von Zeichenfolgevariablen angegeben werden.

Mittels eines Operanden kann gesteuert werden, ob die @ON-Anweisung in jedem angegebenen Zeilenbereich nur die jeweils erste Trefferzeile ermittelt oder alle Zeilen jedes Zeilenbereiches durchsucht. Standardmäßig durchsucht die @ON-Anweisung alle Zeilen jedes angegebenen Zeilenbereiches.

Beispiele

```
@ON 1-3 FIND 'ABC'
```

Bei der @ON-Anweisung wird nur ein einzelner Zeilenbereich durchsucht.

```
@ON 1,2,3 FIND 'ABC'
```

Der Suchbereich besteht also aus 3 Zeilenbereichen mit jeweils einer einzelnen Zeile. Das Suchergebnis kann operandenabhängig anders sein als im vorhergehenden Beispiel.

```
@ON #S01,#S03.-#S04 CHANGE 'ABC' TO 'DEF'
```

Bei der @ON-Anweisung werden die Zeichenfolgevariablen `#S01`, `#S03` und `#S04` nach einem Treffer durchsucht.

Für das Suchen innerhalb einer Zeile kann ein Spaltenbereich angegeben werden. Die Angabe mehrerer Spaltenbereiche ist nicht gestattet. Ist kein Spaltenbereich angegeben, wird ein Standard-Spaltenbereich (siehe @SEARCH-OPTION) verwendet.

Beispiele

```
@ON &:15-15: CHANGE 'A' TO 'D'
```

Bei der @ON-Anweisung wird somit nur eine einzelne Spalte untersucht.

```
@ON &:15-25: PRINT 'XYZ'
```

Bei der @ON-Anweisung wird somit in einem zusammenhängenden Spaltenbereich gesucht.

3.7.7 Sonstige Suchparameter

Der Suchbereich wird operandenabhängig entweder von links nach rechts oder umgekehrt durchsucht (R-Operand). Standardmäßig durchsucht der EDT die Zeilen von links nach rechts.

Beim Suchen innerhalb einer Zeile kann durch Operandeneingabe gesteuert werden, ab dem wievielten Vorkommen ein Suchbegriff als Treffer gewertet wird. Standardmäßig ist bereits das erste Auffinden des Suchbegriffes in einer Zeile ein Treffer.
Nach dem Auffinden des Suchbegriffes führt die @ON-Anweisung in Abhängigkeit vom jeweiligen Format bestimmte Aktionen aus.

Anschließend wird operandengesteuert (ALL-Operand) ggf. die Suche nach weiteren Treffern in einer Zeile fortgesetzt. Hierbei berücksichtigt die @ON-Anweisung, dass sich die einzelnen Trefferzeichenfolgen nicht überlappen dürfen.
Dies gilt auch dann, wenn die Trefferzeichenfolgen variabel lang sind, weil das Musterzeichen *asterisk* interpretiert wird oder weil nach Textbegrenzerzeichen gesucht wird.
Bei der Suche von links nach rechts setzt die @ON-Anweisung deshalb hinter der Trefferzeichenfolge auf. Bei der Suche von rechts nach links ist die weitere Treffersuche auf diejenigen Spalten beschränkt, die vor der Trefferzeichenfolge liegen.
Ab der Suche nach dem zweiten Treffer in einer Zeile wird operandenunabhängig jedes weitere Auffinden des Suchbegriffes als Treffer gewertet. Standardmäßig ermittelt die @ON-Anweisung nur den ersten Treffer innerhalb einer Zeile.

3.7.8 Festhalten eines Treffers

Der EDT hält das Ergebnis der Treffersuche in lokalen Schaltern bzw. in Variablen fest.

Mit @IF (Format 3) kann abgefragt werden, ob bei der Ausführung der letzten @ON-Anweisung ein Treffer festgestellt wurde oder ob die aktuelle Arbeitsdatei leer war. Im Trefferfalle kann zusätzlich noch die Spaltennummer abgefragt werden, in der die erste Trefferzeichenfolge beginnt.

Außerdem hält der EDT in zugreifbaren Variablen die Ergebnisse des Suchvorganges fest.

Die Nummer der Zeile, in der der EDT den ersten Treffer feststellt, wird in der Zeilennummervariablen #L0 und unter dem Zeilennummern-Symbol '?' festgehalten. Wird kein Treffer gefunden oder 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 (unabhängig ob Zeilen oder Zeichenfolgevariable), wird in der Ganzzahlvariablen #I0 gespeichert; die Nummer der Spalte, in der sie endet, in der Ganzzahlvariablen #I1.
Wird kein Treffer festgestellt, bleiben die Werte von #I0 und #I1 unverändert.
Das gilt auch für einen in einer Zeichenfolgevariablen ermittelten Treffer.

Mit @PRINT #L0:#I0-#I1 kann man die Trefferzeichenfolge auf dem Bildschirm ausgeben lassen, wenn der Treffer in einer Zeile lag. Wenn die Operanden V und ALL angegeben sind, werden in der Ganzzahlvariablen #I2 die Anzahl der Trefferzeilen bzw. die Anzahl der Zeichenfolgevariablen, die den Suchbegriff enthalten, und in der Ganzzahlvariablen #I3 die Gesamtanzahl der Treffer gespeichert.
Spaltenangaben bezeichnen unabhängig vom eingestellten Zeichensatz Zeichen- und keine Byteadressen.

Bei negativer Suche wird nach dem Auffinden des ersten Satzes, in dem der Suchbegriff nicht vorkommt, in der Ganzzahlvariablen #I0 die Anfangsposition und in der Ganzzahlvariablen #I1 die Endposition des untersuchten Spaltenbereiches gespeichert.

Wenn die Endposition des Spaltenbereichs größer als die Satzlänge ist, wird in #I1 die Satzlänge gespeichert.

Beispiel

Die Arbeitsdatei enthalte eine Zeile mit der Zeichenfolge 'XXX–ABCD–YYY'.

Suchbegriff	**Trefferstring**	**Inhalt von #I0 und #I1**
'ABC*Y“	ABCD–YYY	#I0 = 5; #I1 = 12
'ABC*Y'	ABCD–Y	#I0 = 5; #I1 = 10
"ABCD'	ABCD	#I0 = 5; #I1 = 8
'*BCD'	BCD	#I0 = 6; #I1 = 8

4 Ablauf des EDT

In diesem Abschnitt wird die Einbettung des EDT in die Systemumgebung des BS2000 beschrieben. Dazu gehören die grundlegenden Abläufe beim Starten, Beenden, Unterbrechen und Überwachen des EDT, eine Übersicht über die Ein- und Ausgabeströme sowie die Darstellung der Möglichkeiten, das System gegen unerwünschte Zugriffe über den EDT zu schützen.

4.1 Starten des EDT

Da der EDT ab V17.0A in zwei Betriebsmodi betrieben werden kann (siehe Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21), wurden die Kommandos zum Starten des EDT entsprechend erweitert.

Man kann sich bereits beim Starten des EDT dafür entscheiden, ob man der Kompatibilität den Vorzug gibt und auf die Funktionserweiterungen verzichtet - d.h. den EDT im Kompatibilitäts-Modus startet, oder ob man die neue Funktionalität nutzen will und dafür gewisse Inkompatibilitäten in Kauf nimmt, d.h. den EDT im Unicode-Modus startet.

Ein Wechsel zwischen dem Kompatibilitäts-Modus und dem neuen Unicode-Modus ist auch im Nachhinein über EDT-Anweisungen möglich.
Ein Starten im Unicode-Modus ist immer dann zwingend erforderlich, wenn man bereits bei der EDT-Initialisierung, z.B. beim Abarbeiten der EDT-Startprozedur (siehe „EDT-Startprozedur“ auf Seite 74) von den Funktionserweiterungen Gebrauch machen will.
Einzelheiten zu den Funktionserweiterungen und Inkompatibilitäten des Unicode-Modus sowie zum Umschalten per EDT-Anweisung sind im Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21 und im Kapitel „Kompatibilitäts-Modus“ auf Seite 633 beschrieben.
Im Folgenden wird der Startvorgang im Unicode-Modus beschrieben.

Zum Starten des EDT im Kompatibilitäts-Modus siehe Kapitel „Kompatibilitäts-Modus“ auf Seite 633.

4.1.1 Startkommando des EDT

Ab EDT V17.0A wird der EDT im Unicode-Modus mit dem Kommando /START−EDTU geladen und gestartet.
Das /START−EDTU-Kommando ermöglicht die Auswahl einer bestimmten EDT-Version bei Koexistenz mehrerer Versionen. Für /START−EDTU kommen dabei nur EDT-Versionen größer oder gleich V17.0A in Betracht.
Das Kommando /START−EDTU darf nur in Kennungen mit bestimmten Privilegien eingegeben werden (siehe Abschnitt „Zugriffsschutz“ auf Seite 103).

Der Alias-Name für das /START−EDTU-Kommando ist /EDTU.

START-EDTU	Kurzname: **EDTU**
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 /SELECT−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 des Software-Produkts JV [9]). Ausführliche Beschreibung siehe Abschnitt „Überwachung des EDT-Laufs mit Monitor-Jobvariablen“ auf Seite 100.

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 Dialogbetrieb standardmäßig im F-Modus (Full-Screen-Modus, siehe Abschnitt „F-Modus“ auf Seite 105) und im Stapelbetrieb im L-Modus (Line-Modus, siehe Abschnitt „L-Modus“ auf Seite 131) gestartet.

Bei gesetztem Auftragsschalter 5 (siehe Abschnitt „Auftragsschalter“ auf Seite 102) wird auch im Dialogbetrieb der L-Modus eingestellt. Der EDT liest in diesem Fall die Eingaben mit RDATA von SYSDTA.

Die Arbeitsweise des EDT wird auch durch die Vereinbarung eines Benutzer-Standardzeichensatzes mit dem Kommando /MODIFY-TERMINAL-OPTIONS bzw. durch den für SYSDTA eingestellten Zeichensatz beeinflusst (siehe dazu die Abschnitte „Einführung zu den Betriebsmodi des EDT“ auf Seite 21 sowie „Zeichensätze“ auf Seite 48).

Während der Startphase des EDT laufen folgende Initialisierungsschritte ab:

1. Übernahme von mittels SDF-P definierten S-Variablen in Zeichenfolgevariablen des EDT (s.u.)
2. Ausführen der EDT-Startprozedur (siehe Abschnitt „EDT-Startprozedur“ auf Seite 74)

Die Abarbeitung erfolgt in der angegebenen Reihenfolge. Beim Start des EDT als Unterprogramm unterbleiben diese Initialisierungsschritte.

Beim Start des EDT werden die Zeichenfolgevariablen #S00..#S20 initialisiert. Falls eine oder mehrere S-Variablen SYSEDT-S00..SYSEDT-S20 existieren und ihr jeweiliger Wert den Typ STRING hat, wird ihr Inhalt der korrespondierenden Zeichenfolgevariablen zugewiesen. S-Variablen mit anderem Typ werden nicht übernommen.

Da zu diesem Zeitpunkt kein Zeichensatz angegeben werden kann, erhalten die Zeichenfolgevariablen den Zeichensatz EDF041.

4.1.2 Aufruf des EDT als Hauptprogramm

Aus Kompatibilitätsgründen wird das Starten des EDT mittels /START−PROGRAM weiterhin unterstützt. Der EDT wird dann als Hauptprogramm mit einem der folgenden BS2000-Kommandos geladen und im Unicode-Modus gestartet:

Kommando	AMODE
START-PROGRAM $.SYSPRG.EDT.170.EDTU	AMODE 31
START-PROGRAM *MODULE ($.SYSLNK.EDT.170,EDTCU, RUN-MODE=*ADVANCED)	AMODE 24

4.1.3 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“ [1] beschrieben.

Besonderheiten beim Zusammenspiel von Unicode-Modus und EDT als Unterprogramm sind in Abschnitt „Unterprogrammschnittstellen und Betriebsmodi“ auf Seite 639 beschrieben.

4.2 Unterbrechen und Beenden des EDT-Laufs

Die folgenden beiden Abschnitte beschreiben Besonderheiten des EDT bei Unterbrechung bzw. Beendigung.

4.2.1 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.

Werden während der Unterbrechung über die Kommando-Schnittstelle des BS2000 andere Programme geladen (z.B. mit /START-PROGRAM oder /LOAD-PROGRAM) bzw. werden Prozeduren gestartet, die andere Programme laden, wird der EDT ohne Rückfrage entladen, ein Fortsetzen des EDT-Laufs ist dann nicht möglich.

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. Bei einer Unterbrechung mit K2 gehen alle noch nicht übertragenen Eingaben verloren.

Die Fortsetzung eines unterbrochenen EDT-Laufs ist auch mit dem Kommando /INFORM-PROGRAM möglich. Eine im Kommando mitgegenen Nachricht wird dabei ignoriert.

Hat man im F-Modus in einer Anweisungsfolge oder im L-Modus in einem Eingabeblock (BLOCK-Modus) die Anweisung @SYSTEM angegeben, und kehrt nach der Unterbrechung mit /INFORM-PROGRAM zurück, so wird eine Meldung ausgegeben und der Rest der Anweisungszeile bzw. des Eingabeblocks nach @SYSTEM wird nicht mehr ausgeführt.

Hat der EDT zum Zeitpunkt der Unterbrechung die Zeilen einer @DO- oder @INPUT-Prozedur noch nicht vollständig abgearbeitet, wird bei Rückkehr mit /INFORM-PROGRAM die Verarbeitung abgebrochen. Es wird eine Meldung ausgegeben und die restlichen Zeilen werden nicht mehr abgearbeitet.

Wird zum Zeitpunkt der Unterbrechung ein Zeilenbereich in einer EDT-Anweisung bearbeitet, wird bei Rückkehr mit /INFORM-PROGRAM die Verarbeitung der Anweisung im Allgemeinen abgebrochen und eine Meldung ausgegeben.

Hinweis

Der EDT-Lauf kann nicht unterbrochen werden, wenn der EDT innerhalb einer BS2000-Systemprozedur gestartet wird, die mit der Option INTERRUPT-ALLOWED=NO gegen Unterbrechung geschützt ist (siehe Abschnitt „Zugriffsschutz“ auf Seite 103).

4.2.2 Beenden des EDT-Laufs

Die Anweisungen @HALT, @RETURN (außerhalb von Prozeduren), @EXEC und @LOAD sowie im F-Modus die Taste K1 beenden normalerweise den EDT. Dabei schließt der EDT alle geöffneten Dateien.

Der EDT kann im Dialogbetrieb unter Umständen auch mit @END beendet werden. Im L-Modus werden zuvor die Meldungen

```
%  EDT4939 '@END' WITHOUT '@PROC' STATEMENT
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)?
```

ausgegeben, um unabsichtliches Beenden des EDT zu verhindern.

Befindet sich der EDT im Bildschirmdialog (nach @DIALOG), so beenden die Anweisungen @HALT, @RETURN (außerhalb von Prozeduren), @END und die Taste K1 nicht den EDT, sondern nur den Bildschirmdialog. Dabei erfolgen dann auch keine Sicherheitsabfragen.

Mit @HALT ABNORMAL kann ein nicht normales Beenden des EDT-Laufs erzwungen werden wenn der EDT als Hauptprogramm gestartet wurde. Wurde EDT als Unterprogramm aufgerufen, wird bei @HALT ABNORMAL mit einem speziellen Returncode in das rufende Programm zurückgekehrt.
Existieren beim Beenden noch ungesicherte Arbeitsdateien, wird der EDT im Dialogbetrieb nicht sofort 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` wird der EDT fortgesetzt, der Benutzer kann weiterarbeiten, z.B. noch nicht gesicherte Dateien zurück schreiben. Bei Antwort `Y` gehen die ungesicherten Arbeitsdateien verloren, der EDT wird beendet.

Beim Beenden mittels der Taste K1 im F-Modus wird die Sicherheitsabfrage

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

auch dann in der Meldungszeile des Arbeitsfensters ausgegeben, wenn keine geöffneten Arbeitsdateien existieren.

Bei Beendigung des EDT wird der Inhalt der Zeichenfolgevariablen `#S00..#S20` den korrespondierenden S-Variablen `SYSEDT-S00..SYSEDT-S20` zugewiesen, falls diese existieren und einen Wert vom Typ `STRING` aufnehmen Können. Der Wert der Zeichenfolgevariablen wird dabei im Zeichensatz `EDF041` übergeben. Treten Konvertierungsfehler auf, wird der Wert nicht zugewiesen, ist der Wert länger als 4096 Bytes, wird abgeschnitten und nur die ersten 4096 Bytes werden zugewiesen. Meldungen werden keine ausgegeben. Die

Zuweisung unterbleibt, wenn sie zuvor manuell mit @SETVAR vorgenommen wurde und dabei der Operand KEEP angegeben war, oder der EDT als Unterprogramm gestartet wurde.

Tritt das Ereignis *Überschreitung 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 (Programmfehler) 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. Der EDT wird nach Erstellen eines Speicherabzugs abnormal beendet.

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

Diese Informationen stehen nicht zur Verfügung, wenn der EDT-Lauf mit den Anweisungen @EXEC oder @LOAD oder mit dem BS2000-Kommando /CANCEL-PROGRAM abgebrochen bzw. durch ein anderes BS2000-Kommando entladen wurde.

4.2.3 Kommando-Returncode des EDT

Der EDT liefert 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 des Meldungsgewichts 0, 1, 2 auf (Informationen, Warnungen, keine Syntaxfehler)
5	0	EDT8000	Normale Beendigung des EDT-Laufs. Es trat mindestens eine Meldung des Meldungsgewichts 4 oder 5 auf (Fehler bei Funktion oder Ausführung, keine Syntaxfehler)
10	0	EDT8000	Normale Beendigung des EDT-Laufs. Es trat mindestens eine Meldung des Meldungsgewichts 3 auf (Syntaxfehler in einer Anweisung)
50	64	EDT8101	Abnormale Beendigung durch den Benutzer (@HALT ABNORMAL)
100	64	EDT8200	Abbruch wegen Zeitablauf (Überschreitung der Programmlaufzeit)
100	64	EDT8292	Fehler beim Lesen. Programm abgebrochen
100	64	EDT8293	Fehler beim Schreiben. Programm abgebrochen
150	64	EDT8910	Programmunterbrechung Abnormaler Abbruch mit Speicherabzug
150	64	EDT8001	Abnormale Beendigung nach Programmunterbrechung
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

Zu Meldungen und Meldungsgewichten siehe Abschnitt „Meldungstexte“ auf Seite 664.

Wenn der EDT abnormal beendet wurde, können die Komponenten des Returncodes mit den SDF-P-Funktionen `SUBCODE1()`, `SUBCODE2()` und `MAINCODE()` abgefragt werden.

Wenn der EDT normal beendet wurde, 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 [7]).

Beispiel zum Abfragen von Returncodes

```
/MODIFY-JOB-SWITCHES ON=5
/START-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
```

4.3 Überwachung des EDT-Laufs mit Monitor-Jobvariablen

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

```
/START-EDTU MONJV=jvname
```

Wird beim Starten des EDT eine Monitor-Jobvariable angegeben, so wird diese beim Beenden des EDT gefüllt.

Der Wert der Jobvariablen besteht aus

– einer Zustandsanzeige von 3 Byte Länge,

– einer Rückkehrcode-Anzeige von 4 Byte Länge.

Folgende Tabelle zeigt, wie die Jobvariablen 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 mit 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.

4.4 Ein- und Ausgabe

Der EDT bekommt seine Eingaben

- primär über den Bildschirm
- von der Systemdatei SYSDTA
- von einer SAM- oder ISAM-Datei
- von einem Bibliothekselement
- von einer POSIX-Datei
- von einer anderen Arbeitsdatei des EDT (@DO-Prozedur)

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

Ob der EDT von der Datensichtstation oder von SYSDTA liest, hängt zum einen von der Systemumgebung (Dialog- oder Stapelbetrieb), zum anderen vom Arbeitsmodus des EDT (F-Modus oder L-Modus) ab (siehe Kapitel „Arbeitsmodi des EDT“ auf Seite 105). Die Arbeitsmodi lassen sich ihrerseits durch Auftragsschalter (siehe Abschnitt „Auftragsschalter“ auf Seite 102) und durch Anweisungen (@EDIT-Anweisung) einstellen.

Die übrigen Eingabequellen müssen explizit in einer Anweisung vereinbart werden (siehe die Anweisungen @OPEN, @COPY, @READ, @GET, @XOPEN, @XCOPY, @INPUT, @DO)

Der EDT gibt den Inhalt der Arbeitsdateien aus

- primär auf den Bildschirm
- in die Systemdatei SYSOUT
- in eine SAM- oder ISAM-Datei
- in ein Bibliothekselement
- in eine POSIX-Datei
- in die Systemdatei SYSLST

Ob der EDT auf die Datensichtstation oder nach SYSOUT schreibt, hängt zum einen von der Systemumgebung (Dialog- oder Stapelbetrieb), zum anderen vom Arbeitsmodus des EDT (F-Modus oder L-Modus) ab. Die Arbeitsmodi lassen sich ihrerseits durch Auftragsschalter (siehe Abschnitt „Auftragsschalter“ auf Seite 102) und durch Anweisungen (@EDIT-Anweisung) einstellen.

Die übrigen Ausgabeziele werden direkt oder indirekt über Anweisungen vereinbart (siehe die Anweisungen @CLOSE, @WRITE, @SAVE, @COPY, @XCLOSE, @XWRITE, @PRINT, @LIST).

Einzelheiten finden Sie im Kapitel „Dateibearbeitung“ auf Seite 137.

4.5 Auftragsschalter

Es gibt 5 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.

4.5.1 Auftragsschalter 4

Wurde der Auftragsschalter 4 vor dem Laden des EDT gesetzt, werden nach dem Laden die Meldungen des Binder-/Lader-Systems (BLS05xx), im L-Modus-Dialog die EDT-Startmeldung (EDT0001) und nach dem Beenden des EDT die Meldung

```
%  EDT8000 EDT TERMINATED
```

unterdrückt. Ebenfalls unterbleiben die Meldungen

```
%  EDT0900 EDITED FILE(S) NOT SAVED!
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)?
```

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

4.5.2 Auftragsschalter 5

Ist der Auftragsschalter 5 beim Start 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 * als Eingabeaufforderung aus.

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

Das explizite Umschalten (mit @EDIT ohne Operanden in den L-Modus bzw. mit @EDIT FULL SCREEN in den F-Modus) bleibt weiterhin möglich.

4.5.3 Auftragsschalter 6

Bei Ausgabe nach SYSLST (z.B. @LIST-Anweisung) schreibt der EDT normalerweise nicht mehr als 132 Zeichen pro Zeile. Ist der Auftragsschalter 6 gesetzt, schreibt der EDT bis zu 160 Zeichen pro Zeile. Längere Ausgaben werden auf mehrere Zeilen verteilt.

Der Auftragsschalter 6 muss bereits beim Start des EDT gesetzt sein.

4.5.4 Auftragsschalter 7

Dieser Auftragsschalter kann sowohl vor dem Start des EDT als auch während des EDT-Laufs gesetzt werden. Er verhindert die automatische Freigabe von vorab zugewiesenem, überschüssigem Plattenspeicherplatz nach dem Schreiben einer SAM- oder ISAM-Datei durch den EDT, der im Normalfall nicht belegten Plattenspeicherplatz durch das `FILE`-Makro freigibt (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

4.5.5 Auftragsschalter 8

Im Stapelbetrieb schreibt der EDT Meldungen und die Ausgabe einer Reihe von Anweisungen (z.B.: @STATUS) nach `SYSLST`. Ist der Auftragsschalter 8 gesetzt, schreibt der EDT diese Ausgaben nach `SYSOUT` (siehe Abschnitt „Systemdateien“ auf Seite 156).

Der Auftragsschalter 8 muss bereits beim Start des EDT gesetzt sein.

4.6 Zugriffsschutz

Zwei Mittel stehen zur Verfügung, um das System vor unzulässigen Zugriffen über den EDT zu schützen:

– Der EDT darf nur gestartet werden, wenn die Kennung ein bestimmtes Privileg besitzt
– Schutz durch nicht unterbrechbare BS2000-Systemprozeduren, die den Aufruf von EDT-Anweisungen kontrollieren

4.6.1 Einschränkungen für privilegierte Kennungen

Das Kommando `/START-EDTU` kann in allen Kennungen eingegeben werden, die das Privileg `TSOS` und/oder `STANDARD-PROCESSING` haben. Wenn eine Kennung nur eins oder mehrere der folgenden Privilegien hat, wird der EDT zwar gestartet, sicherheitsrelevante Anweisungen werden jedoch abgewiesen.

Privileg	Bedeutung	Systemkennung
HARDWARE-MAINTAINANCE	Hardware-Online-Wartung	$SERVICE
SECURITY-ADMINISTRATION	Sicherheitsverwaltung	$SYSPRIV
SAT-FILE-MANAGEMENT	Verwaltung von SAT-Dateien	$SYSAUDIT
SAT-FILE-EVALUATION	Auswertung von SAT-Dateien	$SYSAUDIT

Folgende Anweisungen sind bei Kennungen mit diesen Privilegien sicherheitsrelevant:

Anweisung	**Bedeutung**
@EXEC	Programm starten
@LOAD	Programm laden
@RUN	Ablauf eines Benutzerprogramms als Unterprogramm
@SYSTEM	Systemkommandos absetzen
@UNLOAD	Programm entladen
@USE	Externe Anweisungsroutinen definieren

Diese Anweisungen werden in den genannten Kennungen mit der Fehlermeldung EDT4976 abgewiesen.

4.6.2 Nicht unterbrechbare Prozeduren

Wenn BS2000-Systemprozeduren mit der Option INTERRUPT-ALLOWED=NO gegen Unterbrechung durch den Aufrufer geschützt sind, so gilt für den EDT:

– Der Wechsel in den Systemmodus mit K2 ist nicht möglich.

– Wenn EDT-Prozeduren mit K2 abgebrochen werden, fragt der EDT mit der Meldung EDT0913 ab, ob Aktionen durchgeführt werden sollen.

– Im Dialogbetrieb und bei der Eingabe von einer Datei (Lesen mit RDATA von SYSDTA, Abarbeiten einer EDT-Startprozedur) werden die sicherheitsrelevanten Anweisungen @SYSTEM, @EXEC, @RUN, @LOAD, @UNLOAD und @USE abgewiesen, es sei denn, die Anweisungen werden von der geschützten Prozedur selbst gegeben (SYSDTA=SYSCMD).

5 Arbeitsmodi des EDT

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

– Im FULL-SCREEN-Modus (F-Modus) steht in 23 Arbeitsdateien (0–22) 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 dem Anweisungssymbol (standardmäßig @) oder mit dem Benutzeranweisungssymbol (siehe @USE-Anweisung, Seite 569) beginnen.

5.1 F-Modus

Im F-Modus bietet der EDT bildschirmorientierte Dateibearbeitung für SAM- und ISAM-Dateien, für Bibliothekselemente sowie für POSIX-Dateien an. Insgesamt stehen dem Benutzer dafür 23 Arbeitsdateien (0–22) zur Verfügung.

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

– Daten in beliebiger Reihenfolge überschrieben werden können
– Text an beliebiger Stelle in einer Bildschirmzeile ein- und ausgefügt werden kann
– Text am Ende der Datei oder in neu eingefügten Bildschirmzeilen erfasst werden kann

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

– Anweisungen in der Anweisungszeile
– Kurzanweisungen in der Kurzanweisungsspalte
– Anweisungen im Datenfenster (z.B. Auftrennen von Zeilen)
– 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, Bibliothekselementen oder POSIX-Dateien in diese Arbeitsdatei geschrieben wurden.

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

Unterstützte Datensichtstationen

Im F-Modus des EDT spielen naturgemäß die Eigenschaften der verwendeten Datensichtstation (DSS) eine besondere Rolle. Der EDT ist für die DSS 8160, 9750 und aufwärtskompatible Geräte bzw. für die entsprechenden Terminal-Emulationen und deren Eigenschaften konzipiert.

Ein Arbeiten mit Unicode-Dateien ist meist nur dann sinnvoll möglich, wenn eine Terminal-Emulation benutzt wird, die die Eingabe von Unicode-Zeichen erlaubt und diese auch korrekt im Bildschirm-Fenster darstellen kann (z.B. MT9750 ab V7.0 mit DSS-Typ DSS 9763 und DSS-Modus Unicode). Die DSS 3270 wird im Unicode-Modus des EDT nicht mehr unterstützt, kann allerdings im Kompatibilitäts-Modus noch benutzt werden.

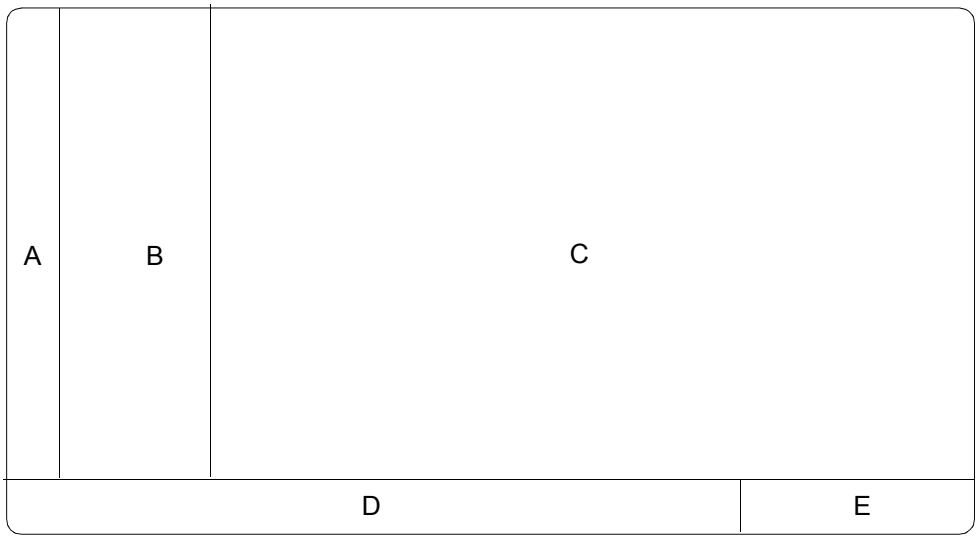
Beim Datenaustausch mit einer Datensichtstation kann für einen Dialogschritt immer nur ein Zeichensatz eingestellt sein. Ein Dialogschritt mit einem veränderten Zeichensatz ändert die Darstellung der DSS global, nicht nur für die gerade übertragenen Daten. Dies führt dazu, dass der Bildschirm gelöscht und neu aufgebaut wird.

Da der EDT die gleichzeitige Bearbeitung von Arbeitsdateien erlaubt, die in unterschiedlichen Zeichensätzen codiert sind, muss es möglich sein, den Zeichensatz der zur Kommunikation mit der Datensichtstation verwendet wird, unabhängig von den gerade sichtbaren Arbeitsdateien festzulegen (insbesondere, wenn die DSS den Zeichensatz der Arbeitsdatei gar nicht unterstützt). Dazu dient die Anweisung @CODENAME (Format 2).

Diese Anweisung ändert nur die Darstellung am Bildschirm und ggf. die Interpretation der Bildschirmeingaben, nicht jedoch den Zeichensatz der Arbeitsdateien oder gar den Zeichensatz der zugrunde liegenden DVS-Datei (siehe dazu auch den folgenden Abschnitt und den Abschnitt „Zeichensätze“ auf Seite 48).

5.1.1 Das Arbeitsfenster

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



A = Kurzanweisungsspalte
A + B = Zeilennummernanzeige
C = Datenfenster
D = Anweisungszeile
E = Zustandsanzeige

Die Darstellung an der Datensichtstation erfolgt grundsätzlich immer in dem für die DSS eingestellten Kommunikationszeichensatz. Dieser kann vom EDT automatisch gewählt werden oder durch den Benutzer explizit mit der Anweisung @CODENAME (Format 2) eingestellt werden (siehe auch Abschnitt „Kommunikationszeichensatz“ auf Seite 54).

Ein mit @CODENAME name, TERMINAL explizit spezifizierter gültiger Zeichensatz wird in jedem Fall eingestellt, unabhängig vom Inhalt der Arbeitsdateien und unabhängig von den für die Arbeitsdateien eingestellten Zeichensätzen. Dieser bleibt eingestellt bis er durch den Benutzer wieder geändert wird oder der Benutzer mit @CODENAME *AUTO, TERMINAL die automatische Zeichensatzwahl des EDT wieder aktiviert.
Beim Start des EDT wird der mittels `/MODIFY-TERMINAL-OPTIONS` vereinbarte Zeichensatz eingestellt.

Wird die automatische Zeichensatzwahl des EDT für die Kommunikation mit der DSS aktiviert, geht der EDT wie folgt vor:

– Wenn die DSS die Darstellung von Unicode-Zeichensätzen unterstützt, wird der Zeichensatz UTFE als Kommunikationszeichensatz eingestellt, auch wenn der mittels /MODIFY-TERMINAL-OPTIONS vereinbarte Zeichensatz davon abweicht.

– Wenn die DSS 8-Bit-Zeichensätze (aber nicht Unicode) unterstützt, wird der mittels /MODIFY-TERMINAL-OPTIONS vereinbarte Zeichensatz eingestellt. Ist dort 7-BIT eingestellt, wird EDF03IRV als Kommunikationszeichensatz eingestellt, andernfalls der dort angegebene Zeichensatz.

– Wenn die DSS nur im 7-Bit Betrieb arbeiten kann, wird EDF03IRV eingestellt.

Die automatische Zeichensatzwahl des EDT für die Kommunikation mit der DSS bewirkt in bestimmten Situationen ein Umschalten des Kommunikationszeichensatzes, um die Darstellung optimal an die angezeigten Inhalte anzupassen. Das Umschalten kann immer dann erfolgen, wenn ein Wechsel der aktuellen Arbeitsdatei erfolgt bzw. wenn die automatische Zeichensatzwahl des EDT für die Kommunikation mit der DSS aktiviert wird:

– Wenn die DSS die Darstellung von Unicode-Zeichensätzen unterstützt, wird sofort (auch ohne Wechsel der Arbeitsdatei) UTFE eingestellt.

– Wenn die DSS die Darstellung von Unicode-Zeichensätzen unterstützt, wird die Einstellung des Kommunikationszeichensatzes beim Wechsel der Arbeitsdatei nicht verändert. Wenn die DSS nur im 7-Bit Betrieb arbeiten kann, erfolgt ebenfalls niemals ein Wechsel des Kommunikationszeichensatzes.

– Wenn die DSS 8-Bit-Zeichensätze (aber nicht Unicode) unterstützt, wird der Zeichensatz eingestellt, der für die im (oberen) Arbeitsfenster angezeigte Arbeitsdatei vereinbart ist, falls er von der DSS unterstützt wird.
  Falls dieser Zeichensatz ein EBCDIC- oder Unicode-Zeichensatz ist und von der verwendeten DSS nicht unterstützt wird, wird EDF041 eingestellt. Falls es sich um einen ISO-Zeichensatz handelt, werden die Daten für den Benutzer transparent nach EBCDIC umgewandelt und der dem ISO-Zeichensatz zugeordnete EBCDIC-Zeichensatz wird eingestellt.
  Falls die im (oberen) Arbeitsfenster angezeigte Arbeitsdatei leer ist und den Zeichensatz *NONE hat, wird der mittels /MODIFY-TERMINAL-OPTIONS vereinbarte Zeichensatz eingestellt.

Der für die Ausgabe eingestellte Zeichensatz legt auch fest, in welchem Zeichensatz die Eingaben an der DSS beim EDT ankommen.
Unabhängig von diesem zur Übertragung verwendeten Zeichensatz müssen die Eingaben jedoch ggf. uminterpretiert werden, je nachdem ob sie global ausgewertet werden (Kurzanweisungen, Dateinamen in Anweisungen etc.) oder sich auf Objekte mit eigenem

Zeichensatz (Arbeitsdateien, Zeichenfolgevariablen etc.) beziehen. Die Regeln für diese Interpretation der Eingaben, insbesondere bei Literalen, sind im Abschnitt „Zeichensätze“ auf Seite 48 beschrieben.

5.1.1.1 Kurzanweisungsspalte

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

In der Standardeinstellung, wenn im Datenfenster Sätze dargestellt wurden, ist die Kurzanweisungsspalte überschreibbar und das Datenfenster vor Überschreiben geschützt. Erst durch Kurzanweisungen in der Kurzanweisungsspalte oder durch Datenübertragung mit F2 werden die Datenfensterzeilen auf überschreibbar gestellt. In den überschreibbaren Bildschirmzeilen können dann keine Kurzanweisungen angegeben werden.

Alternativ zur Standardeinstellung können durch die Anweisung @PAR EDIT-FULL=ON bei eingeschalteter Zeilennummernanzeige (@PAR INDEX=ON) das Datenfenster und die Kurzanweisungsspalte auf überschreibbar gestellt werden. Es ist dann möglich, eine Kurzanweisung einzugeben und gleichzeitig Daten in dieser Bildschirmzeile zu ändern (siehe @PAR EDIT-FULL).

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

5.1.1.2 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 Kurzanweisungsspalte ist, ist die Zeilennummernanzeige nicht überschreibbar.

Die Zeilennummer wird 6-stellig angezeigt. Vier Stellen stehen vor dem Dezimalpunkt, zwei danach, ein nicht überschreibbares Leerzeichen trennt die Zeilennummer von der Datenzeile.

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

5.1.1.3 Datenfenster

Im Datenfenster wird die aktuelle Arbeitsdatei dargestellt. Eine Arbeitsdatei besteht aus Sätzen. Diese Sätze werden in die Bildschirmzeilen 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, sofern ihre Länge nicht die Anzahl der an der DSS darstellbaren Zeichen übersteigt, im EDIT-LONG-Modus vollständig dargestellt werden (siehe @PAR EDIT-LONG).

Enthält die Datei weniger Sätze als das Datenfenster Bildschirmzeilen hat, werden die restlichen Bildschirmzeilen mit Füllzeichen (standardmäßig NIL-Zeichen) aufgefüllt und auf überschreibbar gestellt. Diese Zeilen werden bereits mit der eingestellten Standardschrittweite durchnummeriert. Die gleiche Darstellung wird auch gewählt, wenn das Datenfenster so weit an das Ende der Datei positioniert wurde, dass nur noch weniger Sätze als das Datenfenster Bildschirmzeilen hat, darzustellen sind.

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

Standardmäßig sind die Sätze im Datenfenster nicht überschreibbar. Zum Ändern müssen einzelne Sätze mit den Kurzanweisungen X oder E bzw. alle Datenfensterzeilen 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. Im EDIT-FULL-Modus können daher Kurzanweisungen in der Kurzanweisungsspalte und Eingaben im Datenfenster für die gleiche Zeile gleichzeitig erfolgen (siehe @PAR EDIT-FULL).

Zur Übertragung der Eingaben an der DSS können die Funktionstasten F1 bis F22 sowie DUE und DUE2 verwendet werden. Die Tasten K1 bis K15 übertragen die Eingaben nicht, evtl. eingegebene Texte gehen also verloren. Einige Funktionstasten lösen darüber hinaus im EDT spezielle Aktionen aus, siehe dazu den Abschnitt „Funktionstasten im F-Modus“ auf Seite 128 und „Funktionstasten im L-Modus“ auf Seite 134.

Leerzeilen und neue Zeilen

Die vom EDT zu verarbeitenden Dateien können Sätze der Länge 0 enthalten. Bei POSIX-Dateien oder SAM-Dateien handelt es sich dabei wirklich um Sätze der Länge 0, bei ISAM-Dateien mit Standardeigenschaften um Sätze der Länge 8 bzw. der Länge 16 (bei in UTF16 codierten Dateien).

Um Sätze der Länge 0 im Datenfenster darstellen zu können, kennzeichnet der EDT im Unicode-Modus das Satzende mit dem datenstationsspezifischen Zeichen LZE (Logisches Zeilenende). Der Rest der Bildschirmzeile rechts vom LZE wird von der DSS mit geschützten NIL-Zeichen (X'00') gefüllt. Wenn sich das Satzende außerhalb des Datenfensters befindet, wird LZE nicht dargestellt, die Bildschirmzeile endet dann mit dem letzten noch sichtbaren Zeichen des Satzes bzw. besteht nur aus NIL-Zeichen. Ein Satz der

Länge Null wird also durch eine Bildschirmzeile im Datenfenster dargestellt, die nur aus dem Zeichen LZE in Spalte 1 und geschützten NIL-Zeichen besteht (Leerzeile). Wenn in einer Bildschirmzeile in Spalte 1 LZE eingegeben wird, wird für diese Zeile ein Satz der Länge 0 in der Arbeitsdatei angelegt.

Leerzeilen sind zu unterscheiden von *neuen Zeilen*, die der EDT im F-Modus nach dem letzten Satz der Datei bzw. bei der Bearbeitung der Kurzanweisungen 1..9 oder I anbietet. Diese Bildschirmzeilen entsprechen (noch) keinem Satz in der Arbeitsdatei und bestehen nur aus (überschreibbaren) NIL-Zeichen (X'00') ohne LZE.

Bei der Eingabe kann das Zeichen LZE normalerweise weggelassen werden. Es muss lediglich dann eingegeben werden, wenn der Satz mit NIL-Zeichen enden soll. Der EDT ignoriert bei der Eingabe im F-Modus alle NIL-Zeichen am Ende der eingegebenen Bildschirmzeile, d.h. alle NIL-Zeichen bis zu dem ersten Zeichen ungleich NIL (dies kann ein LZE oder ein anderes Zeichen sein) werden von rechts abgeschnitten. Das LZE selbst wird nicht in den Datensatz übernommen. Da neue Zeilen nur aus NIL-Zeichen bestehen,werden sie bei der Eingabe als Ganzes ignoriert und nicht in die Arbeitsdatei eingefügt. Im Gegensatz dazu würde die Eingabe eines LZE in Spalte n einer neuen Zeile dazu führen, dass nach der Datenübertragung ein Satz mit n−1 Füllzeichen (standardmäßig Blank) in dieArbeitsdatei eingefügt wird. Insbesondere würde für n=1 ein Satz der Länge 0 eingefügt.

Die DSS gestattet keine Eingaben in einer Bildschirmzeile rechts vom Zeichen LZE. Beim Anfügen von Zeichen an eine Zeile ist also entweder der Einfügemodus der DSS einzuschalten oder das Zeichen LZE zu überschreiben.

Behandlung von Füllzeichen im Datenfenster

Da das LZE den Rest der Bildschirmzeile an der DSS löscht, können für diesen Zeilenrest keine anderen Zeichen dargestellt werden, als das von der DSS hardwaremäßig dafür vorgesehene Zeichen (normalerweise NIL). Die EDT-Anweisung @SYMBOLS FILLER kann also im Unicode-Modus nicht mehr dazu verwendet werden, das im F-Modus zwischen Satzende und Bildschirmzeilenende dargestellte Füllzeichen festzulegen. Aus Kompatibilitätsgründen wird das so spezifizierte Füllzeichen (Standardmäßig NIL) bei der Eingabe innerhalb eines Datensatzes aber nach wie vor durch Leerzeichen ersetzt. Sollen alle eingegebenen Zeichen innerhalb eines Datensatzes, also auch NIL-Zeichen, unverändert in die Arbeitsdatei übernommen werden, muss das Füllzeichen mit @SYMBOLS FILLER='␣' auf das Leerzeichen geändert werden.

Behandlung von NIL-Zeichen im Datenfenster

Enthält eine Bildschirmzeile nur NIL-Zeichen, also auch kein LZE, und umfasst der dargestellte Ausschnitt den gesamten Datensatz, wird die Bildschirmzeile nicht in die Arbeitsdatei übernommen bzw. aus der Arbeitsdatei gelöscht.

Bei der Erfassung eines Satzes (durch Eintippen in eine leere Datei, Weiterschreiben am Dateiende, Einfügen in Bildschirmzeilen, die nach einer der Kurzanweisungen 1..9 oder I zum Einfügen angeboten werden), werden NIL-Zeichen vor oder zwischen sonstigen Zeichen (auch vor LZE) in Leerzeichen umgesetzt, sofern nicht mit @SYMBOLS FILLER ein Zeichen ungleich NIL als Füllzeichen spezifiziert wurde (siehe oben).

NIL-Zeichen am Ende einer Bildschirmzeile 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. Dieser Mechanismus wird insbesondere beim Einfügen mit der Kurzanweisung E ausgenutzt.

Zum Löschen eines ganzen Datensatzes sollte grundsätzlich die Kurzanweisung D verwendet werden. Die Tasten LZE und LZF können zum Löschen von Teilen eines Datensatzes verwendet werden und wirken wie folgt:

– LZE löscht alle Zeichen des Datensatzes ab der eingegebenen Position (auch die Zeichen rechts außerhalb des am Bildschirm dargestellten Ausschnitts). Bei Eingabe in Spalte 1 der Bildschirmzeile wird ein Satz der Länge 0 in die Arbeitsdatei eingefügt. LZE kann also niemals zum Löschen eines ganzen Satzes führen.

– LZF löscht nur den Rest der Bildschirmzeile, etwaige Zeichen des Datensatzes außerhalb der Bildschirmzeile werden im nächsten Dialogschritt von rechts nachgezogen. Wenn der dargestellte Ausschnitt den gesamten Datensatz umfasst und nach dem Löschen nur noch NIL-Zeichen enthält, also auch kein LZE, wird der Datensatz ganz gelöscht, d.h. aus der Arbeitsdatei entfernt. LZF wirkt in diesem Fall wie die Kurzanweisung D.

Nicht darstellbare Zeichen im Text

Enthält eine Datei am Bildschirm nicht darstellbare Zeichen, werden diese Zeichen als das gerätespezifische Schmierzeichen ausgegeben, das mit /MODIFY-TERMINAL-OPTIONS als SUBSTITUTE-CHARACTER eingestellt ist.

Wird ein solcher Satz geändert, wird anstelle des Schmierzeichens wieder 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 Bildschirmzeile geschützt dargestellt mit einem '?' in der Kurzanweisungsspalte. Der ursprüngliche Inhalt des Satzes bleibt erhalten.

Warnung

Wird durch Einfügen oder Ausfügen die Position des Schmierzeichens an eine Stelle verschoben, an der vorher ebenfalls ein Schmierzeichen war, kann der EDT nicht feststellen, ob es sich um das gleiche Schmierzeichen oder um ein verschobenes handelt. In diesem Fall findet keine Kennzeichnung mit Fragezeichen '?' statt und der Inhalt des Satzes wird evtl. in nicht beabsichtigter Weise geändert.

Hinweis

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 darstellbare Zeichen enthalten, sollten im HEX-Modus (siehe @PAR HEX) oder über die Ersatzdarstellung (siehe unten) erfasst werden.

5.1.1.4 Kurzanweisungen im F-Modus

Kurzanweisungen sind Anweisungen von der Länge eines Zeichens. Sie werden in der Kurzanweisungsspalte eingegeben. Bei Kurzanweisungen wird zwischen Groß- und Kleinschreibung nicht unterschieden.

Die folgende Übersicht stellt die Kurzanweisungen thematisch gegliedert zusammen.

Kurzanweisungen zum Positionieren des Arbeitsfensters

Kurzanweisung	Funktion
+ / -	Positionieren des Arbeitsfensters (vertikal)
+ / - F1	Positionieren des Arbeitsfensters nach der Strukturtiefe
S	Interaktives Positionieren des Arbeitsfensters (horizontal und vertikal)

Kurzanweisungen zum Kopieren und Verschieben von Datensätzen

Kurzanweisung	Funktion
*	Löschen des Kopierpuffers
C	Aufsammeln von Zeilen zum Kopieren
R	Aufsammeln von Zeilen zum mehrfachen Kopieren
M	Aufsammeln von Zeilen zum Verschieben
A	Kopieren / Verschieben hinter eine Zeile (after)
B	Kopieren / Verschieben vor eine Zeile (before)
O	Kopieren / Verschieben über einen Zeilenbereich (on)

Kurzanweisungen zum Bearbeiten von Sätzen

Kurzanweisung	Funktion
D	Löschen von Sätzen
J	Zusammenketten zweier Sätze
L	Umsetzen von Sätzen in Kleinbuchstaben
U	Umsetzen von Sätzen in Großbuchstaben
X	Ändern von Sätzen
H	Anzeigen / Ändern von Sätzen im Hexadezimalmodus
E	Einfügen von Zeichen
1..9	Einfügen von Datenzeilen
I	Aktivieren der Dauereinfügefunktion

Kurzanweisungen zum Arbeiten mit Satzmarkierungen

Kurzanweisung	Funktion
D F3	Löschen einer Satzmarkierung
1..9 F3	Setzen einer Satzmarkierung

Sonstige Kurzanweisungen

Kurzanweisung	Funktion
K	Kopieren einer Zeile in die Anweisungszeile
T	Syntaxtest durch SDF

Die genaue Beschreibung der einzelnen Kurzanweisungen finden Sie im Kapitel „Kurzanweisungen des F-Modus (alphabetisch)“ auf Seite 591.

Syntax- und Semantikprüfung

Die Syntax- und Semantikprüfung für die Kurzanweisungen wird als erster Schritt bei der Bearbeitung der Eingaben in einem Arbeitsfenster durchgeführt (siehe Abschnitt Abarbeitungsreihenfolge). Werden ungültige Kurzanweisungen oder ungültige Kombinationen (z.B. `M` gefolgt von `C`, siehe unten) erkannt, werden die weiteren Schritte der Eingabebearbeitung nicht mehr ausgeführt. Anstelle der fehlerhaften Kurzanweisungen wird dann ein Fragezeichen `'?'` ausgegeben und die Schreibmarke wird auf die erste fehlerhafte Kurzanweisung positioniert.

Kombinierbarkeit der Kurzanweisungen in einem Arbeitsfenster

In Abhängigkeit von der zur Datenübertragung verwendeten Funktionstaste bzw. der eingegebenen Kurzanweisungen werden bei der Abarbeitung der Kurzanweisungsspalte folgende Fälle unterschieden:

1. Wird F3 verwendet, dann akzeptiert der EDT lediglich Kurzanweisungen, die mit F3 gesendet werden dürfen (Anweisungen zum Setzen und Löschen von Satzmarkierungen). Diese sind beliebig miteinander kombinierbar. Werden nicht erlaubte Kurzanweisungen mit F3 gesendet, betrachtet der EDT diese als ungültig, kennzeichnet sie mit `'?'` und bricht die weitere Bearbeitung der Eingabe ab.
2. Wird F1 verwendet, dann akzeptiert der EDT lediglich Kurzanweisungen, die mit F1 gesendet werden dürfen (+ bzw. − zum Positionieren nach Strukturtiefe). Von diesen ist nur eine pro Arbeitsfenster erlaubt. Werden nicht erlaubte Kurzanweisungen mit F1 gesendet, betrachtet der EDT diese als ungültig, kennzeichnet sie mit `'?'` und bricht die weitere Bearbeitung der Eingabe ab.
3. Werden die Kurzanweisungen mit DUE oder einer Funktionstaste ungleich F1 oder F2 abgesendet, ergibt sich aus folgender Tabelle, welche Kurzanweisungen miteinander in einem Arbeitsfenster kombinierbar sind. Dabei ist die Tabelle so zu lesen, dass die durch die Tabellenzeile bestimmte Kurzanweisung in der Kurzanweisungsspalte des gleichen Arbeitsfensters oberhalb der durch die Tabellenspalte bestimmten Kurzanweisung eingegeben wird. Kurzanweisungen sind (in dieser Reihenfolge) kombinierbar, wenn am Kreuzungspunkt in der Tabelle nichts eingetragen ist, sie sind nicht kombinierbar, wenn ein `X` eingetragen ist. Sonderfälle sind mit einem Kleinbuchstaben gekennzeichnet und werden unten erläutert.

	+	*	-	A	B	C	D	E	H	I	J	K	L	M	O	R	S	T	U	X	1..9
+	X		X														X	a			
*****		X		b	b										b						
-	X		X					X	X	X							X	a		X	X
A		b																			
B		b																			
C														X		X					
D																					
E	X		X														X	X			
H	X		X														X	X			
I	X		X							X							X	X			
J																					
K												X									
L																					
M						X										X					
O		b																			
R						X								X							
S	X		X					X	X	X							X	X		X	X
T	X		X					X	X	X							X			X	X
U																					
X	X		X														X	X			
1..9	X		X														X	X			

a) Bei einem Syntax-Error in der mit `T` getesteten SDF-Anweisung wird + bzw. − ignoriert.

b) Wenn weder `C` noch `M` noch `O` gleichzeitig mit * angegeben wurde, kommt die Meldung `EDT5360`, da der Kopierpuffer geleert wurde und nicht mehr kopiert werden kann.

Reihenfolge der Bearbeitung der Kurzanweisungen

Werden mehrere miteinander kombinierbare Kurzanweisungen in der Kurzanweisungsspalte eines Arbeitsfensters eingegeben, so werden sie in der folgenden Reihenfolge abgearbeitet:

– alle D−Kurzanweisungen
– die *-Kurzanweisung zum Löschen des Kopierpuffers
– die K-Kurzanweisung
– alle C-, M- und R-Kurzanweisungen zum Eintragen in den Kopierpuffer
– alle U- und L-Kurzanweisungen
– alle J-Kurzanweisungen zum Zusammenketten zweier Sätze
– alle A-, B-, O-Kurzanweisungen zum Kopieren oder Verschieben
– alle T-Kurzanweisungen zum Testen von SDF-Syntax
– die Kurzanweisungen + und − zum Positionieren
– die S-Kurzanweisung
– alle Kurzanweisungen X (Ändern), H (Ändern hexadezimal), E (Einfügen von Zeichen), 1..9 und I (Einfügen von Zeilen)

Die Kurzanweisungen X, H, E und I sowie 1..9 werden innerhalb eines Arbeitsfensters von oben nach unten abgearbeitet. Die Kurzanweisungen X, H und E hinter I bzw. 1..9 können verloren gehen, wenn aufgrund des Einfügebereichs die Zeilen nicht mehr am Bildschirm darstellbar sind. Eine Warnung erfolgt nicht.

Die Anweisungszeile wird nach der Abarbeitung der Kurzanweisungen ausgewertet (siehe Abschnitt „Abarbeitungsreihenfolge“ auf Seite 120).

5.1.1.5 Anweisung im Datenfenster - Auftrennen eines Datensatzes

Mit @PAR SEPARATOR kann ein beliebiges Satztrennzeichen definiert werden. Wird dieses Satztrennzeichen im Datenfenster in einer Bildschirmzeile 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 nach dem Verfahren *Einfügen zwischen zwei Zeilen* (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).
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 z.B. durch Einfügen oder Überschreiben des Auftrennzeichens. Insbesondere führt das Kopieren oder Verschieben eines Satzes nicht zum Auftrennen, auch wenn der Satz Auftrennzeichen enthält.

5.1.1.6 Anweisungszeile

Eingaben in der Anweisungszeile werden als Anweisungen interpretiert. Eine Übersicht über die Anweisungen des F-Modus wird im Abschnitt „Anweisungen im F-Modus“ auf Seite 130 gegeben. Das EDT-Anweisungssymbol (standardmäßig @) muss, mit Ausnahme der Anweisung @:, 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, einschließlich der fehlerhaften Anweisung, ausgegeben.

Nach korrekter Abarbeitung einer Eingabe wird die Anweisungszeile bei der Bildschirmausgabe gelöscht. Mit der Anweisung # bzw. n# kann die zuletzt eingegebene Anweisung bzw. die nt-letzte eingegebene Anweisung 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. Alternativ lässt sich auch mit der Anweisung @SHIH der Puffer der letzten vom EDT ausgeführten Anweisungen in die Arbeitsdatei 9 ausgeben (siehe unten).

Der Inhalt einer Anweisungszeile bzw. ein nicht benötigter Zeilenrest kann mit LZF gelöscht werden.

Innerhalb von Literalen wird ein Semikolon nicht als Anweisungstrennzeichen interpretiert.

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

Die maximal mögliche Anweisungslänge ist im F-Modus aufgrund der Begrenzung durch die Datensichtstation kleiner als im L-Modus, siehe dazu auch den folgenden Abschnitt.

Anweisungszeile-Fortsetzung

Ist beim Abschicken des Bildschirms das letzte Zeichen der Anweisungszeile kein NIL-Zeichen, wird angenommen, dass der Benutzer für die Eingabe einen Fortsetzungsbereich benötigt. In diesem Fall wird ihm eine zweite Zeile angeboten, sofern das Arbeitsfenster groß genug ist, um anschließend noch mindestens eine Datenzeile ausgeben zu können. Der Inhalt der Anweisungszeile wird vom EDT in die Bildschirmzeile davor gebracht und die nun leere Anweisungszeile wird als Fortsetzungszeile angeboten. Maximal werden zwei Fortsetzungszeilen angeboten, d.h. die maximale Eingabelänge beträgt 189 Zeichen für eine DSS mit 80 Spalten bzw. 345 für eine DSS mit 132 Spalten.

Behandlung von NIL-Zeichen in der Anweisungszeile

NIL-Zeichen am Ende der Anweisungszeile werden ignoriert. Vor der Analyse der Eingabe werden NIL-Zeichen innerhalb der Anweisungsfolge in Leerzeichen umgesetzt.

5.1.1.7 Anweisungspuffer

Der EDT speichert die letzten im F-Modus eingegebenen Anweisungen in einem Puffer. Dieser Anweisungspuffer kann mit Anweisung @SHIH (*Show Input History*) ausgegeben werden. Mit der Kurzanweisung `K` kann man (nach Ausgabe in eine Arbeitsdatei) die Ausgabezeile, die die gewünschte Anweisung enthält, in die Anweisungszeile bringen.

Alternativ kann auch mit der Anweisung # bzw. `n`# die letzte bzw. die `n`t-letzte Anweisung direkt in die Anweisungszeile zurückgeholt werden.

Der Anweisungspuffer kann maximal 2048 Anweisungen (unabhängig von ihrer Länge) aufnehmen. Alle Blätteranweisungen, die Anweisungen zum Wechseln der Arbeitsdatei und des Betriebs- bzw. Arbeitsmodus, sowie die Anweisungen @SHIH und # selbst werden nicht im Anweisungspuffer abgelegt. Ebenso werden Anweisungen, die nicht ausgeführt werden (z.B. wegen Syntaxfehlern), nicht in den Anweisungspuffer aufgenommen. Dagegen werden Anweisungen die zur Ausführung kommen, immer in den Anweisungspuffer aufgenommen, auch wenn sie einen Fehler melden.

Anweisungen, die in einer Anweisungskette eingegeben werden, werden einzeln in den Anweisungspuffer aufgenommen. Die Aufnahme erfolgt immer in der ursprünglich eingegebenen Form, also unter Umständen mit Kleinbuchstaben, unabhängig von der Einstellung mit @LOWER. Führende Leerzeichen werden dabei entfernt und leere Eingaben werden ignoriert.

5.1.1.8 Zustandsanzeige

Die Zustandsanzeige zeigt, von links nach rechts gelesen:

– die Zeilennummer der ersten Zeile des Arbeitsfensters (6stellig) bzw. `0000.00` wenn die Arbeitsdatei leer ist
– die Spaltennummer, ab der die Sätze im Datenfenster dargestellt werden (5stellig)
– die Nummer der dargestellten Arbeitsdatei (2stellig in runden Klammern)

Die Zustandsanzeige ist nicht überschreibbar. Wenn die Zustandsanzeige das hier beschriebene Format hat (Spaltennummer 5stellig, Arbeitsdateinummer 2stellig), ist das gleichzeitig ein Kennzeichen, dass der EDT sich im Unicode-Modus befindet (im Kompatibilitäts-Modus ist die Spaltennummer 3stellig und die Arbeitsdateinummer 1stellig).

Beispiel

```
.....·0008.00:00001(21)
```

Zeile `8.00` ist die erste Zeile des Arbeitsfensters `21`.

5.1.1.9 Abarbeitungsreihenfolge

In einem Arbeitsfenster werden die Eingaben in folgender Reihenfolge abgearbeitet:

1. Syntax- und Semantikprüfung der Eingaben in der Kurzanweisungsspalte
2. Datenfensterauswertung
3. Ausführen der Kurzanweisungen in der Kurzanweisungsspalte
4. Ausführen der Anweisungen in der Anweisungszeile

Wenn zwei Arbeitsfenster vorhanden sind (siehe auch den Abschnitt „Zweites Arbeitsfenster“ auf Seite 124), wird jeder der oben angegebenen Schritte zuerst für das obere, dann für das untere Arbeitsfenster ausgeführt.

Unabhängig von der Anzahl der Arbeitsfenster werden, solange in der Kurzanweisungsspalte eines der Arbeitsfenster Kurzanweisungen zum Einfügen bzw. Ändern vorhanden sind (`1..9`,`I`,`X`,`H` oder `E`), nur Datenfenster und Kurzanweisungsspalte ausgewertet. Dabei werden zuerst die Sätze aus dem Datenfenster in die Datei übernommen. Anschließend wird die Kurzanweisungsspalte ausgewertet. Der Inhalt der Anweisungszeile bleibt unverändert und wird erst ausgewertet, wenn in keinem der Arbeitsfenster eine der oben genannten Kurzanweisungen mehr angegeben wird. Eine aktive Dauereinfügefunktion (siehe Kurzanweisung `I`) verhindert die Auswertung der Anweisungszeile demgegenüber nicht.

Beim Auftreten von Fehlern während der Abarbeitung gilt folgendes:

1. Bei Fehlern während der Syntax- oder Semantikprüfung in der Kurzanweisungsspalte werden die fehlerhaften Kurzanweisungen mit Fragezeichen (`'?'`) überschrieben und die weitere Bearbeitung der Eingaben wird abgebrochen.
2. Bei Fehlern während der Datenfensterauswertung, etwa wenn das Auftrennen einer Zeile bei @PAR RENUMBER=OFF nicht möglich ist, werden in dem betroffenen Arbeitsfenster die Kurzanweisungen abgearbeitet, die Anweisungszeile wird nicht abgearbeitet, sondern wird unverändert wieder angezeigt. Eingaben in einem evtl. vorhandenen zweiten Arbeitsfenster werden normal verarbeitet.
3. Bei Fehlern während der Bearbeitung von Kurzanweisungen, etwa wenn das Einfügen von Zeilen bei @PAR RENUMBER=OFF nicht möglich ist, werden in dem betroffenen Arbeitsfenster die restlichen Kurzanweisungen abgearbeitet, die Anweisungszeile wird nicht abgearbeitet, sondern wird unverändert wieder angezeigt. Kurzanweisungen in einem evtl. vorhandenen zweiten Arbeitsfenster werden ebenfalls verarbeitet. Die Anweisungszeile wird auch da nicht ausgeführt.
4. Fehler während der Bearbeitung von Eingaben in der Anweisungszeile werden erst entdeckt, wenn Anweisungen im Datenfenster bzw. Kurzanweisungen schon abgearbeitet sind. Anweisungen in einer Anweisungsfolge werden bis zu der fehlerhaften Anweisung ausgeführt. Dies gilt für jedes Arbeitsfenster unabhängig vom anderen (siehe auch Abschnitt „Anweisungszeile“ auf Seite 118).

Hinweis

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

Zeigen beide Arbeitsfenster den gleichen oder überlappende Ausschnitte aus der gleichen Arbeitsdatei können sich Kurzanweisungen und Anweisungen in den beiden Arbeitsfenstern gegenseitig beeinflussen, etwa wenn eine Zeile gleichzeitig in einem Arbeitsfenster gelöscht und im anderen in den Kopierpuffer übertragen werden soll. Von einer derartigen Arbeitsweise wird daher abgeraten. Nicht überlappende Ausschnitte aus der gleichen Arbeitsdatei können dagegen problemlos in zwei Arbeitsfenstern gleichzeitig bearbeitet werden.

5.1.2 Verändern des Arbeitsfensters

Der Benutzer kann das Format des Arbeitsfensters verändern, indem er

- die Zeilennummernanzeige ein- oder ausschaltet,
- lange Sätze als Ausschnitt oder vollständig im Datenbereich darstellen lässt,
- einen Spaltenzähler (Zeilenlineal) ausblendet oder einblendet,
- ein oder zwei Arbeitsfenster auf dem Bildschirm darstellen lässt oder
- die Anzeige der Daten in hexadezimaler Darstellung ein- oder ausschaltet.

Nach dem Starten des EDT ist das Format des Arbeitsfensters wie folgt voreingestellt:

- Zeilennummernanzeige eingeschaltet,
- keine vollständige Anzeige von langen Sätzen,
- kein Zeilenlineal,
- ungeteiltes (nur ein) Arbeitsfenster und
- keine hexadezimale Darstellung.

5.1.2.1 Zeilennummernanzeige

Mit der Anweisung @PAR INDEX bzw. mit der nur im F-Modus verfügbaren Anweisung @INDEX kann die Zeilennummernanzeige im Arbeitsfensters ein- oder ausgeschaltet werden. Standardmäßig (@PAR INDEX=ON bzw. @INDEX ON) wird das Format mit 72 Zeichen je Bildschirmzeile (bzw. 124 Zeichen je Bildschirmzeile, wenn @VDT F2 gegeben wurde), 6-stelliger Zeilennummernanzeige und trennendem Leerzeichen eingestellt.

Mit @PAR INDEX=OFF bzw. @INDEX OFF werden 80 (bzw. 132) Zeichen pro Bildschirmzeile ohne Zeilennummernanzeige eingestellt. In beiden Formaten bildet die erste Spalte jeder Bildschirmzeile die Kurzanweisungsspalte. Bei Darstellung mit Zeilennummern überlappt sich diese mit der ersten Spalte der Zeilennummernanzeige. Bei Darstellung ohne Zeilennummern überlappt sich diese mit der ersten Spalte des Datenfensters.

5.1.2.2 Ausgabe von langen Sätzen

Mit der Anweisung @PAR EDIT-LONG bzw. mit der nur im F-Modus verfügbaren Anweisung @EDIT LONG kann die Ausgabe am Bildschirm verändert werden. Für Datensätze, die größer als die Spaltenzahl des Bildschirms sind, kann bestimmt werden, dass

– die Sätze vollständig im Datenfenster dargestellt werden - soweit ihre Länge das zulässt (@PAR EDIT-LONG=ON bzw. @EDIT LONG ON)
– nur ein Ausschnitt von 72, 80, 124 bzw. 132 Zeichen eines Satzes (je nach Einstellung bei @PAR INDEX) im Datenfenster dargestellt wird (@PAR EDIT-LONG=OFF bzw. @EDIT LONG OFF).

Der EDIT-LONG-Modus arbeitet ohne Zeilennummernanzeige. Ein Satz wird fortlaufend über mehrere Bildschirmzeilen geschrieben.

Bei der Eingabe neuer Sätze am Ende der Arbeitsdatei oder in einem mit den Kurzanweisungen I bzw. 1..9 bereitgestellten Einfügebereich wird eine Zeile, die weder mit LZE noch mit NIL-Zeichen abgeschlossen ist, mit ihrer Nachfolgezeile zu einem Satz verbunden. Auf diese Weise lassen sich im EDIT-LONG-Modus je nach Bildschirmformat Sätze mit einer Länge bis zu 3432 Zeichen direkt eingeben. Dies gilt nur für *neue* Sätze. Wenn ein *existierender* Satz am Ende der Arbeitsdatei oder unmittelbar vor einem Einfügebereich in den neuen Zeilenbereich hinein fortgeschrieben wird, werden die neuen Sätze nicht mit dem existierenden Satz verbunden. Existierende Sätze können nur mit der Kurzanweisung E erweitert werden (siehe unten).

Wird eine Eingabezeile dadurch verkürzt, dass eine Ersatzdarstellung (z.B. %U20AC) in das entsprechende Zeichen umgewandelt wird, wird sie nicht mit der Nachfolgezeile verbunden, selbst wenn die ursprüngliche Eingabezeile keine abschließenden LZE oder NIL-Zeichen enthielt.

Der Hardware-Tabulator (siehe @TABS-Anweisung) wirkt im EDIT-LONG-Modus nur in der ersten Bildschirmzeile eines Satzes. Bei existierenden Sätzen kann innerhalb der nachfolgenden Bildschirmzeilen mit dem Hardware-Tabulator nicht positioniert werden. Bei neuen Sätzen (siehe voriger Abschnitt) sind die Tabulatorpositionen in allen Zeilen mit denen der ersten Bildschirmzeile identisch.

Wird mit F2 das gesamte Datenfenster auf überschreibbar gestellt, bleibt ein nicht vollständig gezeigter letzter Satz im Datenfenster nicht überschreibbar. Wird ein solcher Satz mit der Kurzanweisung X markiert, so wird das Fenster (wenn möglich) so positioniert, dass der Satz vollständig im Datenfenster dargestellt wird. Ist dies nicht möglich, bleibt der Satz nicht überschreibbar. Sätze, die so lang sind, dass sie generell nicht vollständig darstellbar sind, können im EDIT-LONG-Modus nicht editiert werden. Solche Sätze lassen sich nur nach @PAR EDIT-LONG=OFF editieren, der zu editierende Ausschnitt muss dann ggf. durch horizontales Blättern (>, <) in den dargestellten Bereich verschoben werden.

Die Kurzanweisungsspalte ist die erste Spalte des Bildschirmes, bei mehrzeiligen Sätzen ist die Kurzanweisung in der ersten Zeile einzugeben. Soll ein Satz mit der Kurzanweisung E erweitert werden, wird im EDIT-LONG-Modus eine ganze NIL-Zeile zusätzlich ausgegeben. Falls notwendig, wird das Fenster dazu neu positioniert. Die Kurzanweisungen S und H sind im EDIT-LONG-Modus nicht zugelassen.

Im Gegensatz zur Darstellung bei @PAR EDIT-FULL=ON ist im EDIT-LONG-Modus immer nur entweder die Kurzanweisungsspalte oder der zugehörige Satz im Datenfenster überschreibbar.

Im EDIT-LONG-Modus wird weder der mit @PAR SCALE=ON eingestellte Spaltenzähler noch eine mit @PAR INFORMATION=ON angeforderte Informationszeile dargestellt. Spaltenzähler und Informationszeile werden erst wieder eingeblendet, wenn der EDIT-LONG-Modus verlassen wird.

Wird der EDIT-LONG-Modus mit @PAR EDIT-LONG=OFF bzw. @EDIT LONG OFF verlassen, bleibt die Zeilennummernanzeige ausgeschaltet.

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

Hinweis

Die Anweisungen zum horizontalen Blättern (>, <) sowie die Anweisung @PAR EDIT-FULL=ON werden akzeptiert und bearbeitet, die Auswirkungen werden jedoch erst nach dem Ausschalten des EDIT-LONG-Modus bzw. nach dem Wiedereinstellen der Zeilennummernanzeige sichtbar.

5.1.2.3 Spaltenzähler

Mit der Anweisung @PAR SCALE=ON bzw. mit der nur im F-Modus verfügbaren Anweisung @SCALE ON wird ein Spaltenzähler (Zeilenlineal) im Arbeitsfenster ausgegeben. Der Spaltenzähler erscheint als 1. Bildschirmzeile im Arbeitsfenster (nicht im EDIT-LONG-Modus). Die Anweisungen @PAR SCALE=OFF bzw. @SCALE OFF schalten die Anzeige des Spaltenzählers wieder aus.

Falls ein Tabulator definiert ist (siehe @TABS), wird der Spaltenzähler um eine Bildschirmzeile erweitert, in der die aktuellen Positionen des Tabulators angezeigt werden.

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

Im EDIT-LONG-Modus wird die Anweisung @PAR SCALE akzeptiert und bearbeitet. Die Auswirkungen (Einblenden oder Ausblenden des Spaltenzählers) werden jedoch erst nach dem Ausschalten des EDIT-LONG-Modus sichtbar.

5.1.2.4 Zweites Arbeitsfenster

Mit der Anweisung @PAR SPLIT bzw. mit der nur im F-Modus verfügbaren Anweisung @SPLIT wird die Darstellung eines zweiten Arbeitsfensters am Bildschirm ein- bzw. ausgeschaltet. Für die Darstellung mit zwei Arbeitsfenstern gilt folgendes:

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 @PAR SPLIT=OFF und in der unteren Anweisungszeile eine Anweisung eingegeben, wird @PAR SPLIT=OFF mit einer Fehlermeldung abgewiesen.

Darstellung bei Arbeitsfenstern mit unterschiedlichen Zeichensätzen

Durch das Teilen des Bildschirms können zwei Arbeitsfenster dargestellt werden, die Daten in unterschiedlichen Zeichensätzen enthalten. Die DSS ist demgegenüber nur in der Lage, jeweils einen Zeichensatz korrekt darzustellen. Zudem führt das Umschalten des Zeichensatzes an der DSS stets zu einem Löschen des Bildschirms mit anschließendem Neuaufbau (sollte also so selten wie möglich stattfinden).

Der EDT versucht daher, möglichst ohne Umschalten des Zeichensatzes für die DSS auszukommen. Wenn die DSS Unicode-Zeichensätze unterstützt und die automatische Wahl des Zeichensatzes eingestellt ist, wird zur Kommunikation mit der DSS `UTF8` benutzt und auch beim Teilen des Fensters nicht gewechselt (Details siehe in der Einleitung des Abschnitts „Das Arbeitsfenster“ auf Seite 107). In diesem Unicode-Zeichensatz ist die Darstellung in beiden Datenfenstern korrekt oder zumindest lesbar.

Wenn für die Kommunikation mit der DSS ein 7-Bit- oder 8-Bit-Zeichensatz verwendet wird, etwa weil die DSS Unicode nicht unterstützt, bestimmt standardmäßig das obere Arbeitsfenster den zur Kommunikation mit der DSS benutzten Zeichensatz, wenn der Anwender nicht mittels @CODENAME-Anweisung diese Einstellung explizit geändert hat (siehe unten).

In einem Datenfenster, deren zugeordnete Arbeitsdatei nicht in dem Zeichensatz der DSS vorliegt, werden die Zeichen durch ihr Äquivalent im Zeichensatz der DSS bzw. durch Schmierzeichen (wenn das Zeichen in diesem Zeichensatz ungültig ist) dargestellt.

Der zur Kommunikation mit der DSS verwendete Zeichensatz kann mit der Anweisung @CODENAME (Format 2) geändert werden.
Mit der Anweisung wird nur die Darstellung geändert, nicht die Codierung der Daten der Arbeitsdatei (siehe Anweisung @CODENAME).

Für die Interpretation der Eingaben gilt das in der Einleitung des Abschnitts „Das Arbeitsfenster“ auf Seite 107 sowie im Abschnitt „Zeichensätze“ auf Seite 48 beschriebene Verhalten.

5.1.2.5 Hexadezimalmodus

Der EDT kann im F-Modus den Inhalt der Arbeitsdateien in hexadezimaler Form darstellen und editierbar machen. Diese Art der Darstellung wird als Hexadezimalmodus oder kurz HEX-Modus bezeichnet. Dazu dient die Anweisung @PAR HEX=ON bzw. die nur im F-Modus verfügbare Anweisung @HEX ON. Mit @PAR HEX=OFF bzw. @HEX OFF wird der HEX-Modus wieder abgeschaltet. Mit der Kurzanweisung H kann auch eine einzelne Zeile in hexadezimaler Form dargestellt und editierbar gemacht werden.

Die Darstellung im HEX-Modus ist abhängig vom Zeichensatz, mit dem die Daten in der Arbeitsdatei codiert sind. Für 7-Bit- und 8-Bit-Zeichensätze wird jedes Zeichen durch zwei hexadezimale Ziffern codiert, UTF16 benötigt für jedes Zeichen vier hexadezimale Ziffern, bei UTF8 variiert die Anzahl der benötigten hexadezimalen Ziffern je nach Zeichen zwischen zwei, vier und sechs, bei UTFE sogar zwischen zwei und acht. Zusätzlich wird immer noch die abdruckbare Form der Zeile dargestellt.

Die erste Bildschirmzeile zeigt den Satzinhalt in abdruckbarer Form. Unter jedem abdruckbaren Zeichen dieser Bildschirmzeile steht vertikal der hexadezimale Code des Zeichens auf mehrere Bildschirmzeilen aufgeteilt. Diese Bildschirmzeilen werden im Folgenden als Hexzeilen bezeichnet. Bei der Darstellung von 7- und 8-Bit-Zeichensätzen sind es zwei Hexzeilen, bei UTF16 vier, bei UTF8 sechs und bei UTFE acht. Es folgt ein Spaltenzähler wie bei @PAR SCALE=ON. Ein evtl. mit @PAR SCALE=ON pro Datenfenster eingeschalteter Spaltenzähler wird im HEX-Modus nicht ausgegeben.

In den Hexzeilen werden nur Hexadezimalziffern (0..9,A..F) und NIL-Zeichen dargestellt und auch nur solche oder das Leerzeichen als Eingabe zugelassen. Nullbytes am Satzende werden nicht entfernt und noch leere Sätze hinter dem Dateiende, die noch im Datenfenster angezeigt werden, werden auch in den Hexzeilen durch NIL-Zeichen dargestellt.

Der HEX-Modus gilt für die aktuelle Arbeitsdatei, unabhängig vom Arbeitsfenster, d.h. wenn die gleiche Arbeitsdatei in unterschiedlichen Arbeitsfenstern dargestellt wird, werden beide Arbeitsfenster in der gleichen Darstellung gezeigt.

Der HEX-Modus wird auch durch @PAR EDIT-LONG=ON ausgeschaltet.

Ändern von Sätzen im HEX-Modus

Änderungen können sowohl in der ersten Bildschirmzeile (abdruckbare Form) als auch in den Hexzeilen vorgenommen werden. Wurden in einem Dialogschritt im gleichen Satz sowohl in der Zeichendarstellung als auch in den Hexzeilen Zeichen eingegeben, werden die Änderungen in der Zeichendarstellung ignoriert.

In den Hexzeilen sind als Eingabe bei 7- oder 8-Bit Zeichensätzen nur Hexadezimal-Zeichen (`0..9, A..F`) und bei Unicode-Zeichensätzen müssen diese eine legale Codierung ergeben.
Werden nicht zugelassene Zeichen oder Codierungen eingegeben, wird ein *Korrekturdialog* eröffnet, d.h. der Bildschirm wird erneut ausgegeben, wobei die fehlerhaften Zeichen bzw. Codierungen mit einem Fragezeichen überschrieben sind. Die Schreibmarke steht in der ersten fehlerhaften Bildschirmzeile. Die restlichen, fehlerfreien Zeichen oder Codierungen werden am Bildschirm geändert dargestellt aber noch nicht in die Arbeitsdatei übernommen. Will man das fehlerhafte Zeichen nicht korrigieren, kann der *Korrekturdialog* verlassen werden, indem der Bildschirm unverändert mit DUE abgeschickt wird. Es wird dann der alte Inhalt der fehlerhaften Zeile wiederhergestellt, fehlerfreie Zeilen werden in die Arbeitsdatei übernommen. Demgegenüber wird mit K3 der *Korrekturdialog* nicht verlassen sondern nur die letzte Änderung durch Tastatureingabe annuliert. Hexziffern dürfen auch gelöscht (mit `NIL`-Zeichen überschrieben) oder ausgefügt werden. Beim Ausfügen ist darauf zu achten, dass sich die Halbbytes einer Zeichencodierung nicht gegeneinander verschieben.

Darstellung bei geteiltem Bildschirm

Bei geteiltem Bildschirm (@PAR SPLIT) ist die Darstellung im `HEX`-Modus von der Anzahl der Datenzeilen (Bildschirmzeilen im Datenfenster) des jeweiligen Bildschirmes abhängig:

– Fehlt nur eine Bildschirmzeile, wird auf die Spaltenzählerzeile verzichtet.

– Reicht der Platz nicht um alle Hexzeilen darzustellen, wird nur noch die Zeichendarstellungszeile angezeigt. Auf die Spaltenzählerzeile wird, im Gegensatz zur Darstellungsform des Kompatibilitäts-Modus verzichtet, da sie an dieser Stelle nicht hilfreich erscheint.

– Noch verbleibende Bildschirmzeilen werden für Darstellung der nachfolgenden Sätze verwendet. Für diese gelten dann diese Regeln rekursiv.

Da im `HEX`-Modus ein Satz immer durch mehrere Bildschirmzeilen dargestellt wird, ist er nur dann sinnvoll, wenn mindestens ein Satz mit all seinen Hexzeilen dargestellt werden kann. Ist dies nicht der Fall, wird beim Einschalten des `HEX`-Modus die Meldung `EDT2404` ausgegeben, der `HEX`-Modus wird aber eingeschaltet. Der Anwender kann dann durch Vergrößern des Datenfensters die Hexzeilen sichtbar machen.

Besonderheiten bei der Darstellung von UTFE und UTF8

In den im BS2000 unterstützten Teilmengen von UTF8 und UTFE werden Zeichen mit ein bis drei bzw. ein bis vier Bytes pro Zeichen codiert. Dennoch wird jeder Satz mit sechs bzw. acht Hexzeilen dargestellt. Für Zeichen, die mit weniger als sechs bzw. acht Bytes codiert sind, stehen in den restlichen Hexzeilen nur NIL-Zeichen.

Beispiel

Die Daten im Beispiel sind in UTF8 codiert.

```
   1.00 Preisänderung:<·····················································
        57667C66677663·····················································
        025933E4525E7A·····················································
        ·····A·····························································
        ·····4·····························································
        ···································································
        ···································································
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   2.00 <·····························································
        ···································································
        ···································································
        ···································································
        ···································································
        ···································································
        ···································································
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   3.00       200,00 €<·····················································
        2222223332332E·····················································
        000000200C0002·····················································
        ·············8·····················································
        ·············2·····················································
        ·············A·····················································
        ·············C·····················································
·····························································0001.00:00001(00)
```

5.1.3 Funktionstasten im F-Modus

Im F-Modus des EDT lassen sich eine Reihe von Aktionen durch Funktionstasten auslösen. Die Taste K2 nimmt dabei eine Sonderstellung ein, weil sie einerseits als einzige Funktionstaste auch im L-Modus wirkt, ihre Wirkung andererseits aber komplett unterdrückt werden kann, etwa in nicht unterbrechbaren Prozeduren oder wenn der EDT als Unterprogramm (z.B. in der POSIX-Shell) aufgerufen wurde.

5.1.3.1 Die F-Tasten

Alle F-Tasten übertragen die Eingaben im Datenfenster, in der Kurzanweisungsspalte und in der Anweisungszeile von der DSS an den EDT. Die Tasten F1 bis F3 haben darüber hinaus spezielle Funktionen:

F1 Positionieren zu Sätzen mit gleicher Strukturtiefe

Mit F1 kann in Verbindung mit den Kurzanweisungen + und − zum nächsten Satz mit derselben Strukturtiefe positioniert werden (siehe Abschnitt „Kurzanweisungen im F-Modus“ auf Seite 113).

F2 Alle Bildschirmzeilen des Datenfensters überschreibbar stellen

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

Wenn bei Übertragung mit F2 aufgrund der vom EDT eingehaltenen Abarbeitungsreihenfolge die Eingaben in der Anweisungszeile noch nicht ausgeführt wurden, z.B. weil sowohl eine Anweisung als auch eine der Kurzanweisungen `1..9`, `I` oder `E` (siehe Abschnitt Abarbeitungsreihenfolge) mit F2 übertragen wurde, wird das Datenfenster zunächst überschreibbar gestellt und die Anweisungszeile wird unverändert wieder ausgegeben. Die dann im Datenfenster eingegebenen Änderungen werden noch vor dem Abarbeiten der Anweisungszeile wirksam.

F3 Bearbeiten von Satzmarkierungen

Mit F3 werden in Verbindung mit den Kurzanweisungen `1..9` und `D` bzw. mit den Blätteranweisungen +, − und ++, −− folgende Funktionen ausgelöst:

– Setzen von Satzmarkierungen (Kurzanweisungen `1..9`)
– Löschen von Satzmarkierungen (Kurzanweisung `D`)
– Positionieren zu Sätzen mit Satzmarkierungen (Anweisungen +,−, ++, −−)

5.1.3.2 Die K-Tasten

Im Gegensatz zu den F-Tasten erfolgt bei Betätigung einer K-Taste keine Übertragung der geänderten Daten vom Bildschirm an den EDT. Alle Eingaben am Bildschirm gehen also verloren.

K1 Beenden des EDT

Mit K1 wird die Beendigung des EDT verlangt. Im Unterschied zur Beendigung mit @HALT wird auch im Falle, dass die Arbeitsdateien keine ungesicherten Daten enthalten, eine Sicherheitsabfrage in der Meldungszeile des Arbeitsfensters ausgegeben:

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

Bei Eingabe von `Y` wird der EDT beendet. Bei `N` wird der EDT fortgesetzt. Wenn eine der Arbeitsdateien ungesicherte Daten enthält, verhält sich K1 wie @HALT (siehe auch Abschnitt „Beenden des EDT-Laufs“ auf Seite 96).

K2 Unterbrechung des EDT-Laufs

Unterbrechungen des EDT-Laufs mit Wechsel in den Systemmodus können außer über die Anweisung @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 ein anderes Programm geladen (z.B. mit einem der Kommandos `/START-PROGRAM` oder `/LOAD-PROGRAM`) oder wird eine Prozedur gestartet, die ein anderes Programm lädt, wird der EDT ohne Rückfrage entladen.

K3 Wiederherstellen des Bildschirminhaltes, Verwerfen von Benutzereingaben

Wurde der Bildschirminhalt verschoben (z.B. durch eine Broadcast-Meldung), kann mit K3 der ursprüngliche Zustand wiederhergestellt werden. Der Bildschirminhalt wird (einschließlich der evtl. vom EDT ausgegebenen Meldungen) genau so wiederhergestellt, wie er vor Eingabe des ersten Zeichens durch den Benutzer sichtbar war. Die Taste K3 kann daher auch benutzt werden, um alle Benutzereingaben zu verwerfen und die Eingaben neu aufzusetzen, etwa wenn man versehentlich Bildschirmzeilen schon mit neuem Text überschrieben hat, die eigentlich unverändert bleiben sollen.

K4 bis K15 werden wie K3 behandelt (siehe oben).

5.1.4 Anweisungen im F-Modus

Folgende Anweisungen sind im F-Modus erlaubt:

<	@GETJV	@SEARCH-OPTION
<<	@GETLIST	@SEPARATE
+	@GETVAR	@SEQUENCE
++	@HALT	@SET
$0..$22	@HEX	@SETF
-	@INDEX	@SETJV
--	@INPUT (Format 1+2)	@SETLIST
>	@LIMITS	@SETSW
@:	@LIST	@SETVAR
#	@LOAD	@SHIH
@AUTOSAVE	@LOG	@SHOW
@BLOCK	@LOWER	@SORT
@CHECK (Format 2)	@MODE	@SPLIT
@CLOSE	@MOVE	@STAJV
@CODENAME	@ON	@STATUS
@COLUMN	@OPEN	@SUFFIX
@COMPARE	@P-KEYS	@SYMBOLS
@CONVERT	@PAGE	@SYNTAX
@COPY	@PAR	@SYSTEM
@CREATE (Format 1+ 2)	@PREFIX	@TABS
@DELETE	@PRINT	@TMODE
@DELIMIT	@QUOTE	@UNLOAD
@DO (Format 1)	@RANGE	@UNSAVE
@DROP	@READ	@USE
@EDIT	@RENUMBER	@VDT
@ELIM	@RESET	@VTCSET
@END	@RETURN	@WRITE
@ERAJV	@RUN	@XCOPY
@EXEC	@SAVE	@XOPEN
@FSTAT	@SCALE	@XWRITE
@GET	@SDFTEST	0...22

Im F-Modus kann das EDT-Anweisungssymbol (standardmäßig `@`) auch weggelassen werden, mit Ausnahme der Anweisung `'@:'`. Die detaillierte Beschreibung der Anweisungen finden Sie in Kapitel 9.

5.2 L-Modus

Im L-Modus erfolgt die Dateibearbeitung zeilenorientiert, d.h. der EDT bietet im Dialogbetrieb nur eine Zeile (die *aktuelle* Zeile) zur Eingabe an bzw. liest (im Stapel- und im Dialogbetrieb) jeweils eine Zeile von SYSDTA. Diese Zeile kann sowohl Datensätze als auch Anweisungen enthalten und wird nach dem Einlesen sofort verarbeitet.

Datensätze werden in die jeweils aktuelle Zeile geschrieben, anschließend wird die aktuelle Zeile um die aktuelle Schrittweite erhöht. Die aktuelle Zeile kann symbolisch über das Symbol '*' angesprochen werden, z.B. @PRINT *.

Anweisungen werden sofort ausgeführt. Zur Unterscheidung dient das EDT-Anweisungssymbol @ (siehe weiter unten).

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

Bei der Abarbeitung von @DO-Prozeduren und @INPUT-Prozeduren sowie beim Lesen von SYSDTA mit RDATA (Systemprozeduren, Stapelbetrieb) werden die Anweisungen so behandelt, als ob sie im L-Modus eingegeben worden wären. Es sind in diesen Fällen also auch nur Anweisungen des L-Modus erlaubt.

Das Umschalten in den F-Modus erfolgt mit der Anweisung @EDIT FULL.

5.2.1 Eingabe im L-Modus

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

- das erste vom Leerzeichen verschiedene Zeichen das EDT-Anweisungssymbol (standardmäßig @) oder ein Benutzeranweisungssymbol für eine externe Anweisungsroutine ist (siehe @USE-Anweisung)
- und das zweite vom Leerzeichen verschiedene Zeichen nicht mit dem ersten vom Leerzeichen verschiedenen Zeichen identisch ist.

Erkennt der EDT eine Anweisung, wird diese sofort ausgeführt.

In den folgenden beiden Absätzen wird der Ausdruck *Anweisungssymbol* als Synonym sowohl für *EDT-Anweisungssymbol* als auch für *Benutzeranweisungssymbol* benutzt.

Alle Eingaben, deren erstes vom Leerzeichen verschiedenes Zeichen kein Anweisungssymbol ist, werden grundsätzlich als Datensatz interpretiert und unverändert in der aktuellen Zeile abgelegt. Im Dialogbetrieb führt eine leere Eingabe, die mit F1 statt DUE abgeschickt wurde, zum Ablegen einer Leerzeile (Zeile der Länge 0) in der aktuellen Zeile (siehe Abschnitt „Funktionstasten im L-Modus“ auf Seite 134).

Eingaben, bei denen die ersten beiden vom Leerzeichen verschiedenen Zeichen Anweisungssymbole sind, werden ebenfalls als Datensatz interpretiert, erfahren aber eine Sonderbehandlung:
Die Zeichen, die vor dem zweiten Anweisungssymbol stehen (dabei kann es sich nur um Leerzeichen und das erste Anweisungssymbol handeln), werden abgeschnitten.
Diese Sonderbehandlung erleichtert das Erstellen von Prozeduren im L-Modus (siehe auch Abschnitt „EDT-Prozeduren“ auf Seite 66). Die Eingabe wird nicht sofort als Anweisung ausgeführt, sondern als Anweisung abgespeichert und kann somit später mehrfach zur Ausführung gebracht werden.

Beispiele

Eingabe	**Interpretation**
`@RENUMBER`	Anweisung
`␣␣␣␣␣␣@␣␣␣␣␣␣␣RENUMBER`	Anweisung
`RENUMBER`	Datensatz, abgelegt wird ’`RENUMBER`’
`@@RENUMBER`	Datensatz, abgelegt wird ’`@RENUMBER`’
`@␣␣␣␣␣␣␣␣␣@RENUMBER`	Datensatz, abgelegt wird ’`@RENUMBER`’
`␣␣␣@␣␣␣␣␣@␣␣␣RENUMBER`	Datensatz, abgelegt wird ’`@␣␣␣RENUMBER`’

5.2.2 Eingabe von Datensätzen im Zeichen-, Hexadezimal- oder Binärformat

Im L-Modus kann die Eingabe von Datensätzen (aber nicht von Anweisungen) nicht nur in Form von Zeichenfolgen sondern auch als Folge von Hexadezimal- oder Binärzeichen erfolgen. Zwischen diesen Eingabeformaten wird mit der @INPUT-Anweisung (Format 3) umgeschaltet.

Standardmäßig erwartet der EDT im L-Modus die Eingabe in Form von Zeichenfolgen (@INPUT CHAR). Dabei wird der Zeichensatz der jeweiligen Eingabequelle (Datensichtstation, `SYSDTA`, Datei, Bibliothekselement, Arbeitsdatei) zugrunde gelegt. Da die eingegebenen Datensätze in die aktuelle Arbeitsdatei eingefügt werden und der Zeichensatz dieser Arbeitsdatei von dem der Eingabequelle abweichen kann, ist ggf. eine Umcodierung erforderlich. Die genauen Regeln dafür sind im Abschnitt „Zeichensätze“ auf Seite 48 beschrieben.

Wenn Datensätze im L-Modus als Folge von Hexadezimalzeichen bzw. Binärzeichen eingegeben werden sollen, muss dies explizit durch die Anweisung @INPUT HEX bzw. @INPUT BINARY eingestellt werden.

Die Hexadezimalzeichen bzw. Binärzeichen selbst werden in dem Zeichensatz der jeweiligen Eingabequelle erwartet. Die Interpretation der dadurch spezifizierten Codes erfolgt im Zeichensatz der aktuellen Arbeitsdatei. Werden Hexadezimalzeichen eingegeben, die keinem gültigen Zeichen in diesem Zeichensatz entsprechen, wird die Eingabe mit der Meldung `EDT5460` abgewiesen (siehe auch Abschnitt „Zeichensätze“ auf Seite 48).

Nach Einschalten der Hexadezimaleingabe bzw. der Binäreingabe werden nur noch Datensätze in gültiger Hexadezimal- bzw. Binärcodierung akzeptiert. Ungültige Eingaben werden mit der Meldung `EDT3902` bzw. `EDT3901` abgewiesen. Die Eingaben werden von links mit Nullen ergänzt, falls die Anzahl der eingegebenen Zeichen kein Vielfaches von 2 bzw. 8 ist.

Dadurch, dass in der hexadezimalen Darstellung jedes Zeichen mit mindestens zwei Bytes verschlüsselt wird, reduziert sich die Anzahl der pro Zeile eingebbaren Zeichen mindestens auf die Hälfte des sonst für die jeweilige Eingabequelle geltenden Wertes. Für die binäre Darstellung gilt entsprechend eine Reduzierung auf mindestens ein Achtel des sonst für die jeweilige Eingabequelle geltenden Wertes.

Beispiel

```
1.     ABC
2.     @INPUT HEX
2.     C1C2C3
3.     @INPUT BINARY
3.     110000011100001011000011
4.     @PRINT
1.0000 ABC
2.0000 ABC
3.0000 ABC
4.
```

5.2.3 Funktionstasten im L-Modus

Die folgende Beschreibung bezieht sich nur auf den Dialogbetrieb.

Im L-Modus wirken alle F-Tasten wie DUE, unabhängig davon, ob mit RDATA (Eingabeaufforderung ist *) oder WRTRD (Eingabeaufforderung ist die Zeilennummer) gelesen wird.

Die Taste DUE2 bewirkt die Übertragung des gesamten Bildschirminhalts einschließlich ggf. vorhandener LZE - Zeichen und Endemarken. Diese Systemeigenschaft führt im EDT dazu, dass eventuell mehrere Zeilen in die aktuelle Arbeitsdatei eingefügt werden. Da dies im Allgemeinen unerwünscht ist, sollte die DUE2 - Taste bei der Eingabe im L-Modus nicht benutzt werden.

Die Taste F1 wird im L-Modus dazu benutzt, eine Leerzeile (Zeile der Länge 0) einzugeben. Dazu ist an der Eingabeaufforderung nur F1 oder EM F1 einzugeben. Im Gegensatz dazu wird eine Eingabe von nur DUE oder EM DUE (wie bisher) ignoriert und führt zur Wiederholung der Eingabeaufforderung. Werden zusätzlich zu F1 oder EM F1 noch weitere Zeichen eingegeben, wirkt F1 wie DUE.

Alle K-Tasten ungleich K2 führen dazu, dass alle am Bildschirm eingegebenen Zeichen ignoriert werden und die Eingabeaufforderung erneut ausgegeben wird.

K2 bewirkt eine Unterbrechung des EDT und den Wechsel in den Systemmodus sofern dies nicht durch Systemeinstellungen verhindert wird (siehe Abschnitt „Unterbrechen des EDT-Laufs“ auf Seite 95).

5.2.4 Anweisungen im L-Modus

Folgende Anweisungen sind im L-Modus erlaubt:

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

Folgende Anweisungen sind *nicht* in EDT-Prozeduren erlaubt:
@DROP, @DIALOG, @INPUT (Format 1 und 2)

Folgende Anweisungen sind *nur* in EDT-Prozeduren erlaubt:
@GOTO, @DO (Format 2), @PARAMS

Das EDT-Anweisungssymbol (standardmäßig @) muss im L-Modus angegeben werden.
Die detaillierte Beschreibung der Anweisungen finden Sie in Kapitel 9.

6 Dateibearbeitung

Mit dem EDT können ISAM-Dateien, SAM-Dateien, Bibliothekselemente und POSIX-Dateien bearbeitet werden. Zur Bearbeitung werden sie in Arbeitsdateien geladen. Wird im Folgenden der Begriff Datei verwendet, so sind stets alle diese vier Dateitypen gemeint.

Daneben bietet der EDT Ein-/Ausgabe-Schnittstellen zu den BS2000-Systemdateien `SYSDTA`, `SYSOUT` und `SYSLST` an.

Die Behandlung von Zeichensätzen beim Lesen und Schreiben ist im Abschnitt „Zeichensätze“ auf Seite 48 beschrieben. Dateien mit Zeichensätzen, die in XHCS unbekannt sind, können nicht bearbeitet werden.

6.1 Dateitypen

In den folgenden Abschnitten werden die vom EDT unterstützten Dateitypen beschrieben.

6.1.1 SAM-Dateien

Beim Zugriff auf eine existierende SAM-Datei wird die Spezifikation eines Druckvorschubsteuerzeichens im Katalog ignoriert. Ebenso wird bei SAM-Dateien mit variabler Satzlänge die explizite Angabe einer Satzlänge im Katalog ignoriert.

Beim Schreiben in eine neue SAM-Datei wird diese normalerweise auf der Platte mit den folgenden Standard-Attributen angelegt:

– variable Satzlänge ohne Satzlängenangabe,
– Blockgröße 2 für Dateien auf NK4-Platten oder Blockgröße 1 sonst. Falls zum Zeitpunkt des Öffnens der längste Satz, der in die Datei geschrieben werden soll, nicht in diese Blockgröße passt, wird eine höhere Blockgröße (maximal 16) verwendet.

Sollen die Attribute neuer Dateien von diesem Standard abweichen, müssen die Dateiattribute und ein Dateikettungsname vor dem Schreiben in der `Task File Table` abgelegt sein und das Schreiben muss mit diesem Dateikettungsnamen erfolgen.
Es werden allerdings nur die im Abschnitt „Dateikettungsnamen“ auf Seite 145 beschriebenen Attribute ausgewertet.

Im Normalfall werden vom EDT zu bearbeitende Dateien (ganz oder teilweise) in Arbeitsdateien eingelesen. Die Bearbeitung von SAM-Dateien direkt auf der Platte (reale Bearbeitung) ist nicht möglich. Der EDT kann jedoch eine SAM-Datei in eine ISAM-Datei kopieren und diese dann zur realen Bearbeitung öffnen (siehe Anweisung @OPEN, Format 2, Seite 429).

Der Dateiname in den EDT-Anweisungen muss den Anforderungen des BS2000 Datenverwaltungssytems genügen. Der EDT prüft, ob ein Dateiname gültig ist.

Sätze der Länge `0` können vom EDT bearbeitet werden.

SAM-Dateien auf Magnetband können mit den Anweisungen @OPEN (Format 1) und @CLOSE nicht bearbeitet werden. Dafür sind die übrigen Anweisungen (@COPY, @WRITE) zu verwenden. Soll eine neue SAM-Datei auf Magnetband angelegt werden, so müssen vorher ihr Name und ihre Merkmale mit dem BS2000-Kommando `/CREATE-FILE` vereinbart werden.

Die Bearbeitung von SAM-Dateien mit Satzformat `UNDEFINED` ist im EDT nicht möglich. Ebenso wird die Bearbeitung von SAM-Dateien mit Zeichensatz `UTF16` und fester ungerader Satzlänge abgewiesen.

6.1.2 ISAM-Dateien

Beim Zugriff auf eine existierende ISAM-Datei wird die Spezifikation eines Druckvorschubsteuerzeichens im Katalog ignoriert. Ebenso wird bei ISAM-Dateien mit variabler Satzlänge die explizite Angabe einer Satzlänge im Katalog ignoriert.

Beim Schreiben in eine neue ISAM-Datei wird diese normalerweise mit den folgenden Standard-Attributen angelegt:

- variable Satzlänge ohne Satzlängenangabe,
- Schlüssel-Position 5,
- Schlüssellänge 16 bei Dateien mit dem Zeichensatz `UTF16` oder Schlüssellänge 8 bei Dateien mit anderem Zeichensatz und
- Blockgröße 2 für Dateien auf NK4-Platten oder Blockgröße 1 sonst. Falls zum Zeitpunkt des Öffnens der Datei der längste Satz, der in die Datei geschrieben werden soll, nicht in diese Blockgröße passt, wird eine höhere Blockgröße (maximal 16) verwendet.

Beim Schreiben sind standardmäßig keine Mehrfachschlüssel erlaubt.
Sollen die Attribute neuer Dateien von diesem Standard abweichen, müssen die Dateiattribute und ein Dateikettungsname vor dem Schreiben in der `Task File Table` abgelegt sein und das Schreiben muss mit diesem Dateikettungsnamen erfolgen.
Es werden allerdings nur die im Abschnitt „Dateikettungsnamen“ auf Seite 145 beschriebenen Attribute ausgewertet.

Im Normalfall werden vom EDT zu bearbeitende Dateien (ganz oder teilweise) in Arbeitsdateien eingelesen. Daneben gibt es auch die Möglichkeit, ISAM-Dateien direkt auf der Platte zu bearbeiten (reale Bearbeitung). Das ist nur in Arbeitsdatei 0 möglich.

Der Dateiname in den EDT-Anweisungen muss den Anforderungen des BS2000 Datenverwaltungssytems genügen. Der EDT prüft, ob ein Dateiname gültig ist.

Sätze, die nur aus dem ISAM-Schlüssel bestehen, können vom EDT bearbeitet werden.

Bei der Bearbeitung von ISAM-Dateien kann der ISAM-Schlüssel beim Einlesen als Zeilennummer übernommen werden, in den Datenbereich der Arbeitsdatei (als Zeileninhalt) eingelesen werden oder ganz ignoriert werden.

Beim Schreiben kann der ISAM-Schlüssel aus der Zeilennummer gebildet oder aus dem Datenbereich übernommen werden.
Wenn der ISAM-Schlüssel erhalten bleiben soll, muss er also vorher als Zeilennummer oder in den Datenbereich übernommen werden und darf nicht durch EDT-Anweisungen geändert werden.

Wird der ISAM-Schlüssel beim Einlesen als Zeilennummer übernommen, muss er numerisch sein (jedes der 8 Zeichen des Schlüssels ist aus dem Bereich `0..9`) und der Schlüssel `00000000` darf nicht vorkommen.

Hat die Datei eine kürzere Schlüssellänge als 8 (bzw. 16 bei `UTF16`-Dateien) und wird die Zeilennummer aus dem ISAM-Schlüssel gebildet, dann wird diese von links mit Nullen aufgefüllt. Beispielsweise wird bei einer `KEY-LENGTH`-Angabe von 4 der ISAM-Schlüssel `1234` als Zeilennummer `0000.1234` übernommen.

Hat die Datei eine kürzere Schlüssellänge als 8 (bzw. 16 bei `UTF16`-Dateien) und wird der ISAM-Schlüssel aus der Zeilennummer gebildet, so wird diese von links her verkürzt. Beispielsweise wird bei einer `KEY-LENGTH`-Angabe von 4 die Zeilennummer `1234.5678` als ISAM-Schlüssel `5678` geschrieben. Die Eindeutigkeit eines ISAM-Schlüssels ist damit unter Umständen nicht mehr gewährleistet. Sind für das Schreiben in die Datei Mehrfachschlüssel erlaubt, werden Arbeitsdateisätze, für die der gleiche ISAM-Schlüssel erzeugt wurde, in die Datei geschrieben. Andernfalls wird das Schreiben abgebrochen und die Meldung `EDT4208` (DVS-Fehlercode `0AAF`) ausgegeben.

Die Bearbeitung von ISAM-Dateien mit Nicht-Standard-Schlüsselposition (≠5 bei variabler Satzlänge oder ≠1 bei fester Satzlänge), mit einer Schlüssellänge, die größer ist, als im Standard definiert, mit nichtnumerischen Schlüsseln oder mit mehrfach auftretenden Schlüsselwerten kann ausschließlich dadurch erfolgen, dass der ISAM-Schlüssel in den Datenbereich übernommen wird.

Liegt der ISAM-Schlüssel im Datenbereich und wird er im EDT verändert, werden Zeilen vertauscht oder Sätze eingefügt, so muss der Benutzer sicherstellen, dass die Reihenfolge der Arbeitsdateisätze der Reihenfolge der ISAM-Schlüssel entspricht, da sonst das Schreiben mit der Meldung `EDT4208` (DVS-Fehlercode `0AAB`) abgewiesen wird.

Aus ISAM-Dateien mit mehrfach auftretenden Schlüsselwerten (`duplicate keys`) können alle Sätze eingelesen werden, wenn der Schlüssel nicht als Zeilennummer übernommen wird. Wird der ISAM-Schlüssel als Zeilennummer übernommen, so wird nur der jeweils letzte Satz mit dem gleichen Schlüssel eingelesen.
Für eine Bearbeitung solcher Dateien sollte daher der Schlüssel mit in den Datensatz übernommen werden. Beim Schreiben von Dateien mit Mehrfachschlüsseln muss das Attribut `DUPLICATE-KEY=YES` in der `Task File Table` abgelegt sein.
Die Bearbeitung von ISAM-Dateien mit Zeichensatz `UTF16` und fester ungerader Satzlänge oder ungerader Schlüssellänge ist nicht möglich.

Die Definition von Sekundärschlüsseln einer ISAM-Datei in einem Sekundärindex bleibt erhalten, wenn die Datei mit den Anweisungen @OPEN und @CLOSE bearbeitet wird und die Schlüsselfelder im Datenbereich nicht inkonsistent geändert werden.

Nach einer inkonsistenten Änderung wird die Meldung `EDT5246` ausgegeben und der Sekundärindex wird gelöscht. Der Sekundärindex wird ebenfalls gelöscht, wenn die Datei mit der Anweisung @SAVE oder @WRITE (Format 2) ganz überschrieben wird (nicht bei `UPDATE`).

6.1.3 POSIX-Dateien

Voraussetzung für die Bearbeitung von Dateien des POSIX-Dateisystems ist, dass das POSIX-Subsystem aktiviert ist.

Die POSIX-Dateinamen in den Anweisungen, die einen Zugriff auf POSIX-Dateien erlauben, sind Pfadnamen des POSIX-Dateisystems. Mit dem EDT kann man nicht innerhalb des POSIX-Dateisystems positionieren. Falls nicht ein absoluter Pfadname angegeben ist (beginnend mit `'/'`), beziehen sich die Dateinamen immer auf das aktuelle Verzeichnis. Beim Aufruf aus dem BS2000 ist dies das Home-Verzeichnis des Benutzers.

Der POSIX-Dateiname in den EDT-Anweisungen muss den Anforderungen des POSIX-Dateisystems genügen. Insbesondere darf die Länge maximal 1023 Bytes betragen. Er kann direkt als Zeichenfolge oder indirekt über eine Zeichenfolgevariabe angegeben werden. Enthält der Name einer POSIX-Datei Leerzeichen oder andere Sonderzeichen, so muss er über eine Zeichenfolgevariable angegeben werden.

In POSIX-Dateinamen werden Groß- und Kleinbuchstaben unterschieden. Deshalb sollte bei Eingabe von einer Datensichtstation @PAR LOWER=ON eingeschaltet sein.

Der EDT liest die Daten zeichenweise ein. Das Satzende wird durch das (zeichensatzspezifische) Satzendekennzeichen (beispielsweise `X'15'` in EBCDIC- bzw. `X'0A'` in ISO-Zeichensätzen) erkannt. Ist ein Satz erkannt, wird er in die aktuelle Arbeitsdatei gebracht. Folgen zwei Satzendekennzeichen aufeinander, wird ein Satz der Länge 0 in der Arbeitsdatei erzeugt.

Ist beim Lesen aus einer POSIX-Datei nach 32768 Zeichen noch kein Satzende erkannt, dann gibt der EDT die Meldung EDT1253 aus, schneidet die Eingabe ab und ignoriert die Zeichen bis zum nächsten Satzendekennzeichen.

Beim Schreiben des Arbeitsdateiinhalts in die POSIX-Datei wird nach jedem Arbeitsdateisatz ein (zeichensatzspezifisches) Satzendekennzeichen (beispielsweise X'15' in EBCDIC- bzw. X'0A' in ISO-Zeichensätzen) geschrieben.

Für Sätze der Länge 0 wird nur ein Satzendekennzeichen geschrieben.

Beim Schreiben von schreibgeschützten POSIX-Dateien unter der Benutzerkennung TSOS wird im Dialogbetrieb der Benutzer mit der Meldung EDT0244 gefragt, ob er den Schreibzugriff erlauben will. Wenn nicht, gibt der EDT die Meldung EDT5312 aus. Im Stapelbetrieb gibt der EDT die Meldung EDT5312 aus und erlaubt den Schreibzugriff nicht.

Anders als bei BS2000-Dateien oder Bibliothekselementen erlaubt das System das mehrfache Öffnen der gleichen POSIX-Datei. Der EDT verhindert dies ebenfalls nicht. Der Anwender muss also die entsprechende Vorsicht walten lassen.

6.1.4 Bibliothekselemente

Mit dem EDT können Elemente von Programmbibliotheken (PLAM-Bibliotheken, von *Program Library Access Method*) bearbeitet werden. Nähere Informationen zu diesen Bibliotheken können dem Handbuch LMS [14] entnommen werden.

Programmbibliotheken werden in diesem Handbuch kurz als Bibliotheken bezeichnet.

Delta-Elemente können nicht mit den EDT-Anweisungen bearbeitet werden. Ein Lesezugriff ist zwar möglich, für Modifikationen von Delta-Elementen müssen jedoch die Anweisungen des LMS im EDT verwendet werden. Das genaue Vorgehen ist im Handbuch LMS [14] beschrieben.

Der Name der Bibliothek in den EDT-Anweisungen muss den Anforderungen genügen, die das BS2000 Datenverwaltungssystem an Dateinamen stellt. Er kann direkt als Zeichenfolge oder indirekt über eine Zeichenfolgevariable angegeben werden. Ist die zu öffnende Bibliothek Teil einer Dateigenerationsgruppe, muss der Bibliotheksname (wegen der Klammern) über eine Zeichenfolgevariable angegeben werden.

Sätze der Länge 0 können vom EDT bearbeitet werden.

Elemente sind in Bibliotheken über ihre Elementbezeichnung einzeln ansprechbar.

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

```
elname[(vers)][,eltype]
```

Dabei bezeichnet elname den Namen des Elements, vers die Versionsbezeichnung des Elements und eltype den Typ des Elements.

Die Angabe der Version und des Elementtyps ist wahlfrei. Wird in einer Anweisung kein Wert für `vers` eingegeben, wird beim Lesen die höchste existierende Version und beim Schreiben die höchstmögliche Version verwendet. Wird in einer Anweisung kein Wert für `eltype` eingegeben oder ist der Wert `*STD`, wird der bei @PAR ELEMENT-TYPE angegebene Wert eingesetzt. Die Voreinstellung beim Start des EDT ist `S`.

Die Elementbezeichnung muss den Namenskonventionen genügen, wie sie Handbuch LMS [14] festgelegt sind.

In den Anweisungen zum Lesen und Schreiben ist die Angabe des Elementtyps auf textartige Typen und benutzerdefinierte Typen beschränkt. Die textartigen Standardtypen sind:

Typ	Elementinhalt
`S`	Quellprogramm
`M`	Makros
`P`	Druckaufbereitete Daten
`J`	Prozeduren
`D`	Textdaten
`X`	Daten beliebigen Formats

Für die Standard-Typen wird nicht geprüft, ob der Inhalt des Elements tatsächlich textartig ist. Ist der Elementtyp ein benutzerdefinierter Typ, so wird nicht geprüft, ob er von einem textartigen Basistyp abgeleitet ist.

Elemente mit Format B Sätzen können vom EDT nicht bearbeitet werden. Bei Format A Sätzen berücksichtigt der EDT nur Sätze der Satzart 1, andere Satzarten lässt der EDT unverändert.

6.2 Grundsätzliches zum Lesen und Schreiben von Daten

Sind beim Einlesen einer Datei Zeichen in dieser Datei enthalten, die im Zeichensatz der Arbeitsdatei nicht unterstützt werden, wird die Datei nicht eingelesen, wenn nicht ein Ersatzzeichen definiert ist, das dann eingesetzt wird. Enthält die Datei eine ungültige Bytefolge (in Unicode-Zeichensätzen möglich), wird die Datei nicht eingelesen.

Da der EDT intern Zeilen bis zur Länge von 32768 Zeichen unterstützt, kann es beim Schreiben in DVS-Dateien oder Bibliothekselemente möglich sein, dass ein Satz nicht geschrieben werden kann, weil er zu lang ist. Insbesondere beim Schreiben in eine Datei mit dem Zeichensatz `UTF16`, `UTFE` oder `UTF8` kann dies bei deutlich kürzeren Zeilen auftreten, wenn nämlich der Arbeitsdateisatz hinreichend viele Zeichen enthält, die in 2 oder in 3 Byte codiert sind. Ebenso kann das leicht bei Dateien mit fester Satzlänge auftreten. Kann ein Satz nicht geschrieben werden, weil er zu lang ist, gibt der EDT die Meldung `EDT5444` aus und bricht den Schreibvorgang ab. Der Anwender muss dann selbst dafür sorgen, dass er Zeilen entsprechend verkürzt oder ein anderes Dateiformat benutzt.

Ist beim Schreiben in eine Datei mit fester Satzlänge ein Arbeitsdateisatz kürzer als die Satzlänge der Datei, dann füllt der EDT den Satz bis zur definierten Satzlänge mit Leerzeichen auf.

Für eine DVS-Datei wird die benötigte Dateigröße vor dem Schreiben abgeschätzt und wenn möglich die erforderliche Seitenanzahl als *Primary-Allocation* zugewiesen, um die Aufsplitterung der Datei in zu viele Extents zu vermeiden.

Nachdem eine DVS-Datei nach dem Schreiben geschlossen wurde (Anweisungen @WRITE, @CLOSE, @SAVE), gibt der EDT normalerweise den nicht benötigten Speicherplatz der Datei frei. Diese Freigabe wird nicht durchgeführt, wenn entweder vor dem Schreiben der Auftragsschalter 7 gesetzt wurde oder der Dateiname mit einer Benutzerkennung beginnt.

Beim Schreiben mit den alten Anweisungen (siehe Kapitel 9) @WRITE (Format 2) und @SAVE kann man das auch erreichen, wenn dem Dateinamen ein fest vorgegebener Dateikettungsname zugeordnet wird und anstelle des Dateinamens die symbolische Bezeichnung `'/'` angegeben wird.

6.3 Lesen und Schreiben beliebiger Dateitypen

Mit den folgenden Anweisungen ist es möglich, Dateien aller im EDT unterstützter Dateitypen zu bearbeiten. Es wird empfohlen, nur noch diese neuen Anweisungen zur Dateibearbeitung zu verwenden.

6.3.1 Lesen

Eine Datei kann mit der Anweisung @OPEN (Format 1) in die aktuelle Arbeitsdatei zur Bearbeitung eingelesen werden oder neu erzeugt werden. Sie bleibt geöffnet, bis sie durch @CLOSE geschlossen wird.

Eine Datei kann auch mit der Anweisung @COPY (Format 1) in die aktuelle Arbeitsdatei eingelesen werden. Nach dem Einlesen ist die Datei wieder geschlossen.

Beim Lesen von ISAM-Dateien kann mit dem Operand KEY der Anweisung @OPEN oder @COPY ausgewählt werden, ob der ISAM-Schlüssel ignoriert, als Zeilennummer oder in den Datenbereich übernommen werden soll.

Eine Datei kann mit der Anweisung @INPUT (Format 1) als Prozedur ausgeführt werden.

6.3.2 Schreiben

Die aktuelle Arbeitsdatei kann mit @CLOSE in eine Datei zurück geschrieben werden, wenn diese vorher mit @OPEN (Format 1) eingelesen oder neu angelegt wurde. Nach dem Schreiben wird die Datei geschlossen und die Arbeitsdatei wird gelöscht.

Die aktuelle Arbeitsdatei kann mit @WRITE (Format 1) in eine neue Datei geschrieben werden. Nach dem Schreiben bleibt die Arbeitsdatei erhalten.

Die aktuelle Arbeitsdatei kann mit @WRITE (Format 1) in eine existierende Datei geschrieben werden. Der alte Dateiinhalt wird dabei vollständig ersetzt. Nach dem Schreiben bleibt die Arbeitsdatei erhalten.

Die aktuelle Arbeitsdatei kann mit @WRITE (Format 1) in die mit @OPEN (Format 1) geöffnete Datei zurück geschrieben werden. Nach dem Schreiben bleibt die Datei geöffnet und die Arbeitsdatei bleibt erhalten.

Beim Neuanlegen einer SAM- oder ISAM-Datei vor dem Schreiben werden normalerweise die Standardattribute des Dateityps verwendet. Sollen vom Standard abweichende Dateiattribute verwendet werden, müssen sie zusammen mit einem frei wählbaren Dateikettungsnamen vor der Anweisung @OPEN oder @WRITE in der `Task File Table` abgelegt werden und die Anweisung @OPEN oder @WRITE muss mit diesem Dateikettungsnamen erfolgen.

6.3.3 Dateikettungsnamen

Zum Ansprechen vom SAM- und ISAM-Dateien können bei den neuen Anweisungen auch frei wählbare Dateikettungsnamen verwendet werden. Dabei muss zumindest der Dateiname in der `Task File Table` zwingend angegeben sein.

Für **existierende** Dateien werden auch nur einige der Attribute aus der `Task File Table` übernommen. Diese sind (falls angegeben):

```
FILE-NAME
WRITE-CHECK
```

Für **ISAM**-Dateien zusätzlich noch die Attribute:

```
WRITE-IMMEDIATE
DUPLICATE-KEY
PADDING-FACTOR
SHARED-UPDATE
```

Alle anderen Angaben werden ignoriert und von der existierenden Datei übernommen.
Beim **Neuanlegen** von Dateien werden mehr der in der `Task File Table` spezifizierten Attribute übernommen, diese sind (falls angegeben):

```
FILE-NAME
ACCESS-METHOD
RECORD-FORMAT
RECORD-SIZE
PRINT-CONTROL
BLOCK-SIZE
BLOCK-CONTROL
WRITE-CHECK
```

Für **ISAM**-Dateien zusätzlich noch die Attribute:

```
KEY-LENGTH
KEY-POSITION
WRITE-IMMEDIATE
DUPLICATE-KEY
PADDING-FACTOR
VALUE-FLAG-LENGTH
PROPAGATE-VALUE-FLAG
LOGICAL-FLAG-LENGTH
SHARED-UPDATE
```

Alle anderen Angaben werden immer ignoriert.

Die Angabe zur Zugriffsmethode wird nur verwendet, wenn in der entsprechenden Anweisung keine explizite Angabe erfolgt.

6.4 Eigenschaften der alten Dateizugriffsanweisungen

Vorbemerkung

Die Spezifizierung **alt** bezieht sicht **nicht** auf die Unterscheidung dieser Anweisungen im Unicode- und Kompatibilitätsmodus, sondern auf die ursprünglich für den EDT entwickelten Dateizugriffsanweisungen.
Sie sind für beide Betriebsmodi weiterhin gültig.

Die alten Dateizugriffsanweisungen @READ, @GET, @WRITE (Format 2), @SAVE, @ELIM und @OPEN (Format 2) haben einige Einschränkungen. So können mit ihnen nicht alle Dateitypen bearbeitet werden, und manche Zugriffe sind nur über Umwege (Definition von Dateikettungsnamen) möglich. Für die unterstützten Dateitypen bieten sie einige Spezialfunktionen, die in diesem Abschnitt beschrieben werden sollen.

Wird eine Datei mit den Anweisungen @READ oder @GET in die aktuelle Arbeitsdatei eingelesen, dann wird die Datei unmittelbar nach dem Einlesen wieder geschlossen. Erst durch die Anweisungen @WRITE oder @SAVE wird die Datei dann wieder zum Schreiben geöffnet. Wird in diesen Anweisungen der Operand `UPDATE` nicht angegeben, wird dabei die alte Datei gelöscht und eine neue Datei unter dem Namen der gelöschten Datei angelegt. Während der Bearbeitung ist sie für andere Benutzer nicht gesperrt. Beim Schreiben kann es daher vorkommen, dass zwischenzeitliche Änderungen anderer Benutzer überschrieben werden.

6.4.1 Dateinamen voreinstellen

Nach dem Einlesen mit @READ oder @GET ist die Arbeitsdatei mit dem Dateinamen verknüpft (impliziter lokaler @FILE-Eintrag). Sie bewirkt die Voreinstellung des Dateinamens für die Anweisungen @WRITE (Format 2) und @SAVE.

Eine Verknüpfung kann auch explizit mit der Anweisung @FILE erfolgen. Dann wirkt die Voreinstellung des Dateinamens für die Anweisungen @READ, @GET, @WRITE (Format 2), @SAVE, @ELIM und @OPEN (Format 2).

Die Verknüpfung wird wieder aufgehoben durch das vollständige Löschen der Arbeitsdatei mit @DELETE (Format 2) bzw. mit Anweisungen, die implizit ein @DELETE (Format 2) ausführen, oder explizit mit der Anweisung @FILE.

Zusätzlich kann man mit der Anweisung @FILE eine Voreinstellung definieren, die global für alle Arbeitsdateien gilt. Ein expliziter (auch globaler) @FILE-Eintrag hat immer Vorrang vor einem impliziten @FILE-Eintrag.

6.4.2 Teilweises Lesen und Schreiben

Mit den Anweisungen @READ und @GET kann eine Auswahl der Sätze getroffen werden, die aus der Datei in die Arbeitsdatei eingelesen werden. Dabei kann die Reihenfolge der Sätze in der Datei verändert werden. Sätze können mehrfach in die Arbeitsdatei eingelesen werden.

Es kann auch eine Auswahl der Zeichen (Spalten) getroffen werden, die aus den ausgewählten Sätzen in die Arbeitsdatei eingelesen werden. Auch hier kann die Reihenfolge der Zeichen verändert werden und Zeichen können mehrfach eingelesen werden. Werden Spaltenwerte angegeben, die die Länge eines Satzes überschreiten, so werden dafür Leerzeichen in die Arbeitsdatei eingelesen. Für die Anweisung @INPUT (Format 2) gilt entsprechendes.

Werden Zeilenbereiche zur Auswahl angegeben, dann werden nur die angegebenen (und evtl. benachbarte) Zeilen auf ungültige Byte-Sequenzen untersucht, allerdings komplett, unabhängig von einer eventuellen Spaltenauswahl. In nicht gelesenen Zeilen vorkommende ungültige Byte-Sequenzen werden nicht bemerkt.

Mit den Anweisungen @WRITE (Format 2) und @SAVE kann eine Auswahl der Arbeitsdateisätze getroffen werden, die in die Datei geschrieben werden. Dabei kann die Reihenfolge verändert werden, in der die Arbeitsdateisätze geschrieben werden.

Arbeitsdateisätze können mehrfach geschrieben werden. Es kann auch eine Auswahl der Zeichen (Spalten) getroffen werden, die aus den ausgewählten Arbeitsdateisätzen in die Datei geschrieben werden. Auch hier kann die Reihenfolge der Zeichen verändert werden und Zeichen können mehrfach geschrieben werden. Werden Spaltenwerte angegeben, die die Länge eines Arbeitsdateisatzes überschreiten, so werden dafür Leerzeichen in die Datei geschrieben.

6.4.3 Versionsnummern

In den Anweisungen @GET, @SAVE, @READ, @WRITE (Format 2), @OPEN (Format 2), @ELIM, @INPUT (Format 2), @FILE und @UNSAVE kann zusätzlich zum Dateinamen eine Versionsnummer zwischen 0 und 255 oder * als aktuelle Versionsnummer angegeben werden. Sie dient dem Schutz vor unabsichtlichem Überschreiben.

Eine erstmalig angelegte Datei erhält nach dem Schreiben auf Platte die Versionsnummer 1. Bei jedem Schreiben der Datei wird die Versionsnummer vom DVS um 1 erhöht.
Die Versionsnummer wird bis 255 hoch gezählt. Die darauf folgende Versionsnummer ist wieder 0.

Lesende Zugriffe (@GET, @READ) mit einer Versionsnummer ungleich der aktuellen werden ausgeführt. Zusätzlich wird die richtige Versionsnummer in der Meldung `EDT0902` angezeigt.

Bei Angabe von * als Versionsnummer wird ebenfalls die aktuelle Versionsnummer in der Meldung EDT0902 angezeigt. Bei @INPUT (Format 2) ist die Angabe zwar möglich, wird aber vollständig ignoriert. Schreibende Zugriffe (@SAVE, @WRITE (Format 2), @ELIM, @UNSAVE) sowie das Öffnen zur realen Bearbeitung (@OPEN, Format 2) mit einer Versionsnummer ungleich der aktuellen werden nicht ausgeführt. Stattdessen wird die richtige Versionsnummer in der Meldung EDT4985 angezeigt.
Bei expliziter oder symbolischer Angabe der aktuellen Versionsnummer wird nach dem Schreiben die neue, um 1 erhöhte Versionsnummer in der Meldung EDT0902 angezeigt.

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.

Die Versionsnummer des EDT darf nicht verwechselt werden mit der Generationsnummer von Dateigenerationsgruppen im BS2000. Diese ist Bestandteil des Dateinamens.
Für jede Generation kann zusätzlich eine Versionsnummer angegeben werden.
In ihrer Bedeutung entspricht die EDT-Versionsnummer eher der Variantennummer von Bibliothekselementen.

6.4.4 Dateikettungsnamen

In den Anweisungen @ELIM, @GET, @SAVE, @READ und @WRITE (Format 2) kann man auch '/' anstelle des Dateinamens angeben, wenn man der Datei vor dem Zugriff durch den EDT den festen Dateikettungsnamen EDTISAM (bei @ELIM, @GET, @SAVE) bzw. EDTSAM (bei @READ, @WRITE) zugeordnet hat.

Ist einer Datei der feste Dateikettungsname EDTISAM zugeordnet, werden beim Öffnen der Datei zum Lesen und zum Schreiben mit @GET bzw. @SAVE die Dateiattribute aus der Task File Table verwendet. Ist der Datei EDTSAM zugeordnet, werden beim Öffnen der Datei zum Lesen und zum Schreiben mit @READ bzw. @WRITE die Dateiattribute aus der Task File Table verwendet.

Wenn beim Schreiben die Dateiattribute vom Standard-Dateiformat des EDT (siehe Abschnitte ISAM-Dateien und SAM-Dateien) abweichen sollen, kann man die Dateien mit den Anweisungen @SAVE und @WRITE (Format 2) nur schreiben, wenn man vorher die abweichenden Attribute in die Task File Table eingetragen hat und der Datei den festen Dateikettungsnamen EDTISAM bzw. EDTSAM zugeordnet hat.

Wird beim Schreiben mit @SAVE oder @WRITE die symbolische Bezeichnung '/' für eine Datei angegeben, wird nach dem Schließen der Datei der überschüssige Speicherplatz nicht freigegeben und es erfolgt keine Sicherungsabfrage (EDT0903), ob eine bestehende Datei überschrieben werden soll.

6.5 Lesen und Schreiben von SAM-Dateien mit den alten Anweisungen

Auf SAM-Dateien kann auch weiter mit den alten Anweisungen @READ und @WRITE (Format 2) zugegriffen werden. Es wird aber empfohlen, nur noch die neuen Anweisungen (siehe Abschnitt „Lesen und Schreiben beliebiger Dateitypen“ auf Seite 144) zu verwenden.

Mit diesen Anweisungen ist eine Auswahl bestimmter Zeilen und Spalten sowie die Angabe einer Versionsnummer möglich. Wenn der Datei der feste Dateikettungsname `EDTSAM` zugeordnet ist, kann man anstelle des Dateinamens auch `''` angeben (siehe Abschnitt „Eigenschaften der alten Dateizugriffsanweisungen“ auf Seite 146).

Mit diesen Anweisungen ist es möglich, die ersten 8 Zeichen einer SAM-Datei (bei `UTF16`-Dateien sind dies die ersten 16 Byte) als Zeilennummer zu interpretieren.
Beim Schreiben wird dafür die jeweilige EDT-Zeilennummer verwendet, der Arbeitsdatei-satz wird ab Zeichen 9 geschrieben.
Beim Lesen werden die ersten 8 Zeichen des einzulesenden Satzes als Zeilennummer übernommen, der Rest des Satzes ab Zeichen 9 wird in die Arbeitsdatei übernommen. Dabei wird geprüft, ob die Zeichen der Zeilennummer numerisch sind (Details siehe bei den einzelnen Anweisungen).

6.5.1 Lesen

Eine SAM-Datei kann mit der alten Anweisung @READ in die aktuelle Arbeitsdatei zur Bearbeitung eingelesen werden. Nach dem Einlesen ist die Datei wieder geschlossen.

Eine impliziter lokaler @FILE-Eintrag wird angelegt, der in einer nachfolgenden @WRITE-Anweisung verwendet werden kann.

6.5.2 Schreiben

Die aktuelle Arbeitsdatei kann mit der alten Anweisung @WRITE (Format 2) in eine SAM-Datei geschrieben werden. Diese kann bereits existieren oder sie wird vor dem Schreiben neu angelegt. Nach dem Schreiben bleibt die Arbeitsdatei erhalten.

Existierende SAM-Dateien mit fester Satzlänge können im gleichen Satzformat nur überschrieben werden, wenn dieses Dateiattribut zusammen mit dem Dateinamen und dem festen Dateikettungsnamen `EDTSAM` vor der Anweisung @WRITE in der `Task File Table` abgelegt worden ist.

Beim Neuanlegen einer SAM-Datei vor dem Schreiben werden normalerweise die Standardattribute verwendet. Sollen vom Standard abweichende Dateiattribute verwendet werden, müssen sie zusammen mit dem Dateinamen und dem festen Dateikettungsnamen `EDTSAM` vor der Anweisung @WRITE in der `Task File Table` abgelegt werden.

6.6 Lesen und Schreiben von ISAM-Dateien mit den alten Anweisungen

Auf ISAM-Dateien kann auch weiter mit den alten Anweisungen @GET, @SAVE und @ELIM zugegriffen werden. Es wird aber empfohlen, nur noch die neuen Anweisungen (siehe Abschnitt „Lesen und Schreiben beliebiger Dateitypen“ auf Seite 144) zu verwenden.

Mit diesen Anweisungen ist eine Auswahl bestimmter Zeilen und Spalten sowie die Angabe einer Versionsnummer möglich.
Wenn der Datei der feste Dateikettungsname `EDTISAM` zugeordnet ist, kann man in den Anweisungen @GET und @SAVE anstelle des Dateinamens auch `'/'` angeben (siehe Abschnitt „Eigenschaften der alten Dateizugriffsanweisungen“ auf Seite 146).

6.6.1 Lesen

Eine ISAM-Datei kann mit der alten Anweisung @GET in die aktuelle Arbeitsdatei zur Bearbeitung eingelesen werden. Nach dem Einlesen ist die Datei wieder geschlossen.

Eine impliziter lokaler @FILE-Eintrag wird angelegt, der in einer nachfolgenden @SAVE-Anweisung verwendet werden kann.

Eine Übernahme des ISAM-Schlüssels als Zeilennummer erfolgt normalerweise nicht, kann aber durch den Operanden `NORESEQ` veranlasst werden.
Eine Übernahme des ISAM-Schlüssels in den Datenbereich ist mit @GET nicht möglich, hierzu müssen der Dateiname, der feste Dateikettungsname `EDTSAM` und die Zugriffsmethode ISAM in der `Task File Table` abgelegt werden und die Anweisung @READ `'/'` verwendet werden. Die Angabe des Dateinamens in der @READ-Anweisung ist hier nicht möglich.

Soll eine ISAM-Datei mit Nicht-Standard-Schlüsselposition (≠5 bei variabler Satzlänge oder ≠1 bei fester Satzlänge), mit einer Schlüssellänge, die größer ist, als im Standard definiert, mit nichtnumerischen Schlüsseln oder mit mehrfach auftretenden Schlüsselwerten eingelesen werden, muss der ISAM-Schlüssel in den Datenbereich übernommen werden.
Hierzu müssen die abweichenden Dateiattribute zusammen mit dem Dateinamen, dem festen Dateikettungsnamen `EDTSAM` und der Zugriffsmethode ISAM in der `Task File Table` abgelegt werden. Danach wird die Datei mit der Anweisung @READ `'/'` eingelesen.

6.6.2 Schreiben

Die aktuelle Arbeitsdatei kann mit der alten Anweisung @SAVE in eine ISAM-Datei geschrieben werden. Diese kann bereits existieren oder sie wird vor dem Schreiben neu angelegt. Nach dem Schreiben bleibt die Arbeitsdatei erhalten.

Existierende ISAM-Dateien mit fester Satzlänge und/oder mit kleinerer Schlüssellänge, als im Standard definiert, können unter Beibehaltung dieser Attribute nur überschrieben werden, wenn die abweichenden Dateiattribute zusammen mit dem Dateinamen und dem festen Dateikettungsnamen `EDTISAM` vor der Anweisung @SAVE in der `Task File Table` abgelegt worden sind.

Existierende ISAM-Dateien, deren ISAM-Schlüssel in den Datenbereich übernommen wurde, können überschrieben werden, wenn die abweichenden Dateiattribute zusammen mit dem Dateinamen, dem festen Dateikettungsnamen `EDTSAM` und der Zugriffsmethode ISAM in der `Task File Table` abgelegt worden sind und mit der Anweisung @WRITE `'/'` geschrieben wird. Die Angabe des Dateinamens in der @WRITE-Anweisung ist hier nicht möglich.

Beim Neuanlegen einer ISAM-Datei vor dem Schreiben werden normalerweise die Standardattribute verwendet.

Soll beim Neuanlegen einer Datei der ISAM-Schlüssel aus der Zeilennummer gebildet werden, müssen eventuell vom Standard abweichende Dateiattribute zusammen mit dem Dateinamen und dem festen Dateikettungsnamen `EDTISAM` vor der Anweisung @SAVE in der `Task File Table` abgelegt werden.

Soll beim Neuanlegen einer Datei der ISAM-Schlüssel aus dem Datenbereich entnommen werden, müssen eventuell vom Standard abweichende Dateiattribute zusammen mit dem Dateinamen, dem festen Dateikettungsnamen `EDTSAM` und der Zugriffsmethode ISAM in der `Task File Table` abgelegt werden. Danach wird mit der Anweisung @WRITE `'/'` geschrieben. Die Angabe des Dateinamens in der @WRITE-Anweisung ist hier nicht möglich.

Mit der Anweisung @ELIM kann man eine ISAM-Datei teilweise oder ganz löschen.

6.7 Reale Bearbeitung von ISAM-Dateien

ISAM-Dateien können direkt auf der Platte bearbeitet werden, ohne dass sie vorher vollständig in den Speicherbereich des EDT eingelesen wurden. Dafür muss Arbeitsdatei 0 die aktuelle Arbeitsdatei sein.
Sie muss leer sein oder eine Datei muss zur realen Bearbeitung geöffnet sein.
Im ersten Fall muss der Zeichensatz der Arbeitsdatei 0 `*NONE` sein oder mit dem Zeichensatz der zu öffnenden Datei übereinstimmen.
Im zweiten Fall wird die geöffnete Datei implizit geschlossen. Dabei wird implizit die Arbeitsdatei gelöscht und der Zeichensatz `*NONE` eingestellt.

6.7.1 Öffnen

Zum Öffnen einer ISAM-Datei zur realen Bearbeitung wird die Anweisung @OPEN (Format 2) verwendet. Falls die Datei noch nicht existiert, wird sie vor dem Öffnen neu angelegt. Dabei werden die Standard-Attribute für ISAM-Dateien verwendet, wenn nicht über den festen Dateikettungsnamen `EDTMAIN` etwas anderes festgelegt wurde. Die Datei bleibt bis zum Ende ihrer Bearbeitung durch explizites oder implizites Schließen geöffnet.

Wird der Dateikettungsname `EDTMAIN` verwendet, muss der Dateiname in der Anweisung @OPEN trotzdem angegeben werden, die Angabe von `'/'` ist nicht möglich.

Die reale Bearbeitung von ISAM-Dateien mit Nicht-Standard-Schlüsselposition oder Schlüssellänge größer als im Standard definiert wird nicht unterstützt.

Die reale Bearbeitung von ISAM-Dateien mit mehrfach auftretenden Schlüsselwerten oder nichtnumerischem Schlüssel wird nicht unterstützt. Wird trotzdem eine solche Datei geöffnet, so führt dies zum Fehler, sobald einer dieser Sätze bearbeitet wird.

Hat die Datei eine kürzere Schlüssellänge als 8 (bzw. 16 bei `UTF16`-Dateien), dann wird beim Einlesen die Zeilennummer von links mit Nullen aufgefüllt. Beispielsweise wird bei einer `KEY-LENGTH`-Angabe von 4 der ISAM-Schlüssel `1234` als Zeilennummer `0000.1234` übernommen. Außerdem wird in diesem Fall die aktuelle Schrittweite auf einen *angepassten* Wert eingestellt. Hat die aktuelle Schrittweite zum Zeitpunkt des Öffnens den Standardwert 1 oder ist die eingestellte Schrittweite größer als die größtmögliche Zeilennummer, dann wird sie für Schlüssellängen kleiner oder gleich 2 auf den Wert `0.0001`, für Schlüssellängen kleiner oder gleich 5 auf den Wert `0.01` und ansonsten auf den Wert `1.0` gesetzt.

6.7.2 Bearbeiten

In die aktuelle Arbeitsdatei wird nur der gerade benötigte Teil der ISAM-Datei eingelesen. Im F-Modus sind das die Sätze, die ganz oder teilweise am Bildschirm angezeigt werden, im L-Modus der in einer EDT-Anweisung angegebene Zeilenbereich. Beim Einlesen wird der ISAM-Schlüssel als Zeilennummer übernommen.

Im F-Modus wird jede Änderung nach dem Drücken der Taste DUE oder einer anderen Funktionstaste zur Datenübertragung sofort in die Plattendatei übernommen, im L-Modus nach der Ausführung einer EDT-Anweisung. Dabei werden nur die geänderten Sätze zurück geschrieben. Beim Schreiben wird der ISAM-Schlüssel aus der Zeilennummer gebildet. Das Umnummerieren von Zeilen ist bei der realen Bearbeitung nicht möglich.

Der Zeichensatz einer Datei kann bei der realen Bearbeitung nicht geändert werden. Eine @CODENAME-Anweisung wird dann mit der Meldung `EDT5452` abgewiesen.

Zeilennummern, die länger sind als die Schlüssellänge, können bei realer Bearbeitung nicht erzeugt werden. Entsprechende Anweisungen werden ebenso wie das Einstellen einer zu großen Schrittweite mit einer Fehlermeldung abgewiesen.

Fehler in der Datei (z.B. nicht-numerische Schlüssel, doppelte Schlüssel, illegale Byte-Folgen u.a.) werden möglicherweise erst bemerkt, wenn die entsprechenden Sätze gelesen werden sollen. Beim Öffnen werden jedenfalls nicht alle Sätze der Datei eingelesen. Später festgestellte Fehler meldet dann das jeweilige Kommando, bei dessen Abarbeitung er auftrat. In diesen Fällen wird dann immer die real geöffnete Datei automatisch geschlossen.

6.7.3 Schließen

Eine zur realen Bearbeitung geöffnete Datei wird mit @CLOSE geschlossen. Sie wird auch geschlossen, wenn eine der Anweisungen @HALT, @EXEC, @LOAD, @OPEN (Format 2), @DELETE (Format 2) oder @DROP eingegeben wird.

Nach dem Schließen wird die Arbeitsdatei gelöscht.

6.8 Lesen und Schreiben von POSIX-Dateien mit den alten Anweisungen

Auf POSIX-Dateien kann auch weiter mit den alten Anweisungen @XOPEN, @XCOPY und @XWRITE zugegriffen werden. Es wird aber empfohlen, nur noch die neuen Anweisungen (siehe Abschnitt „Lesen und Schreiben beliebiger Dateitypen“ auf Seite 144) zu verwenden.

6.8.1 Lesen

Eine POSIX-Datei kann mit der Anweisung @XOPEN in die aktuelle Arbeitsdatei zur Bearbeitung eingelesen werden. Sie bleibt geöffnet, bis sie durch @CLOSE geschlossen wird.

Eine POSIX-Datei kann auch mit der Anweisung @XCOPY in die aktuelle Arbeitsdatei eingelesen werden. Nach dem Einlesen ist die Datei wieder geschlossen.

6.8.2 Schreiben

Die aktuelle Arbeitsdatei kann mit @CLOSE in eine POSIX-Datei zurück geschrieben werden, wenn diese vorher mit @XOPEN eingelesen wurde oder wenn vorher mit @XOPEN, MODE=NEW eine neue POSIX-Datei angelegt wurde. Nach dem Schreiben wird die Datei geschlossen und die Arbeitsdatei wird gelöscht.

Die aktuelle Arbeitsdatei kann mit @XWRITE in eine neue POSIX-Datei geschrieben werden. Nach dem Schreiben bleibt die Arbeitsdatei erhalten.

Die aktuelle Arbeitsdatei kann mit @XWRITE in eine existierende POSIX-Datei geschrieben werden. Der alte Dateiinhalt wird dabei vollständig ersetzt. Nach dem Schreiben bleibt die Arbeitsdatei erhalten.

Die aktuelle Arbeitsdatei kann mit @XWRITE in eine POSIX-Datei zurück geschrieben werden, die mit @XOPEN geöffnet ist. Nach dem Schreiben bleibt die Datei geöffnet und die Arbeitsdatei bleibt erhalten.

6.9 Dateikataloge

Mit der Anweisung @SHOW (Format 1) kann man sich Listen von Dateien aus dem BS2000-Katalog, aus einem POSIX-Verzeichnis oder aus einer Bibliothek ausgeben lassen.

Mit der Anweisung @DELETE (Format 3) kann man aus dem EDT heraus Dateien aus dem BS2000-Katalog, aus einem POSIX-Verzeichnis oder Elemente aus einer Bibliothek löschen.

Wenn sich die Anweisungen auf Bibliothekselemente beziehen, dann sind alle Elementtypen zugelassen.

BS2000-Dateikataloge können darüber hinaus auch mit den alten Anweisungen bearbeitet werden. Mit der Anweisung @FSTAT kann man sich Listen von BS2000-Dateien ausgeben lassen. Mit der Anweisung @UNSAVE kann man aus dem EDT heraus Dateien aus dem BS2000-Katalog löschen.

6.10 Systemdateien

Mit dem EDT kann man von SYSDTA lesen und nach SYSOUT und SYSLST schreiben.

Die Behandlung von Zeichensätzen beim Zugriff auf Systemdateien ist im Abschnitt „Zeichensätze“ auf Seite 48 beschrieben. Die Systemdateien SYSLST und SYSOUT sollten dabei nur dann Dateien oder Bibliothekselementen mit Unicode-Zeichensatz zugewiesen werden, wenn man sicher sein kann, dass nur der EDT Ausgaben in diese Dateien macht. Da andere Systemkomponenten den SYSLST bzw. SYSOUT zugeordneten Zeichensatz in der Regel nicht berücksichtigen, könnten andernfalls Dateien entstehen, die ungültige Zeichen enthalten.

6.10.1 Die Systemdatei SYSDTA

In folgenden Fällen wird im EDT von SYSDTA gelesen:

– Einlesen von Zeilen im L-Modus, falls @EDIT ONLY eingestellt ist, oder im Stapelbetrieb. Handelt es sich bei der Zeile um eine EDT-Anweisung, wird diese sofort ausgeführt, andernfalls wird die Zeile in der aktuellen Arbeitsdatei abgelegt.

– Einlesen mit der Anweisung @CREATE ... READ ohne Eingabeaufforderung, oder im Stapelbetrieb

Die von SYSDTA einzulesenden Sätze dürfen maximal 32763 Byte lang sein (RDATA erlaubt einen Eingabebereich von 32767 Bytes, die ersten 4 Bytes sind das Satzlängenfeld).

Bei der Initialisierung des EDT wird der Zeichensatz von SYSDTA ermittelt.
Dieser wird für das Lesen von SYSDTA verwendet. Ändert sich die Zuweisung zu SYSDTA, wird der Zeichensatz erneut ermittelt und danach mit dem neuen Zeichensatz gelesen.

6.10.2 Die Systemdatei SYSOUT

Im Dialogbetrieb werden folgende Ausgaben des EDT nach SYSOUT geschrieben:

1. Sowohl im F-Modus als auch im L-Modus werden die Ausgaben der Anweisungen @COMPARE (Format 1), @LIMITS, @ON COLUMN, @SEQUENCE CHECK, @TABS VALUES nach SYSOUT geschrieben.
2. Nur im L-Modus werden die Ausgaben der Anweisungen @COMPARE (Format 2), @FSTAT, @PROC (Format 2), @SHOW, @STAJV, @STATUS nach SYSOUT geschrieben.
3. Nur im L-Modus werden EDT-Fehlermeldungen nach SYSOUT geschrieben.
4. Die Logging-Ausgaben als Folge der Anweisungen @CHECK (nur L-Modus), @DO PRINT, @EDIT PRINT, @INPUT PRINT werden nach SYSOUT geschrieben.
   Hier werden Arbeitsdateisätze oder EDT-Anweisungen (zum Teil mit Zusatzinformationen) ausgegeben.
5. Sowohl im F-Modus als auch im L-Modus werden die Ausgaben der Anweisungen @ON PRINT und @PRINT (ohne Operand V) nach SYSOUT geschrieben. Hier werden Arbeitsdateisätze zusammen mit Zeilennummern ausgegeben.

Im Stapelbetrieb werden nur die Meldungen zur normalen oder abnormalen Beendigung (z.B. EDT8000) nach SYSOUT geschrieben, sofern nicht der Auftragsschalter 8 gesetzt ist.

Tritt beim Schreiben nach SYSOUT ein nicht behebbarer Fehler auf, so wird der EDT beendet.

Das erste Zeichen jedes Satzes bei der Ausgabe nach SYSOUT ist ein Vorschubsteuerzeichen. Ist SYSOUT der Datensichtstation zugeordnet, wird es nicht mit angezeigt. Ist SYSOUT einer Datei zugeordnet, wird es Bestandteil der Datei. Durch das System wird bei Zuweisung einer nicht existierenden Datei im Katalog gekennzeichnet, dass die Datei EBCDIC-Steuerzeichen enthält. Der EDT wertet den Katalog-Eintrag aber nicht aus, sondern erzeugt immer EBCDIC-Vorschubsteuerzeichen bzw. die diesen EBCDIC-Zeichen entsprechenden Steuerzeichen des Zeichensatzes, der SYSOUT zugeordnet ist.
Wenn der Anwender die Datei ausdrucken will, können diese Vorschubsteuerzeichen ausgewertet werden. Bei der Ausgabe nach SYSOUT werden die gleichen Vorschubsteuerzeichen benutzt wie bei SYSLST (siehe Abschnitt „Die Systemdatei SYSLST“ auf Seite 159).

Die Ausgaben nach SYSOUT sind in der Länge beschränkt (2032 Byte inklusive Satzlängenfeld und Vorschubsteuerzeichen bei Ausgabe in Dateien und 32767 Byte bei Ausgabe auf das Terminal).

Hat eine Ausgabe eine größere Länge, dann wird der auszugebende Satz in Stücke von max. 2027 Byte bei Dateiausgabe bzw. 32762 Byte bei Terminalausgabe aufgeteilt und in mehreren Teilen ausgegeben.
Dabei wird, falls SYSOUT in einem Unicode-Zeichensatz vorliegt, sichergestellt, dass der Umbruch immer an einer Zeichengrenze stattfindet.

Die Ausgaben nach SYSOUT erfolgen durch den EDT im zugeordneten Zeichensatz, der mit dem BS2000-Makro GCCSN ermittelt wird, wenn es sich nicht um die Datensichtstation handelt. Ausgaben auf die Datensichtstation erfolgen immer im eingestellten Kommunikations-Zeichensatz. Ändert sich die Zuweisung zu SYSOUT, wird der Zeichensatz erneut ermittelt und danach mit dem neuen Zeichensatz geschrieben. Ist dieser *NONE, wird EDF03IRV verwendet. Falls die Ausgabe Zeichen enthält, die im Ziel-Zeichensatz nicht darstellbar sind, wird das mit @PAR SUBSTITUTION-CHARACTER definierte Ersatzzeichen eingesetzt. Ist kein solches definiert, wird ein Leerzeichen eingesetzt.

Bei der Zuordnung eines Zeichensatzes zu SYSOUT sollte man aber immer beachten, wer alles Ausgaben dorthin erzeugt, da noch nicht alle System-Komponenten einen Zeichensatz von SYSOUT korrekt berücksichtigen.
So werden z.B. Ausgaben des EDT im Dialogbetrieb, die mit WRTRD erzeugt werden im Kommunikationszeichensatz zum Terminal gesendet.
Ist in diesem Fall SYSOUT einer Datei zugewiesen, werden vom System diese im Kommunikationszeichensatz (z.B. UTFE) vorliegenden Daten und die darauf erfolgten Benutzereingaben (ebenfalls in diesem Zeichensatz) ohne jede Berücksichtigung des Zeichensatzes der Datei (auf die SYSOUT gelegt wurde) zusätzlich in diese Datei geschrieben, was problematisch ist, wenn diese einen anderen Zeichensatz hat.
Es muss daher zurzeit empfohlen werden, als Zeichensatz für SYSOUT nur EBCDIC-Zeichensätze zu verwenden und im Dialogbetrieb möglichst SYSOUT nicht in eine Datei umzulenken.

Bei eingeschalteter Bildschirmkontrolle und wenn SYSOUT einem Terminal zugeordnet ist, kann eine Ausgabe mit K2 unterbrochen werden. Wird der EDT-Lauf mit /RESUME-PROGRAM oder /INFORM-PROGRAM fortgesetzt, so wird die Ausgabe abgebrochen und ggf. eine oder mehrere Meldungen ausgegeben.

6.10.3 Die Systemdatei SYSLST

Sowohl im Dialog- als auch im Stapelbetrieb werden folgende Ausgaben des EDT nach SYSLST geschrieben:

1. Die Ausgabe der Anweisungen @LIST (ohne Operand I) und @PAGE wird nach SYSLST geschrieben. Hier werden Arbeitsdateisätze zusammen mit Zeilennummern ausgegeben.
2. Die Logging-Ausgaben als Folge der Anweisung @LOG werden nach SYSLST geschrieben. Hier werden Arbeitsdateisätze oder EDT-Anweisungen ausgegeben.

Im Stapelbetrieb werden zusätzlich folgende Ausgaben des EDT nach SYSLST geschrieben, sofern nicht der Auftragsschalter 8 gesetzt ist:

3. Die Ausgaben der Anweisungen @COMPARE, @FSTAT, @LIMITS, @ON COLUMN, @PROC (Format 2), @SEQUENCE CHECK, @SHOW, @STAJV, @STATUS, @TABS VALUES werden nach SYSLST geschrieben.
4. EDT-Fehlermeldungen werden nach SYSLST geschrieben.
5. Die Logging-Ausgaben als Folge der Anweisungen @CHECK, @DO PRINT, @EDIT PRINT, @INPUT PRINT werden nach SYSLST geschrieben. Hier werden Arbeitsdateisätze oder EDT-Anweisungen (zum Teil mit Zusatzinformationen) ausgegeben.
6. Die Ausgaben der Anweisungen @ON PRINT und @PRINT werden nach SYSLST geschrieben. Hier werden Arbeitsdateisätze zusammen mit Zeilennummern ausgegeben.

Ist Auftragsschalter 8 gesetzt, werden diese Ausgaben im Stapelbetrieb nach SYSOUT ausgegeben. Dies ist in der Anweisungsbeschreibung nicht explizit formuliert, gilt aber, auch wenn nur *'wird im Stapelbetrieb nach* SYSLST *ausgegeben'* beschrieben ist.

Wenn im Stapelbetrieb nicht nach SYSLST geschrieben werden kann, so wird die Ausgabe abgebrochen und die Fehlermeldung EDT5498 nach SYSOUT ausgegeben.

Das erste Zeichen jedes Satzes bei der Ausgabe nach SYSLST ist ein Vorschubsteuerzeichen. Durch das System wird bei Zuweisung einer nicht existierenden Datei im Katalog gekennzeichnet, dass die Datei EBCDIC-Steuerzeichen enthält. Der EDT wertet den Katalog-Eintrag aber nicht aus, sondern erzeugt immer EBCDIC-Vorschubsteuerzeichen bzw. die diesen EBCDIC-Zeichen entsprechenden Steuerzeichen des Zeichensatzes, der SYSLST zugeordnet ist. Beim Drucken von SYSLST während der Taskbeendigung werden sie ausgewertet. Beim Drucken einer SYSLST zugeordneten Datei kann dies vom Anwender veranlasst werden.
Hat SYSLST einen Unicode-Zeichensatz, so werden die Steuerzeichen entsprechend umcodiert. Die Datei kann also im EDT bearbeitet werden. Beim Drucken der Datei werden die Steuerzeichen vom BS2000-Subsystem SPOOL wieder zurückgewandelt.

Die vom EDT generierten Vorschubsteuerzeichen sind in der Beschreibung der Anweisung @LIST ausführlich beschrieben, ebenso wie die Behandlung von Unicode-Dateien mit Vorschubsteuerzeichen beim Drucken.

Etliche Anweisungen erzeugen am Beginn ihrer Ausgabe bzw. bei Beginn eines neuen Abschnitts zusätzliche Zeilenvorschübe.
Dies unterbleibt generell, wenn die Ausgabe an einem Seitenanfang erfolgt. Die Einstellung der Seitengröße für SYSLST erfolgt mit den Anweisungen @PAGE und @LIST und wirkt für alle Ausgaben nach SYSLST.
Wird die Zuordnung von SYSLST während des EDT-Laufs geändert, geht der EDT davon aus, dass dies an einem Seitenanfang geschieht und startet seine Zeilenzählung entsprechend neu.

Die Ausgaben nach SYSLST erfolgen normalerweise mit einer maximalen Länge von 132 Zeichen (plus Vorschubsteuerzeichen). Ist der Auftragsschalter 6 gesetzt, geschieht dies mit einer maximalen Länge von 160 Zeichen.
Hat eine Ausgabe eine größere Länge, dann wird sie entsprechend umgebrochen und in mehreren Teilen ausgegeben. Der Umbruch erfolgt immer an Zeichengrenze.

Die Ausgaben nach SYSLST erfolgen im zugeordneten Zeichensatz, der mit dem BS2000-Makro GCCSN ermittelt wird. Ändert sich die Zuweisung zu SYSLST, wird der Zeichensatz erneut ermittelt und danach mit dem neuen Zeichensatz geschrieben. Ist dieser *NONE, wird EDF041 verwendet. Falls die Ausgabe Zeichen enthält, die im Ziel-Zeichensatz nicht darstellbar sind, wird das mit @PAR SUBSTITUTION-CHARACTER definierte Ersatzzeichen eingesetzt. Ist kein solches definiert, wird ein Leerzeichen eingesetzt.

6.10.4 Die Systemdateien SYSLST01 .. SYSLST99

Die Logging-Ausgaben des EDT als Folge der Anweisung @LOG können sowohl im Dialog- als auch im Stapelbetrieb nach SYSLST01 bis SYSLST99 geschrieben werden.

In diese Systemdateien kann nur ausgegeben werden, wenn sie mit einer Datei, einem Bibliothekselement oder einer S-Variablen verknüpft werden.

Das erste Zeichen jedes Satzes bei der Ausgabe nach SYSLST01 bis SYSLST99 ist ein Vorschubsteuerzeichen.
Durch das System wird bei Zuweisung einer nicht existierenden Datei im Katalog gekennzeichnet, dass die Datei EBCDIC-Steuerzeichen enthält. Der EDT wertet den Katalog-Eintrag aber nicht aus, sondern erzeugt immer EBCDIC-Vorschubsteuerzeichen bzw. die diesen EBCDIC-Zeichen entsprechenden Steuerzeichen des Zeichensatzes, der der jeweiligen SYSLSTnn-Datei zugeordnet ist. Wenn der Anwender die Datei ausdrucken will, können diese Vorschubsteuerzeichen ausgewertet werden.
Bei der Ausgabe nach SYSLST01 bis SYSLST99 werden vom EDT nur wenige Vorschubsteuerzeichen benutzt, insbesondere erfolgt keine Überwachung der Seitengröße wie bei SYSLST.

Die Ausgaben nach SYSLST01 bis SYSLST99 erfolgen normalerweise mit einer maximalen Länge von 132 Zeichen (plus Vorschubsteuerzeichen). Ist der Auftragsschalter 6 gesetzt, geschieht dies mit einer maximalen Länge von 160 Zeichen. Hat eine Ausgabe eine größere Länge, dann wird sie entsprechend umgebrochen und in mehreren Teilen ausgegeben. Der Umbruch erfolgt immer an Zeichengrenze.

Die Ausgaben nach SYSLST01 bis SYSLST99 erfolgen im zugeordneten Zeichensatz, der mit dem BS2000-Makro GCCSN ermittelt wird. Ändert sich die Zuweisung, wird der Zeichensatz erneut ermittelt und danach mit dem neuen Zeichensatz geschrieben. Ist dieser *NONE, wird EDF03IRV verwendet. Falls die Ausgabe Zeichen enthält, die im Ziel-Zeichensatz nicht darstellbar sind, wird das mit @PAR SUBSTITUTION-CHARACTER definierte Ersatzzeichen eingesetzt. Ist kein solches definiert, wird ein Leerzeichen eingesetzt.

7 Anweisungsbeschreibungen

In diesem Abschnitt werden die in den detaillierten Beschreibungen der Anweisungen und Kurzanweisungen des EDT verwendeten Darstellungsmittel, die grundsätzliche Struktur der Beschreibungen sowie die Operandentypen der verschiedenen Anweisungen erläutert.

7.1 Metasyntax

Für die formale Darstellung der Anweisungen werden folgende Metasyntax und typografische Darstellungsmittel verwendet.

Formale Darstellung	**Erläuterung**	**Beispiele**
GROSSBUCHSTABEN **und** Sonderzeichen	Großbuchstaben und Sonderzeichen bezeichnen Konstanten oder Schlüsselwörter, die der Benutzer in dieser Form eingeben muss.	UPDATE, OVERWRITE
GROSSBUCHSTABEN in Halbfett	Halbfette Großbuchstaben kennzeichnen die Kurzform der Schlüsselwörter. Zulässig sind alle Eingaben zwischen Kurzform und Langform.	**@LOW**ER Einzugeben ist: @LOW, @LOWE oder @LOWER
kleinbuchstaben	Kleinbuchstaben bezeichnen variable Operanden, die der Benutzer bei der Eingabe durch aktuelle Werte ersetzen muss.	@GOTO line Einzugeben ist z.B.: @GOTO 3

{ }	Geschweifte Klammern schließen Alternativen ein, d.h. eine der Angaben muss ausgewählt werden.	**@LOW**ER { **ON** / **OF**F } Einzugeben ist: @LOWER ON oder @LOWER OFF
|	| trennt Alternativen, wenn diese nicht übereinander sondern nebeneinander stehen.	**@LOW**ER {**ON** | **OF**F}
. .	**. .** bezeichnet Alternativen, die nicht einzelnen aufgeführt sind, sondern aus einem fortlaufenden Bereich auszuwählen sind.	1**. .**22 Einzugeben ist eine Zahl zwischen 1 und 22. $1**. .** $22 Einzugeben ist ein Symbol zwischen `$1` und `$22`.
[ ]	Angaben in eckigen Klammern sind optional; sie können wahlweise angegeben werden.	
[,...]	Diese Konstruktion mit 3 Punkten bedeutet eine mögliche Wiederholung der davor stehenden syntaktischen Einheit, wobei als Trennzeichen zwischen den Wiederholungen ein Komma anzugeben ist.	`line` **[,...]** Einzugeben ist z.B.: `1, 3, 7` oder `10`.
<u>Unterstreichung</u>	Wert innerhalb von weglassbaren Alternativen, der bei Weglassung vom EDT benutzt wird. Ist in einem solchen Fall keine der Alternativen als Standardwert gekennzeichnet, ist der ausführlichen Operandenbeschreibung zu entnehmen, was dann gilt.	**@LOW**ER [<u>**ON**</u> / **OF**F] Die Eingaben @LOWER und @LOWER ON sind gleichwertig.

7.2 Anweisungssyntax

In diesem Abschnitt werden die grundlegenden Konzepte bei der Syntaxanalyse von EDT-Anweisungen erläutert.

Wegen der unterschiedlichen Behandlung muss der EDT bei jeder Eingabe zuerst entscheiden, ob es sich um eine Dateneingabe oder um eine EDT-Anweisung handelt (es können auch mehrere EDT-Anweisungen sein, im F-Modus durch Semikolons (;), im L-Modus bei eingeschaltetem BLOCK-Modus durch LZE-Zeichen voneinander getrennt).

Im F-Modus trifft der EDT diese Entscheidung an Hand des Eingabeorts. Eingaben in der Anweisungszeile interpretiert der EDT grundsätzlich als EDT-Anweisung, Eingaben im Datenfenster dagegen als Dateneingaben und Eingaben in der Kurzanweisungsspalte als Kurzanweisungen. Ähnliches gilt für die Unterprogramm-Schnittstelle, wo die Eingabe von EDT-Anweisungen bzw. Daten in jeweils eigens dafür vorgesehenen Eingabebereichen zu erfolgen hat.

Im L-Modus dagegen erfolgen alle Eingaben in einer Zeile. Um im L-Modus Dateneingaben von EDT-Anweisungen unterscheiden zu können, müssen bei der Eingabe im L-Modus alle EDT-Anweisungen mit dem EDT-Anweisungssymbol (standardmäßig @) beginnen.
Im F-Modus und an der Unterprogramm-Schnittstelle darf das EDT-Anweisungssymbol auch weggelassen werden, da dort keine Verwechslungsgefahr mit Dateneingaben besteht.

Im L-Modus ist noch folgende Besonderheit zu beachten.
Beginnt eine Eingabe mit zwei EDT-Anweisungssymbolen (@@, wobei jeweils vor und nach dem ersten @ ein oder mehrere Leerzeichen stehen dürfen), interpretiert der EDT diese Eingabe als Dateneingabe, wobei das zweite EDT-Anweisungssymbol als erstes Zeichen der Dateneingabe gilt. Alle Zeichen (erstes EDT-Anweisungssymbol und alle Leerzeichen), die vor dem zweiten EDT-Anweisungssymbol stehen, schneidet der EDT ab. Damit können im L-Modus auf einfache Weise Zeilen mit EDT-Anweisungen in EDT-Prozeduren geschrieben werden (wenn eine Eingabe nur mit einem EDT-Anweisungssymbol beginnt, wird sie sofort als EDT-Anweisung ausgeführt und nicht in eine Zeile geschrieben). Im L-Modus wird eine Dateneingabe also nur dann als Anweisung interpretiert, wenn das erste von einem Leerzeichen verschiedene Zeichen das EDT-Anweisungssymbol ist und das erste von einem Leerzeichen verschiedene Zeichen, das dem EDT-Anweisungssymbol folgt, kein EDT-Anweisungssymbol ist.

Das eben Gesagte gilt auch in gleicher Weise für Benutzeranweisungssymbole. Beginnt eine Eingabe mit zwei **gleichen** Benutzeranweisungssymbolen, so wird diese Eingabe als Dateneingabe interpretiert, wobei das zweite Benutzeranweisungssymbol als erstes Zeichen der Dateneingabe gilt. Beginnt dagegen die Eingabe mit einem Benutzeranweisungssymbol oder mit zwei verschiedenen Benutzeranweisungssymbolen, so wird alles nach dem ersten Benutzeranweisungssymbol (Leerzeichen werden überlesen) als Benutzeranweisung interpretiert und ausgeführt.

Einige EDT-Anweisungen (z.B. @SET, Format 6) haben den Operanden `text` (siehe Abschnitt „Operandensyntax“ auf Seite 173). Dieser Operand `text`, der aus Sicht des EDT wie eine eigenständige Eingabe behandelt wird, kann nun selbst wieder entweder eine EDT-Anweisung oder eine Dateneingabe sein. Welche der beiden Möglichkeiten zutrifft, entscheidet der EDT nach den für den L-Modus gültigen Regeln.

Hat der EDT eine Eingabe als EDT-Anweisung erkannt, so wird zunächst für den Fall, dass die Eingabe aus mehreren EDT-Anweisungen besteht, die jeweils erste, noch nicht bearbeitete EDT-Anweisung der Eingabe separiert. Im F-Modus geschieht dies an Hand des Semikolons, wobei Semikolons in Literalen bei der Zerlegung nicht berücksichtigt werden, bzw. im L-Modus an Hand des LZE-Zeichens. Die (separierte) EDT-Anweisung wird dann in zwei interne Puffer kopiert. Einer der beiden Puffer enthält dann die EDT-Anweisung so, wie sie eingegeben wurde, während der andere Puffer zur Vereinfachung der Erkennung von Anweisungsnamen und Operanden in Großbuchstaben konvertiert wird.

Als nächstes versucht der EDT den Anweisungsnamen zu ermitteln. Konnte die Erkennung des Anweisungsnamens erfolgreich durchgeführt werden und handelt es sich um eine EDT-Anweisung mit indirekter Operandenangabe (siehe Abschnitt „Indirekte Angabe von Operanden“ auf Seite 169), so werden die Operanden nun in den beiden internen Puffern eingesetzt, wobei in einem der beiden Puffer in diesem Fall noch einmal eine Konvertierung in Großbuchstaben erfolgt.

Nun folgt die syntaktische Prüfung der EDT-Anweisung.
Bei der Analyse von EDT-Anweisungen wird bei denjenigen Teilen einer EDT-Anweisung, bei denen die Unterscheidung zwischen Groß- und Kleinschreibung relevant ist, z.B. Literale, auf die ursprünglich eingegebene EDT-Anweisung zurückgegriffen.
Eine Interpretation von Unicode-Ersatzdarstellungen findet bei EDT-Anweisungen nur innerhalb von Literalen (außer bei @DO und @PARAMS) statt.
Außerdem wird innerhalb von EDT-Anweisungen keine Tabulatorexpansion durchgeführt.
Treten keine Syntaxfehler auf, so wird die EDT-Anweisung anschließend ausgeführt und danach wird die ursprünglich eingegebene EDT-Anweisung in den Anweisungspuffer eingetragen (indirekte Operanden sind dort nicht aufgelöst).

Eine EDT-Anweisung beginnt mit einem Anweisungsnamen (z.B. @OPEN, @COPY, @WRITE), dem ein oder mehrere Operanden folgen können. Bei einigen EDT-Anweisungen ist nach den Operanden auch noch ein Kommentar erlaubt. Vor dem EDT-Anweisungssymbol sowie zwischen dem EDT-Anweisungssymbol und dem Anweisungsnamen sind ein oder mehrere Leerzeichen zulässig (aber nicht nötig).

Hat eine EDT-Anweisung Operanden, so folgen diese, evtl. durch ein oder mehrere Leerzeichen getrennt, dem Anweisungsnamen. Die Operanden sind in der vorgegebenen Reihenfolge anzugeben. Vor bzw. nach jedem Operanden können beliebig viele Leerzeichen eingegeben werden. Es gibt Operanden, die immer angegeben werden müssen, andere Operanden sind aber auch optional.

Werden optionale Operanden weggelassen, so werden für diese Operanden Standardwerte angenommen. Welche Operanden optional sind und welche nicht und welches die Standardwerte für weggelassene optionale Operanden sind, ist der Syntaxbeschreibung der jeweiligen Anweisung zu entnehmen.

Die Leerzeichen zwischen Anweisungsnamen und Operanden bzw. zwischen den einzelnen Operanden können auch entfallen. Sie müssen jedoch dann angegeben werden, wenn Anweisungsname und Operand bzw. zwei aufeinander folgende Operanden sonst nicht unterscheidbar sind.

Beispiel

@SYMBOLS='?' ist falsch; richtig ist @SYMBOL S='?', da @SYMBOL eine zulässige Abkürzung der Anweisung @SYMBOLS ist.

Das Weglassen von Leerzeichen zwischen Anweisungsnamen und Operanden bzw. zwischen den einzelnen Operanden wird generell nicht empfohlen, da die Lesbarkeit einer Anweisung unter Umständen stark beeinträchtigt wird.

Neben der soeben beschriebenen direkten Angabe von Operanden kann man diese auch indirekt über eine Zeichenfolgevariable angeben. Damit hat man z.B. die Möglichkeit, Operanden erst zur Laufzeit festzulegen, was eine wesentlich größere Flexibilität insbesondere in EDT-Prozeduren erlaubt. Diese Möglichkeit der Operandenangabe ist genauer im Abschnitt „Indirekte Angabe von Operanden“ auf Seite 169 beschrieben.

Bei einigen Anweisungen ist nach den Operanden, sofern vorhanden, auch noch ein Kommentar erlaubt. Ob bei einer Anweisung ein Kommentar zulässig ist, kann der Syntaxbeschreibung der jeweiligen Anweisung entnommen werden.

Die meisten EDT-Anweisungen können abgekürzt werden. Die Abkürzung entsteht in der Regel durch Weglassen von einem oder mehreren Zeichen am Ende des Anweisungsnamens. In einigen Fällen gibt es aber auch Abkürzungen, die nicht durch Weglassen von Zeichen entstehen. Die Anweisung @BLOCK kann z.B. auch mit @BK, die Anweisung @QUOTE auch mit @QE und die Anweisung @SETF kann im F-Modus auch mit dem Zeichen # abgekürzt werden. Bei der Anweisung @SEARCH-OPTIONS können auch einige Zeichen in der Mitte des Anweisungsnamens weggelassen werden, so dass z.B. @SEA aber auch @SEA-OPTIONS gültige Abkürzungen für diese Anweisung sind. Eine Ausnahme ist noch die Anweisung @SET, bei der der Anweisungsname sogar vollständig entfallen kann. Bei @SET (Format 6) muss dann allerdings auch im F-Modus und an der Unterprogramm-Schnittstelle das Anweisungssymbol angegeben werden, um Eindeutigkeit herzustellen. Der Teil des Anweisungsnamens, der mindestens vorhanden sein muss, damit die Anweisung vom EDT eindeutig erkannt werden kann, ist in den Syntaxdiagrammen jeweils durch Fettdruck hervorgehoben.

Ausgehend von den kürzest möglichen Abkürzungen versucht der EDT, den Anweisungsnamen eindeutig zu identifizieren. Bei Erfolg werden eventuell noch vorhandene weitere Zeichen des Anweisungsnamens überlesen. Dies wird solange fortgesetzt, bis entweder

der vollständige Anweisungsname abgearbeitet ist oder ein Zeichen erkannt wird, welches nicht mehr zum Anweisungsnamen passt (dies kann auch ein Leerzeichen sein). Das erste von einem Leerzeichen verschiedene Zeichen, das nicht mehr zum Anweisungsnamen passt, wird, sofern vorhanden, als erstes Zeichen des Operandenteils interpretiert.

Bei drei Paaren von Anweisungen (@DELETE/@DELIMIT, @PAR/@PARAMS und @UNSAVE/@UPDATE) reicht die Analyse des Anweisungsnamens nicht aus, um eine Anweisung eindeutig zu identifizieren, da die kürzesten möglichen Abkürzungen jeweils identisch sind (@D, @PAR bzw. @U). In diesen drei Fällen wird das erste Zeichen des Operandenteils zur Unterscheidung herangezogen. Bei der Anweisung @DELIMIT beginnt z.B. der Operandenteil mit dem Zeichen =, in der Anweisung @DELETE dagegen kommt dieses Zeichen im Operandenteil nicht vor (bei der Anweisung @PARAMS ist das Zeichen `&` das erste Zeichen des Operandenteils und bei der Anweisung @UNSAVE das Zeichen `'`, das bei der jeweils anderen Anweisung nicht im Operandenteil vorkommt).

Beispiel

Wir betrachten die Eingabe @DEL& (Leerzeichen zwischen dem Anweisungsnamen und dem Operandenteil können ja weggelassen werden). Die Abkürzungen @DIALOG, @DO und @DR (für @DROP) passen nicht. Es bleibt nur noch die Abkürzung @D übrig, die aber in gleicher Weise für die Anweisung @DELETE oder die Anweisung @DELIMIT stehen kann. Die Anweisung @DELIMIT scheidet aber aus, da in der restlichen Anweisung das Zeichen = nicht vorkommt. Damit steht @DELETE als Anweisungsname fest. Nun werden in der Eingabe noch die Zeichen `E` und `L` überlesen, da sie mit den entsprechenden Zeichen im Anweisungsnamen übereinstimmen. Das Zeichen `&` stimmt dagegen nicht mehr mit dem entsprechenden Zeichen `(E)` im Anweisungsnamen überein und ist deshalb das erste (und in diesem Fall auch das einzige) Zeichen des Operandenteils.

Auf Grund der soeben beschriebenen Vorgehensweise bei der Anweisungsanalyse kann auch erklärt werden, warum manchmal nicht immer gleich nachvollziehbare Fehlermeldungen ausgegeben werden.
Wenn in einer Eingabe keine Anweisung identifiziert werden kann, wird die Meldung `EDT3101` (Unzulässige Anweisung) ausgegeben (z.B. @XD). Es gibt jedoch viele Situationen, wo man eigentlich die Ausgabe der Meldung `EDT3101` erwarten würde, wo aber stattdessen die Meldung `EDT3002` (Operanden-Fehler) ausgegeben wird. Angenommen, man hätte in dem vorangegangenen Beispiel statt @DEL& versehentlich @DDL& eingegeben. Genau wie vorher wäre die Anweisungsanalyse zu dem Ergebnis gekommen, dass es sich um eine @DELETE-Anweisung handelt (@DELIMIT scheidet ja nach wie vor wegen des fehlenden Zeichens = aus). Aber bereits das nächste Zeichen in der Eingabe nach @D unterscheidet sich von dem entsprechenden Zeichen im Anweisungsnamen @DELETE. Also wird alles ab diesem Zeichen als zum Operandenteil gehörig betrachtet `(DL&)`. Die Zeichenfolge `DL&` ist jedoch für keines der drei Formate der @DELETE-Anweisung als ein zulässiger Operand interpretierbar, weshalb folgerichtig die Meldung `EDT3002` ausgegeben wird.

Hinweise

– Die Begrenzersymbole für Literale (standardmäßig die Zeichen ' und ") können mit der Anweisung @QUOTE umdefiniert werden. Bei den Anweisungen @DO und @PARAMS wird jedoch, ungeachtet einer eventuell mit der Anweisung @QUOTE vorgenommenen Umdefinition, als Begrenzersymbol für Literale immer das Zeichen ' verwendet.

– Innerhalb von @DO-Prozeduren erfolgt die Anweisungsanalyse erst nach der Ersetzung der Prozedurparameter. Erst dann wird auch eine eventuelle indirekte Operandenangabe aufgelöst.

– Es ist nicht erlaubt, in den Schlüsselwörtern von Anweisungen Leerzeichen einzustreuen oder am Ende von Anweisungen Kommentare anzuhängen (Kommentare sind am Ende einer Anweisung nur erlaubt, wenn dies in der Syntaxbeschreibung explizit angegeben ist).

– Mit der Anweisung @SYNTAX TESTMODUS=ON wird der Test-Modus aktiviert. EDT-Anweisungen werden in diesem Fall im L-Modus bis auf einige Ausnahmen nicht mehr ausgeführt, sondern nur noch syntaktisch geprüft. Damit lassen sich z.B. EDT-Prozeduren vor deren Einsatz hinsichtlich ihrer Ablauffähigkeit prüfen (genaueres siehe Anweisung @SYNTAX, Seite 552).

7.2.1 Indirekte Angabe von Operanden

Bei der indirekten Operandenangabe werden alle Operanden, die man bei der betreffenden EDT-Anweisung angeben möchte, vor der Ausführung der EDT-Anweisung in einer Zeichenfolgevariablen (#S00..#S20) abgelegt. In der EDT-Anweisung selbst wird dann die indirekte Operandenangabe durch das Zeichen & eingeleitet, welches dem Anweisungsnamen, getrennt durch ein oder mehrere Leerzeichen, folgt. Direkt nach dem Zeichen & muss dann der Name der Zeichenfolgevariablen folgen, welche die Operanden der EDT-Anweisung enthält. Weitere Zeichen (außer Leerzeichen) darf die Anweisung nicht enthalten.

Nach Erkennen des Anweisungsnamens wird der Anweisungsrest (Zeichen & sowie der nachfolgende Name der Zeichenfolgevariablen) 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. Bei Ausgabe der verursachenden Zeile vor einer Fehlermeldung, wird nur die Originaleingabe ausgegeben.

Wenn die Länge von Anweisungsname und der Ersetzung der Zeichenfolgevariablen größer als 32768 ist, wird die Abarbeitung der Anweisung mit der Fehlermeldung EDT5485 abgewiesen.

Bei den Anweisungen @+, @-, @IF (Format 1) und @SET (Format 6) kann der Operand `text` selbst wieder eine EDT-Anweisung sein. Auch bei diesen, im Operanden `text` angegebenen EDT-Anweisungen funktioniert die indirekte Operandenangabe.

Bei den Anweisungen @: (Umdefinieren des EDT-Anweisungssymbols), @+, @- und bei Wertzuweisungen zu EDT-Variablen mit der @SET-Anweisung, wobei der Anweisungsname @SET weggelassen wird, ist keine indirekte Operandenangabe erlaubt. Dies gilt auch für die Anweisung @PARAMS, da bei dieser Anweisung alle Operanden mit dem Zeichen `&` beginnen.

Achtung

Bei den Anweisungen @IF und @SET (Format 6) sind bei einer indirekten Operandenangabe Endlosschleifen im EDT möglich, wenn der Operand `text` in diesen Anweisungen selbst wieder eine der Anweisungen @IF oder @SET (Format 6) ist, z.B.:

```
1.     @SET #S1='1: @SET &#S1' ------------------------------------ (1)
2.     @SET &#S1 -------------------------------------------------- (2)
```

oder:

```
1.     @SET #S1='1: @IF &#S2' ------------------------------------- (1)
2.     @SET #S2='NO ERRORS: @SET &#S1' ---------------------------- (1)
3.     @SET &#S1 -------------------------------------------------- (2)
```

(1) Die Zeichenfolgevariablen `#S1` bzw. `#S2` werden für eine indirekte Operandenangabe mit einem geeigneten Inhalt gefüllt.

(2) Nach Eingabe dieser Anweisung mit indirekten Operanden gerät der EDT in eine Endlosschleife.

Unter Verwendung von noch mehr verschiedenen Zeichenfolgevariablen lassen sich nahezu beliebig komplexe Anweisungsfolgen konstruieren, die zu einer Endlosschleife im EDT führen.

7.3 Gliederung der Anweisungsbeschreibungen

Die ausführlichen Anweisungsbeschreibungen haben einen einheitlichen Aufbau:

1. Beschreibung der Funktion der Anweisung
2. Formale Anweisungssyntax
3. Ausführliche Operandenbeschreibung
4. Beschreibung von Besonderheiten und Einschränkungen der Anweisung sowie Anwendungshinweise
5. Beispiele

Jede Anweisungsbeschreibung beginnt mit einer allgemeinen Beschreibung der Funktion dieser Anweisung. Unmittelbar darauf folgt eine formale Beschreibung der Anweisungssyntax in folgender Form:

Operation	**Operanden**	**Modi**
Operation	Operanden	

Dabei steht im Feld *Operation* der Name der Anweisung in ausgeschriebener Form, wobei die maximal erlaubte Abkürzung des Anweisungsnamens durch halbfette Hervorhebung gekennzeichnet ist. Bei allen Anweisungen, die im L-Modus mit Anweisungssymbol einzuleiten sind bzw. die im F-Modus oder an der Unterprogrammschnittstelle mit Anweisungssymbol eingeleitet werden dürfen, ist das Standard-Anweisungssymbol @ mit angegeben. Bei Anweisungen die nicht mit dem Anweisungssymbol eingeleitet werden dürfen (einige F-Modus-Anweisungen), ist es auch nicht angegeben. Bei einigen wenigen Anweisungen sind zwei mögliche Anweisungsnamen angegeben. Diese können dann alternativ verwendet werden. Eventuelle weitere Besonderheiten, die bei der Verwendung des Anweisungsnamens oder seiner Abkürzungen zu beachten sind, werden im nachfolgenden Textteil der jeweiligen Anweisungsbeschreibung erläutert.

Im Feld *Operanden* werden die für die Anweisung möglichen Operanden formal syntaktisch beschrieben, wie sie bei direkter Operandenangabe in einer Anweisungszeile anzugeben wären. Dies entspricht auch der Syntax, die bei Ablage in einer Zeihenfolgevariablen einzuhalten ist, wenn man die Anweisung mit indirekter Operandenangabe aufrufen will. Auf die indirekte Operandeneingabe selbst wird in den Anweisungsbeschreibungen ansonsten nicht eingegangen (siehe Abschnitt „Anweisungssyntax“ auf Seite 165). Lediglich wenn diese für die jeweilige Anweisung verboten ist, wird darauf im nachfolgenden Textteil der Anweisung hingewiesen.

Die Benennung von variablen Operanden kennzeichnet gleichzeitig den Operandentyp und beschreibt damit die Syntax der Werte, die dafür eingesetzt werden können. Die Beschreibung aller Operandentypen finden Sie im Abschnitt „Operandensyntax“ auf Seite 173.

Im Feld *Modi* wird spezifiziert, in welchen Arbeitsmodi des EDT die Anweisung benutzt werden darf. Folgende Angaben sind möglich:

F-Modus — Die Anweisung darf im F-Modus benutzt werden. Enthält das Feld keine weiteren Einträge, darf die Anweisung nur im F-Modus verwendet werden. Auf weitere Besonderheiten und Einschränkungen für die Verwendung der Anweisung im F-Modus wird gegebenenfalls im nachfolgenden Textteil der Anweisung hingewiesen.

L-Modus — Die Anweisung darf im L-Modus und damit auch in Prozeduren verwendet werden. Enthält das Feld keine weiteren Einträge, darf sie nicht im F-Modus verwendet werden. Auf weitere Besonderheiten und Einschränkungen für die Verwendung der Anweisung im L-Modus wird gegebenenfalls im nachfolgenden Textteil der Anweisung hingewiesen.

@PROC — Die Anweisung ist ausschließlich in Prozeduren erlaubt. Ihre Verwendung im L-Modus außerhalb von Prozeduren oder im F-Modus ist nicht möglich.

Auf die formale Beschreibung der Anweisungssyntax folgt die ausführliche Beschreibung der einzelnen Operanden, normalerweise in der Reihenfolge ihres Auftretens. Dabei werden neben der Funktion des jeweiligen Operanden insbesondere semantische Besonderheiten der Operanden, Standardwerte beim Weglassen von Operanden und Wechselwirkungen zwischen Operanden beschrieben. Die Syntax von variablen Operanden wird dagegen nicht beschrieben, diese findet man im Abschnitt „Operandensyntax“ auf Seite 173.

Auf die Operandenbeschreibung folgt dann meist ein Textteil, in dem weitere Besonderheiten und Einschränkungen der jeweiligen Anweisung erläutert werden. Dieser Abschnitt enthält auch besondere Anwendungshinweise.

Die meisten Anweisungsbeschreibungen enden mit einem oder mehreren Beispielen, an denen nochmals Besonderheiten der Anweisung demonstriert werden. In allen Beispielen wird, wenn nichts anderes gesagt ist, von Standardeinstellungen des EDT ausgegangen.

7.4 Operandensyntax

Dieser Abschnitt enthält die genaue syntaktische Definition der verschiedenen variablen Operanden, die in den EDT-Anweisungen auftreten. Aus der Benennung eines Operanden kann man immer eindeutig auf seine syntaktische Definition schließen. Eventuelle semantische Besonderheiten und Einschränkungen im Kontext der jeweiligen Anweisung enthält die Operandenbeschreibung der ausführlichen Anweisungsbeschreibung.

Alle Operandentypen werden in den folgenden Abschnitten definiert. In den ausführlichen Anweisungsbeschreibungen werden diese Operandentypen als Operandennamen verwendet. Treten in einer Anweisungsbeschreibung syntaktisch äquivalente Operanden an verschiedenen Stellen auf, so werden sie durch anfügen einer fortlaufenden Nummer unterschieden. Die Definition erfolgt nur für den Basisnamen solcher Operanden.

Die folgende Beschreibung der Operandensyntax ist in thematisch zusammengehörende Abschnitte untergliedert. Innerhalb der Abschnitte sind die Beschreibungen so angeordnet, dass jeder Operandentyp nach Möglichkeit vor seiner erstmaligen Verwendung definiert wird. Zum gezielten Nachschlagen einzelner Definitionen dient die folgende alphabetisch geordnete Übersicht.

Operand	**Kurzbeschreibung**	**Seite**
binary	Binärziffer	176
char	Beliebiges Zeichen	177
char*	Beliebiges Zeichen oder Unicode-Ersatzdarstellung	177
chars	Zeichenfolge	182
chars*	Zeichenfolge mit Unicode-Ersatzdarstellung	182
col	Spaltennummer	190
cols	Spaltenbereich	190
cols*	Spaltenbereich relativ zum Satzende	190
comment	Beliebiger Kommentar	183
dd	Dezimalziffer	176
elname	Name eines Bibliothekselements	192
eltype	Typ eines Bibliothekselements	191
entry	Name einer Einsprungstelle oder einer CSECT	191
escseq	Unicode-Ersatzdarstellung	177
escsymb	Fluchtsymbol für Unicode-Ersatzdarstellung	177
file	Name einer DVS-Datei (in Hochkommas)	192

Operand	Kurzbeschreibung	Seite
formal	Formalparameter (bei @DO-Prozeduren)	194
fraction	Teil einer Zeilennummer (hinter dem Dezimalpunkt)	181
freetype	Freier Typname eines Bibliothekselements	191
hd	Hexadezimalziffer	176
hex	Folge von Hexadezimalziffern	181
hpos	Relative horizontale Positionieranweisung	195
inc	Schrittweite bei Zeilennummern	187
int	Ganzzahl	181
intex	Ganzzahliger Ausdruck	182
ivar	Ganzzahlvariablen	179
line	Direkt oder als Ausdruck angegebene Zeilennummer	188
lines	Zusammenhängender Bereich von Zeilennummern	189
linkname	Kettungsname für Dateien oder Jobvariablen	192
lnum	Direkt angegebene Zeilennummer	187
loopsymb	Schleifenzähler	178
lsym	Symbolisch angegebene Zeilennummer	187
lvar	Zeilennummervariablen	179
m	Satzmarkierung	195
message	Beliebiger Meldungstext	183
modlib	Bibliothek aus der Module nachgeladen werden	193
n	Vorzeichenlose Ganzzahl	181
name	Zeichenfolge von maximal acht Zeichen	183
op	Mathematischer Operator + oder -	178
param	Parameter bei @DO-Prozeduren	194
path	Pfadname einer DVS-Datei oder Jobvariablen	193
procnr	Name einer Arbeitsdatei	195
progname	Name eines Programms	192
rangesymb	Bereichssymbol	178
rel	Relation bei @IF-Anweisung	178
search	Suchbegriff bei @ON-Anweisung	186
spec	Sonderzeichen	176

Operand	Kurzbeschreibung	Seite
str	In Hochkommas eingeschlossene Folge von Zeichen	183
strchar	Einzelzeichen in Hochkommas	184
strspec	Einzelnes Sonderzeichen in Hochkommas	185
string	Direkt oder indirekt angegebene Zeichenfolge	185
svar	Zeichenfolgevariablen	179
svarex	Indirekte Angabe einer Zeichenfolgevariablen	180
svars	Zusammenhängender Bereich von Zeichenfolgevariablen	180
text	Folgeeingabe bei L-Modus-Anweisungen	186
unicode	UTF16-Code eines Zeichens (4 Hexadezimalziffern)	176
ver	Versionsnummer einer katalogisierten DVS-Datei	193
vers	Versionsnummer eines Bibliothekselements	193
vpos	Relative vertikale Positionieranweisung	195
vpos–op	Vertikaler Positionieroperand	195
xpath	Pfadname einer POSIX-Datei	194

7.4.1 Zeichen und Symbole

Dieser Abschnitt enthält die Definition elementarer Operandentypen bzw. von solchen, die nur für die Definition anderer Operandentypen benötigt werden und nicht selbst als realer Operand in einer der Anweisungen vorkommen.

Operand	Definition
binary	0 | 1

Die Ziffer `0` oder `1`.

Operand	Definition
dd	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Dezimalziffer.

Operand	Definition
hd	dd | A | B | C | D | E | F | a | b | c | d | e | f

Hexadezimalziffer.

Operand	Definition
spec	! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ‘ { | } ~

Ein Sonderzeichen aus dem oben angegebenen Zeichenvorrat (siehe auch Abschnitt „Zeichensatz einer Anweisung“ auf Seite 59).

Operand	Definition
unicode	hd hd hd hd

Folge von genau vier Hexadezimalziffern, die den `UTF16`-Code für ein Zeichen angeben.

Operand	Definition
escsymb	spec

Das aktuelle Fluchtsymbol, das eine Unicode-Ersatzdarstellung `escseq` einleitet. Es kann mit der Anweisung @PAR ESCAPE-CHARACTER definiert werden. Standardmäßig ist kein Fluchtsymbol zugewiesen.

Operand	Definition
escseq	escsymb U unicode

Ersatzdarstellung für ein Unicode-Zeichen. Die Folge von Hexadezimalziffern des Operandentyps unicode muss der `UTF16`-Codierung des Zeichens entsprechen. Falls man mit @PAR ESCAPE-CHARACTER='%' das Fluchtsymbol `%` definiert hat, wäre etwa `%U20AC` eine gültige Unicode-Sequenz, die dem Zeichen € entspricht.

Operand	Definition
char	beliebiges Zeichen

Ein beliebiges Zeichen.
Der Zeichenvorrat ist einerseits von dem verwendeten Zeichensatz abhängig, andererseits von der Eingabequelle. So lässt sich nicht jedes Zeichen direkt über die Tastatur eingeben, auch wenn die Datensichtstation dieses Zeichen darstellen kann. Für Zeichen, die nicht direkt eingegeben werden können, bietet der Operandentyp `char*` die Möglichkeit einer Unicode-Ersatzdarstellung.

Operand	Definition
char*	char | escseq

Ein beliebiges Zeichen, das direkt oder in seiner `UTF16`-Codierung in Form einer Unicode-Ersatzdarstellung angegeben wird (siehe auch Abschnitt „Unicode-Ersatzdarstellung“ auf Seite 53).

Operand	Definition
rangesymb	spec

Das aktuelle Bereichssymbol, das mit der Anweisung @RANGE verändert werden kann. Standardmäßig ist dies das Zeichen `&`.

Operand	Definition
loopsymb	spec

Der aktuelle Schleifenzähler, der in der Anweisung @DO definiert und innerhalb der aufgerufenen @DO-Prozedur wie eine Zeilennummervariable benutzt werden kann.

Operand	Definition
op	+ | -

Einer der mathematischen Operatoren + oder −.

Operand	Definition
rel	GT | LT | GE | LE | EQ | NE | > | < | >= | <= | = | <>

Zeichen für eine Relation, die über die @IF-Anweisung abgefragt werden kann.

Die Zeichen `GT` bzw. > (größer als), `LT` bzw. < (kleiner als), `GE` bzw. >= (größer oder gleich), `LE` bzw. <= (kleiner oder gleich), `EQ` bzw. = (gleich) und `NE` bzw. <> (ungleich) haben die übliche mathematische Bedeutung.

7.4.2 Variablen

Dieser Abschnitt enthält die Definition von Ganzzahlvariablen, Zeilennummervariablen und Zeichenfolgevariablen, sowie von Ausdrücken, die solche Variablen bestimmen. Die Variablen dürfen in vielen EDT-Anweisungen anstelle von explizit angegebenen Zahlen, Zeilennummern oder Zeichenfolgen angegeben werden.

Operand	Definition
ivar	#I00..#I20

Eine der Ganzzahlvariablen `#I00,#I01,...,#I20` (siehe Abschnitt „EDT-Variablen“ auf Seite 62). Führende Nullen im numerischen Teil der Variablenbezeichnung dürfen weggelassen werden. In Ganzzahlvariablen können positive und negative Ganzzahlen abgelegt werden. Der erlaubte Wertebereich liegt zwischen -2^{31} und $2^{31}-1$. Wird die Ganzzahlvariable anstelle einer explizit angegebenen Zahl in einer Anweisung verwendet, gelten je nach Anweisung unterschiedliche Grenzen.

Operand	Definition
lvar	#L00..#L20

Eine der Zeilennummervariablen `#L00,#L01,...,#L20` (siehe Abschnitt „EDT-Variablen“ auf Seite 62). Führende Nullen im numerischen Teil der Variablenbezeichnung dürfen weggelassen werden. Eine Zeilennummervariable darf Werte zwischen `0.0001` und `9999.9999` enthalten. Nach dem Starten des EDT sind alle Zeilennummervariablen mit dem ungültigen Wert `0.0` belegt. Demnach müssen den Zeilennummervariablen vor ihrer Benutzung zugelassene Werte zugewiesen werden.

Operand	Definition
svar	#S00..#S20

Eine der Zeichenfolgevariablen `#S00,#S01...,#S20` (siehe Abschnitt „EDT-Variablen“ auf Seite 62). Führende Nullen im numerischen Teil der Variablenbezeichnung dürfen weggelassen werden. Jeder Zeichenfolgevariablen ist ein Inhalt (die Zeichenfolge) und ein Zeichensatz zugeordnet. Löscht man eine Zeichenfolgevariable, so hat diese danach als Inhalt ein Leerzeichen und den Zeichensatz `EDF041`. Dies ist auch die Vorbelegung aller Zeichenfolgevariablen. Zeichenfolgevariablen können in vielen EDT-Anweisungen wie Zeilennummern einer Arbeitsdatei verwendet werden.

Operand	Definition
svarex	svar[op ivar] | svar[op nL]

Indirekte Angabe einer Zeichenfolgevariablen in Form eines Ausdrucks, der ihre Position relativ zu einer gegebenen Zeichenfolgevariablen beschreibt. Die relative Position der gewünschten Zeichenfolgevariablen kann dabei durch den Inhalt einer Ganzzahlvariablen oder explizit durch die Angabe nL bestimmt werden.

Beispiele

– Enthält #I10 den Wert 5, so bezeichnet #S0+#I10 die Variable #S5.

– Enthält #I3 den Wert 7, so bezeichnet #S10-#I3 die Variable #S3.

– Der Ausdruck #S15-5L bezeichnet die Variable #S10.

– Der Ausdruck #S3+8L bezeichnet die Variable #S11.

Operand	Definition
svars	svarex [[.] - [.] svarex]

Ein zusammenhängender Bereich von Zeichenfolgevariablen.

Hinweis

Die Angabe von Punkten vor und/oder nach dem Bereichs-Trenner sowie die Angabe von führenden Nullen vor einem Variablennamen werden aus Kompatibilitätsgründen in der bisherigen Form unterstützt, sind aber nicht mehr notwendig.

Die Angabe svarex1-svarex2 (z.B. #S1-#S10) bewikt das gleiche wie svarex2-svarex1 (z.B. #S10-#S1). Wird nur ein svarex Operand angegeben, so besteht der Bereich nur aus dieser einen Zeichenfolgevariablen.

Beispiele

– Mit #S3 wird die Zeichenfolgevariable #S3 ausgewählt.

– Mit #S4-#S7 werden #S4,#S5,#S6 und #S7 ausgewählt.

– Mit #S2+1L-#S6-#I3 werden #S3 und #S4 ausgewählt, falls #I3 den Inhalt 2 hat.

7.4.3 Zahlen

In diesem Abschnitt werden verschiedene Zahlenformate definiert, die in EDT-Anweisungen Verwendung finden. Wenn bestimmte Formate bereits eine Semantik für die angegebene Zahl festlegen, wird dies hier auch beschrieben.

Operand	**Definition**
n	dd | n dd

Vorzeichenlose Ganzzahl.
Die Anzahl der erlaubten Ziffern hängt von der jeweiligen Anweisung ab. Daher muss `00005` nicht unbedingt gleichwertig mit `5` sein.

Operand	**Definition**
fraction	.dd | fraction dd

Der hinter dem Dezimalpunkt stehende Teil einer Zeilennummer.
Erlaubt sind die Werte `.0001` bis `.9999`.

Operand	**Definition**
hex	hd | hex hd

Folge von Hexadezimalziffern.

Operand	**Definition**
int	n | op n | ivar

Ganzzahl, die entweder explizit oder über Ganzzahlvariablen angegeben werden kann, z.B.: `5,0,-23456` oder `#I0,#I1,...#I20`.
Im Gegensatz zum Operandentyp `n` ist bei `int` nur der numerische Wert von Bedeutung, führenden Nullen führen zu keiner Unterscheidung.
Bei expliziter Angabe liegt der erlaubte Wertebereich zwischen -2^{31} und $2^{31}-1$.

Operand	Definition
intex	int | op int | intex op int

Ganzzahliger Ausdruck.

7.4.4 Zeichenfolgen

In diesem Abschnitt werden Zeichenfolgen definiert, die in EDT-Anweisungen mit unterschiedlicher Semantik benutzt werden, etwa in Suchanweisungen, als Kommentar oder als Sonderzeichen.

Zeichenfolgen ohne definierten Begrenzer (z.B. Hochkomma) werden bei der Syntax-Analyse zunächst ohne Berücksichtigung ihrer semantischen Einschränkungen extrahiert. Dabei werden (wenn beim Operandentyp nicht anders beschrieben) alle Zeichen ab dem ersten Nichtleerzeichen bis zu einem intern definierten Begrenzerzeichen oder dem Anweisungs-Ende benutzt. Als Begrenzer verwendet der EDT normalerweise das Leerzeichen, das Komma, das Gleichheitszeichen und die runden Klammern. Erst nach dieser Extraktion des Operanden erfolgt die Prüfung auf Länge, Zeichenvorrat und erlaubte Syntax.

Operand	Definition
chars	char | chars char

Zeichenfolge.

Operand	Definition
chars*	char* | chars* char*

Zeichenfolge, in der neben Zeichen aus dem jeweiligen Zeichensatz auch Ersatzdarstellungen für Unicode-Zeichen vorkommen dürfen. Die Ersatzdarstellung `escseq` (siehe `char*`) für Unicode-Zeichen kann verwendet werden, wenn ein Zeichen nicht direkt über die Tastatur eingeben werden kann oder wenn Unicode-Zeichen über Dateien eingegeben werden sollen, die selbst nicht in Unicode codiert sind (siehe auch Abschnitt „Unicode-Ersatzdarstellung“ auf Seite 53).

Wenn als `escsymb` mit @PAR ESCAPE-CHARACTER=‘%‘ etwa das Zeichen `%` definiert ist, wäre `'Hast du mal nen %U20ac?'` ein gültiger Operand vom Typ `chars*`.

Operand	Definition
comment	chars

Beliebiger Kommentar.

Operand	Definition
message	chars

Beliebiger Text (bis Anweisungs-Ende inclusive Leerzeichen), der bei Aufruf des EDT als Unterprogramm mit @RETURN oder @HALT an das rufende Programm übergeben wird. Die Zeichenfolge darf maximal 80 Zeichen enthalten und nur aus abdruckbaren Zeichen aus dem Zeichenvorrat von `EDF03IRV` bestehen.

Operand	Definition
name	chars

Zeichenfolge mit maximaler Länge von 8 Zeichen, die dem SDF-Datentyp `<alphanum-name 1..8>` entspricht.

Operand	Definition
str	’ [chars*] ’ [*int] | B ’ binary ’ [*int] | X ’ hex ’ [*int]

Folge von in Hochkommas eingeschlossenen Zeichen, die entweder durch Zeichen aus dem jeweiligen Zeichensatz oder in ihrer binären- bzw. hexadezimalen Codierung angegeben werden. Bei der Zeichendarstellung ist auch die Unicode-Ersatzdarstellung `escseq` erlaubt. Ob in der Zeichendarstellung die leere Zeichenfolge erlaubt ist, hängt von der jeweiligen Anweisung ab und wird dort beschrieben.

Wird `B` oder `X` verwendet, so erfolgt die Interpretation der Binär- oder Hexadezimalzahl immer im Zeichensatz der aktuellen Arbeitsdatei (oder in `EDF041` wenn die aktuelle Arbeitsdatei keinen Zeichensatz hat), unabhängig davon, was mit der Zeichenfolge durch das verwendete Kommando dann geschieht.
Soll die Zeichenfolge ein Hochkomma enthalten, so müssen hierfür zwei Hochkommas geschrieben werden. Das gültige Zeichen für Hochkomma kann über die Anweisung @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 32768 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

– Die Angabe 'A''BC''D' erzeugt die Zeichenfolge A'BC'D.
– Die Angabe 'ABC'*5 erzeugt die Zeichenfolge ABCABCABCABCABC.
– Die Angabe X'C1F2'*4 erzeugt die Zeichenfolge A2A2A2A2, falls der Zeichensatz EDF041 eingestellt ist.
– Die Angabe B'11110000'*3 erzeugt die Zeichenfolge 000, falls der Zeichensatz EDF041 eingestellt ist.
– Die Angabe 'Das ist das %U0391 und %U03a9' erzeugt die Zeichenfolge 'Das ist das Α und Ω', falls ein Unicode-Zeichensatz eingestellt ist und für @PAR ESCAPE-CHARACTER das Zeichen % vereinbart wurde.

Hinweise

– Wird bei hexadezimaler Angabe eine ungerade Anzahl von Zeichen verwendet, so wird linksbündig mit Nullen aufgefüllt. So ist etwa 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. So ist etwa B'1' gleichwertig mit B'00000001' oder B'1111'*2 gleichwertig mit B'00001111'*2.

Operand	Definition
strchar	str | U'unicode'

Einzelnes Zeichen in Hochkomma oder in binärer- bzw. hexadezimaler Codierung oder Direktangabe der UTF16-Codierung. Die resultierende Zeichenfolge muss genau die Länge 1 haben.

Operand	Definition
strspec	str

Einzelnes Zeichen in Hochkomma oder in binärer- bzw. hexadezimaler Codierung. Die resultierende Zeichenfolge muss genau die Länge 1 haben und aus dem Zeichenvorrat `spec` stammen.

Operand	Definition
string	str | line[:cols[,...] [:]] | svarex[:cols[,...] [:]]

Eine direkt oder indirekt angegebene Zeichenfolge.

Wird `string` indirekt über eine Zeichenfolgevariable oder eine Zeilennummer angegeben, so verwendet der EDT als `string` den Inhalt der Zeichenfolgevariablen bzw. den 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 oder Zeichenfolgevariablen gewünscht, so kann dies über entsprechende Spaltenangaben zum Ausdruck gebracht werden. Werden Spaltenwerte angegeben, die die Zeilenlänge überschreiten, so wird dafür die entsprechende Anzahl von Leerzeichen eingesetzt.
Enthält z.B. die Zeile 6 die Zeichenfolge `AB3CD6EF9` und ist `string` angegeben als `6:1-3,9,8,9,8-9,5-7,30,1,30-32,1:`, so ist die hierüber ausgedrückte Zeichenfolge `AB39F9F9D6E␣A␣␣␣A`.

Wird in einer Anweisung als `string` eine Zeilennummer angegeben, die selbst durch diese EDT-Anweisung verändert wird, so wird als Operand der ursprüngliche Inhalt der Zeile genommen. 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`.

Operand	Definition
search	string

Analog zum Operandentyp `string`, kann jedoch innerhalb der Alternative `str` anstelle eines *apostrophe (')* auch ein *quotation mark (")* verwendet werden. Beide Zeichen können mit der Anweisung @QUOTE umdefiniert werden.

Der Operandentyp `search` wird lediglich in der @ON-Anweisung zur Bezeichnung des Suchbegriffs verwendet.
Zur Bedeutung der Zeichen *apostrophe* und *quotation mark* bei der Suchanweisung siehe Abschnitt „Suchen mit @ON“ auf Seite 81.

Operand	Definition
text	chars*

Folgeeingabe bei einigen L-Modus-Anweisungen.

Der Text wird wie eine Eingabe im L-Modus behandelt, d.h. er wird entweder als Anweisung betrachtet und sofort ausgeführt oder als Dateneingabe betrachtet und an der Position der aktuellen Zeilennummer in die aktuelle Arbeitsdatei eingefügt. Die Unterscheidung erfolgt, je nachdem ob die ersten vom Leerzeichen verschiedenen Zeichen des Textes aus einem, zwei oder keinem EDT-Anweisungssymbol bzw. Benutzeranweisungssymbol bestehen (siehe Abschnitt „Eingabe im L-Modus“ auf Seite 131).

Stellt der Text eine Dateneingabe dar und enthält Unicode-Ersatzdarstellungen `escseq`, werden diese nur dann als Unicode-Zeichen interpretiert, wenn sowohl das verwendete Fluchtsymbol mit @PAR ESCAPE-CHARACTER definiert ist als auch @PAR DATA-REPLACEMENT=ON gesetzt ist. Stellt der Text eine Anweisung dar, werden Unicode-Ersatzdarstellungen nur innerhalb von Literalen als Unicode-Zeichen interpretiert, dann aber unabhängig von der Einstellung bei @PAR DATA-REPLACEMENT.

7.4.5 Zeilen und Zeilenbereiche

In diesem Abschnitt werden die verschiedenen Formate definiert, mit denen in EDT-Anweisungen Zeilen und Zeilenbereiche angesprochen werden können.

Operand	Definition
lnum	n | fraction | n fraction

Zeilennummer.
Die erlaubten Werte für `lnum` liegen zwischen `0.0001` und `9999.9999`.

Operand	Definition
inc	lnum

Schrittweite für Zeilennummern.
Die erlaubten Werte für `inc` liegen zwischen `0.0001` und `9999.9999`.

Operand	Definition
lsym	lvar | * | % | $ | ? | loopsymb

Symbolisch angegebene Zeilennummer, die entweder als Zeilennummervariable oder als eines der im Folgenden erklärten Symbole angegeben wird (siehe auch Abschnitt „Symbolische Zeilennummern“ auf Seite 36).

`*` Aktuelle Zeilennummer, also die Zeilennummer, die der EDT im L-Modus zuletzt als Quittung auf die Datenstation geschrieben hat. Ist die Datei leer, hat `*` den Wert `1`.

`%` Niedrigste Zeilennummer der Datei. Ist die Datei leer, hat `%` den Wert `1`.

`$` Höchste Zeilennummer der Datei. Ist die Datei leer oder enthält sie nur eine einzige Zeile, so ist `$=%`.

`?` Zeilennummer der 1. Trefferzeile einer vorausgegangenen @ON-Anweisung. Der Wert beim Start des EDT ist `0`. Dieser wird lediglich durch eine erfolgreiche @ON-Anweisung verändert. Die symbolische Zeilennummer `?` hat also nach @ON den gleichen Wert, wie `#L00`.

Die symbolischen Zeilennummern `*`,`%`,`$` und `?` beziehen sich immer auf die aktuelle Arbeitsdatei, auch wenn sie in einer Bereichsangabe für eine andere Arbeitsdatei oder für eine externe Datei verwendet werden. Ihre jeweiligen Werte werden auch durch die @STATUS-Anweisung ausgegeben.

Operand	Definition
line	{ lsym [op inc] | lnum } [op { ivar | nL | lsym }]

Mit dem Operanden `line` können Zeilennummern direkt oder als Ausdruck, der ihre Position relativ zu anderen Zeilennummern beschreibt, angegeben werden.
Kommt in dem für `line` spezifizierten Ausdruck weder `ivar` noch `nL` vor, wird die Zeilennummer absolut berechnet, d.h. die Zeilennummer ergibt sich durch Addition oder Subtraktion der Werte von `lsym` bzw. `lnum`.
Wenn `ivar` oder `nL` spezifiziert sind, wird die Zeilennummer logisch bestimmt, d.h. es wird ausgehend von einem Absolutwert die durch `ivar` oder `nL` angegebene Anzahl von existierenden Zeilen übersprungen, unabhängig von der Schrittweite der Zeilennummerierung.

Im Ausdruck `nL` darf `n` nicht den Wert `0` annehmen. Jedoch ist es möglich, in einer Ganzzahlvariablen diesen Wert abzulegen. Eine logisch bestimmte Zeilennummer kann man nur dann zuweisen, wenn es eine solche Zeile auch gibt. Andernfalls wird eine Fehlermeldung ausgegeben.

Beispiele

– Mit `17.1` wird die gegebene Zeile direkt und absolut angesprochen.

– Ist `*=50.1` und `%=1.0000`, so spricht man mit `*+3.5-%` die Zeile `52.6000` absolut an.

– Ist `%=1.0000` und `#I15=6`, so wird mit `%+#I15` die 6. logische Zeile hinter Zeilennummer `1.0000` angesprochen (dies muss nicht unbedingt Zeile `7.0000` sein).

– Ist `%=1.000`, so spricht man mit `%+2L` die 2. logische Zeile hinter `1.0000` an.

– Ist `%=1.0000` und `*=3.0000`, so spricht man mit `%+*` die Zeile `4.0000` an.

– Ist `*=50.1` und `#I5=1`, so spricht man mit `*+3.5+#I5` die Zeile an, die der Zeile `53.6000` logisch folgt (dies muss nicht unbedingt die Zeile `53.7000` sein).

– Ist `*=50.1000`, so spricht man mit `*+3.5+6L` die 6. logische Zeile hinter `53.6000` an.

Operand	Definition
lines	rangesymb | line[[.] - [.] line]

Ein zusammenhängender Zeilenbereich.

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.
Der Operand `rangesymb` stellt das Bereichssymbol dar, das über die @RANGE-Anweisung vereinbart werden kann und das mit dem Zeichen `'&'` und dem Bereich `0.0001-9999.9999` vorbelegt ist.
Da das Minuszeichen sowohl als Symbol zur Festlegung der Bereichsgrenzen wie auch innerhalb des Ausdrucks `line` in seiner arithmetischen Bedeutung benutzt werden kann, können Mehrdeutigkeiten entstehen. Um dieses Problem zu vermeiden, gelten die folgenden Konventionen:
– Endet die erste Bereichsgrenze mit `lsym`, so schreibt man `lsym.-line`.
– Beginnt die zweite Bereichsgrenze mit `lsym`, schreibt man `line-.lsym` ... bzw. `line-.lsym op` ...

Mit der Angabe von `.` (Punkt) wird zum Ausdruck gebracht, dass es sich um einen Bereich, nicht etwa um eine Differenz handelt. Statt des Punktes kann auch die Ziffer `0` geschrieben werden. Anders als bei `svars` darf hier der Punkt nur in den angegebenen Fällen geschrieben werden.

Beispiele
– Mit `1-10` werden die Zeilen 1 bis 10 spezifiziert.
– Mit `%.-5` wird der Bereich von der 1. Zeile der Datei bis Zeile 5 ausgewählt.
– Mit `%+5L-.$-10L` wird der Bereich von der 6. Zeile der Datei bis zur 10. Zeile vor Ende der Datei ausgewählt.
– Mit `%.-$` wird die gesamte Datei spezifiziert.
– Mit `*+2.1-?.-.%+5L` wird der Bereich von `*+2.1-?` bis zur 6. Zeile der Datei erfasst.
– Mit `#L1.-#L2` bezeichnet man den Bereich von `#L1` bis `#L2`.
– Mit `12.011` wird lediglich Zeile `12.011` ausgewählt.
– Mit `#L9` wird lediglich die Zeile ausgewählt, deren Nummer in `#L9` abgelegt ist.

7.4.6 Spalten und Spaltenbereiche

In diesem Abschnitt werden die verschiedenen Formate definiert, mit denen in EDT-Anweisungen Spalten und Spaltenbereiche angesprochen werden können.

Operand	Definition
col	int

Spaltennummer, deren Wert zwischen `1` und `32768` liegen darf. Einige EDT-Anweisungen verlangen jedoch zwingend einen kleineren `col`-Wert.

Operand	Definition
cols	col[-col]

Ein zusammenhängender Spaltenbereich

Das zweite `col` darf nicht kleiner als das erste sein. Wird es nicht angegeben, so kann das erste `col` entweder nur die angegebene Spalte oder auch den Bereich von `col` bis zum Zeilenende bezeichnen. Welcher Fall zutrifft, ist bei der jeweiligen Anweisung beschrieben.

Ist das zweite `col` angegeben und größer als die Zeilenlänge, so erstreckt sich der Spaltenbereich bis an das Ende solcher Zeilen. Wie verfahren wird, wenn bereits das erste `col` größer als die Zeilenlänge ist, wird bei der jeweiligen Anweisung beschrieben.

Operand	Definition
cols*	col[-col]

Spaltenbereich, bei dem die jeweilige Spaltennummer relativ zum Satzende angegeben wird. Die für `cols` gegebenen Regeln gelten für `cols*` analog, wobei der Bereich `col1-col2` für jeder Zeile durch (`zeilenlänge-col2+1`)-(`zeilenlänge-col1+1`) zu ersetzen ist. Wenn dabei negative Spaltennummern entstehen, ist der Wert `1` einzusetzen.

7.4.7 Dateinamen und andere Systembezeichner

In diesem Abschnitt werden spezielle Formate für Zeichenfolgen definiert, die in EDT-Anweisungen externe Objekte, z.B. Dateien oder Bibliothekselemente bezeichnen.

Zeichenfolgen ohne definierten Begrenzer (z.B. Hochkomma) werden bei der Syntax-Analyse zunächst ohne Berücksichtigung ihrer semantischen Einschränkungen extrahiert. Dabei werden (wenn beim Operandentyp nicht anders beschrieben) alle Zeichen ab dem ersten Nichtleerzeichen bis zu einem intern definierten Begrenzerzeichen oder dem Anweisungs-Ende benutzt.
Als Begrenzer verwendet der EDT normalerweise das Leerzeichen, das Komma, das Gleichheitszeichen und die runden Klammern. Erst nach dieser Extraktion des Operanden erfolgt die Prüfung auf Länge, Zeichenvorrat und erlaubte Syntax.

Operand	**Definition**
entry	chars | .svar

Name einer Einsprungstelle (`ENTRY`) oder eines `CSECT`-Abschnittes in einem Programm, wobei Groß-/Kleinschreibung relevant ist. Der Name darf nicht länger als 32 Zeichen sein und muss den Restriktionen für Symbol-Namen des BLS genügen.

Operand	**Definition**
freetype	name

Freier Typname eines Bibliothekselements als Zeichenfolge der Länge 2 bis 8 Zeichen, die nicht mit `$` oder `SYS` beginnen darf.

Operand	**Definition**
eltype	S | M | R | C | P | J | D | X | H | L | U | F | *STD | freetype | .svar

Typ eines Bibliothekselements.
Zur Angabe des Elementtyps sind auch freie Typnamen zugelassen. Es erfolgt keine Prüfung auf den Basistyp. Bei einigen Anweisungen sind nur textartige Typen zugelassen.

Operand	Definition
elname	chars | .svar

Name eines Bibliothekselements, der dem SDF-Datentyp <composed-name 1..64 with-under> entspricht.

Operand	Definition
progname	chars

Name eines Programms, dessen Anweisungen vom EDT syntaktisch geprüft werden sollen. Der Name muss dem SDF-Datentyp <structured-name 1..30> entsprechen.

Operand	Definition
file	str

Dateiname, der über abdruckbare, hexadezimale oder binäre Darstellung angegeben werden kann.
Der Dateiname darf aus max. 54 Zeichen ohne Wildcards bzw. 80 Zeichen mit Wildcards bestehen, wobei die DVS-Restriktionen für Dateinamen zu berücksichtigen sind.
Die Angabe von Wildcards ist nicht bei allen Anweisungen erlaubt. Es handelt sich um die vom DVS akzeptierten Wildcards, die nicht mit den im EDT vereinbarten Musterzeichen zu verwechseln sind.
Ob partielle Dateinamensangaben erlaubt sind, hängt ebenfalls von der Anweisung ab und wird dort definiert.
Bei einigen Anweisungen ist zusätzlich die Angabe von '/' als Kennzeichnung für die Verwendung eines bestimmten Kettungsnamen erlaubt. Das ist bei den jeweiligen Anweisungen erläutert.

Operand	Definition
linkname	chars

Spezifiziert eine Datei oder Jobvariable über ihren Kettungsnamen.
Der Name muss dem SDF-Datentyp <filename 1..8 without-gen> entsprechen.

Operand	Definition
path	chars | .svar

Pfadname einer Datei bzw. einer Jobvariablen, der direkt oder über eine Zeichenfolgevariable angegeben werden kann.
Der Pfadname darf aus max. 54 Zeichen ohne Wildcards bzw. 80 Zeichen mit Wildcards bestehen, wobei die DVS-Restriktionen für Dateinamen zu berücksichtigen sind.
Die Angabe von Wildcards ist nicht bei allen Anweisungen erlaubt.
Es handelt sich um die vom DVS akzeptierten Wildcards, die nicht mit den im EDT vereinbarten Musterzeichen zu verwechseln sind. Ob partielle Dateinamensangaben erlaubt sind, hängt ebenfalls von der Anweisung ab und wird dort definiert.

Operand	Definition
modlib	path

Bibliothek, in der sich ein Modul befindet, der vom EDT geladen werden soll.
Der Name muss dem SDF-Datentyp `<filename 1..54 without-vers>` entsprechen.

Operand	Definition
ver	* | int

Versionsnummer einer katalogisierten Datei.
Hierfür kann entweder `*` oder `int` angegeben werden, wobei `int` für eine Zahl zwischen 0 und 255 steht. Zur Verwendung von Versionsnummern beim Lesen und Schreiben von Dateien siehe Abschnitt „Versionsnummern“ auf Seite 147.

Operand	Definition
vers	chars | *STD

Versionsbezeichnung eines Bibliothekselements.
Die Bezeichnung muss dem SDF-Datentyp `<composed-name 1..24 with-under>` entsprechen.

Operand	Definition
xpath	chars | .svar

Zeichenfolge, die den Namen einer POSIX-Datei angibt.
Die Angabe des vollständigen Pfadnamens ist erlaubt. Wird kein vollständiger Pfadname angegeben, wird die Datei im jeweils aktuellen POSIX-Verzeichnis lokalisiert.
Leerzeichen und Komma innerhalb des Namens sind nur bei Angabe in `svar` möglich.
Der Pfadname muss dem SDF-Datentyp `<posix-pathname 1..1023>` entsprechen.

7.4.8 Sonstige

In diesem Abschnitt werden Syntaxelemente definiert, die in keine der oben aufgeführten Kategorien passen.

Operand	Definition
formal	&id

Formalparameter der Form `&id`, der in der @PARAMS-Anweisung einer @DO-Prozedur anzugeben ist.
Der Operanden-Rest `id` ist der Name, der aus bis zu 7 Buchstaben oder Ziffern bestehen kann. Das erste Zeichen muss ein Buchstabe sein.
Dieser Operand findet für Schlüsselwort- und Stellungsparameter Verwendung.

Operand	Definition
param	' [chars*] ' | chars

Parameter, die über @DO an eine auszuführende Prozedurdatei übergeben werden. Diese bestehen aus einer beliebigen Zeichenfolge, die man dann in Hochkommas einzuschließen hat, wenn man ein Komma oder eine schließende runde Klammer oder ein Unicode-Zeichen in Ersatzdarstellung als Bestandteil der Parameterfolge übergeben will.

In diesem Fall ist jedes zu übergebende Hochkomma im Parameterausdruck durch zwei zu schreibende Hochkommas zu kennzeichnen. Die eingrenzenden Hochkommas können durch @QUOTE nicht redefiniert werden.

Operand	Definition
procnr	int

Nummer einer Arbeitsdatei.
Es sind Werte zwischen `0` und `22` zugelassen. Bei einigen Anweisungen ist der Wert `0` nicht zugelassen.

Operand	Definition
m	dd | ivar

Satzmarkierung `1..9`.

Operand	Definition
hpos	>[n] | <[n] | <<

Relative horizontale Positionieranweisung.

Operand	Definition
vpos	op n | vpos-op | vpos-op (m[,...])

Relative vertikale Positionieranweisung.

Operand	Definition
vpos-op	+ | - | ++ | --

Vertikaler Positionieroperand.

8 Übersicht der Anweisungen

In dieser Übersicht werden die Anweisungen des EDT thematisch geordnet vorgestellt und stichwortartig beschrieben. Die genaue Operandensyntax, eine ausführliche Funktionsbeschreibung, sowie Hinweise zu Einschränkungen und Fehlermeldungen entnehme man der nachfolgenden alphabetisch geordneten Beschreibung. Wenn die gleiche Anweisung mehreren Themenkreisen zugeordnet werden kann, wird sie auch mehrfach aufgeführt.

Es werden auch die im Unicode-Modus des EDT V17.0 nicht mehr unterstützten Anweisungen mit einem entsprechenden Hinweis genannt.

8.1 Einstellungen des EDT

Die folgenden Anweisungen dienen dazu, die vordefinierten Standardeinstellungen des EDT zu ändern und erlauben so dem Anwender, das Verhalten und das Erscheinungsbild des EDT weitgehend seinen Bedürfnissen anzupassen.

@:	Definiert ein neues Anweisungssymbol.	F-Modus L-Modus
@AUTOSAVE	Schaltet das automatische, zeitgesteuerte Sichern von Arbeitsdateien ein- oder aus.	F-Modus L-Modus
@BLOCK **@BK**	Schaltet den geblockten Ein-Ausgabemodus (BLOCK-Modus) des EDT ein bzw. aus.	F-Modus L-Modus
@CHECK (Format 1)	Veranlasst die Protokollierung jeder Zeile, die in einer Arbeitsdatei oder in einer Zeichenfolgevariablen durch eine Anweisung aufgebaut oder verändert wird. Ferner kann die Überprüfung der Anzahl Zeichen pro Zeile gesteuert werden.	F-Modus L-Modus
@CHECK (Format 2)	Veranlasst die Prüfung, ob der angegebene Bereich der aktuellen Arbeitsdatei bzw. von Zeichenfolgevariablen verlustfrei in den angegebenen Ziel-Zeichensatz konvertiert werden kann.	F-Modus L-Modus
@CODE	Diese Anweisung wird nur noch im Kompatibilitäts-Modus unterstützt.	F-Modus L-Modus

@**CODEN**AME (Format 1)	Bestimmt die Zeichensätze für Arbeitsdateien und Zeichenfolgevariablen.	F-Modus L-Modus
@**CODEN**AME (Format 2)	Legt den vom EDT im Dialogbetrieb verwendeten Kommunikations-Zeichensatz für den Datenaustausch mit der Datensichtstation fest.	F-Modus L-Modus
@**D**ELIMIT	Vereinbart Zeichen, die beim Suchen mit @ON als Textbegrenzerzeichen wirken.	F-Modus L-Modus
@**I**NPUT (Format 3)	Legt fest, wie der EDT Texteingaben im L-Modus interpretieren soll.	L-Modus
@**LOW**ER	Legt fest, ob der EDT bei der Eingabe von Daten und Anweisungen von der Datensichtstation Kleinbuchstaben in Großbuchstaben umsetzt oder nicht.	F-Modus L-Modus
@**P-K**EYS	Lädt im Dialogbetrieb die programmierbaren Tasten (P-Tasten) der Tastatur mit einer vom EDT vorgegebenen Standardbelegung oder zeigt die vom EDT vorgegebene Standardbelegung an.	F-Modus L-Modus
@**PAR**	Legt Einstellungen des EDT fest. Dabei handelt es sich um Einstellungen für die Darstellung am Bildschirm, für die Steuerung des Verhaltens bei der Eingabe, um Standardwerte für Anweisungen sowie um die Vereinbarung von Sonderbedeutungen für bestimmte Zeichen.	F-Modus L-Modus
@**Q**UOTE @**QE**	Definiert die Begrenzersymbole *apostrophe* (Hochkomma) und *quotation mark* (Anführungszeichen) um.	F-Modus L-Modus
@**RA**NGE	Vereinbart ein Symbol für einen Zeilenbereich.	F-Modus L-Modus
@**SEA**RCH-OPTION	Trifft Voreinstellungen für die Suche mit der @ON-Anweisung.	F-Modus L-Modus
@**SETS**W	Benutzer- und Auftragsschalter werden gesetzt oder zurückgesetzt.	F-Modus L-Modus
@**SYM**BOLS	Vereinbart die Jokersymbole *asterisk* und *slash* zur Suche mit Platzhaltern. Mit dem Operanden `FILLER` wird ein Füllzeichen vereinbart.	F-Modus L-Modus
@**SYN**TAX	Für die Eingabe im L-Modus wird die Art der Syntaxkontrolle eingestellt. Außerdem kann der Test-Modus ein- oder ausgeschaltet werden.	F-Modus L-Modus
@**T**ABS (Format 1)	Für das Positionieren mit dem Hardwaretabulator werden Tabulatorpositionen definiert und ihre aktuellen Werte ausgegeben.	F-Modus L-Modus

@**T**ABS (Format 2)	Für das Positionieren mit Softwaretabulatoren werden Tabulatorzeichen und Tabulatorpositionen definiert und deren aktuelle Werte ausgegeben.	F-Modus L-Modus
@**V**DT	Es wird das Bildschirmformat für den F-Modus eingestellt. Für den L-Modus hat die Anweisung keine Funktion mehr und wird nur noch aus Kompatibilitätsgründen unterstützt.	F-Modus L-Modus
@**VT**CSET	Legt fest, ob bei der Ausgabe nach SYSOUT die Line-Modus-Steuerzeichen, die eventuell in den auszugebenden Datenzeilen enthalten sind, unverändert übermittelt werden oder in Schmierzeichen umgesetzt werden.	F-Modus L-Modus
@**E**DIT (Format 2)	Schaltet im L-Modus des Dialogbetriebs den Eingabestrom auf Datenstations-Eingabe um. Es wird mit WRTRD gelesen und die aktuelle Zeilennummer als Eingabeaufforderung ausgegeben. Bei Eingabe der Anweisung im F-Modus wird zunächst in den L-Modus gewechselt.	F-Modus L-Modus
@**E**DIT (Format 3)	Schaltet im L-Modus des Dialogbetriebs den Eingabestrom auf Eingabe von SYSDTA um. Es wird mit RDATA gelesen. Bei Eingabe der Anweisung im F-Modus wird zunächst in den L-Modus gewechselt.	F-Modus L-Modus
@**E**DIT (Format 4)	Schaltet im F-Modus für die aktuelle Arbeitsdatei zwischen der vollständigen Darstellung der Sätze und der Darstellung eines Satzausschnitts im Datenfenster der aktuellen Arbeitsdatei um.	F-Modus
@**HEX**	Schaltet den Hexadezimal-Modus für die aktuelle Arbeitsdatei ein oder aus.	F-Modus
@**I**NDEX	Schaltet die Zeilennummernanzeige im F-Modus für die aktuelle Arbeitsdatei im jeweiligen Datenfenster ein bzw. aus.	F-Modus
@**SC**ALE	Schaltet im F-Modus die Anzeige eines Spaltenzählers (Zeilenlineal) für die aktuelle Arbeitsdatei im Arbeitsfenster ein oder aus.	F-Modus
@**SPLIT**	Schaltet im F-Modus die Darstellung eines zweiten Arbeitsfensters am Bildschirm ein- bzw. aus.	F-Modus
@**Z**ERO-RECORDS	Diese Anweisung wird nur noch im Kompatibilitäts-Modus unterstützt.	F-Modus L-Modus

8.2 Bearbeiten von Dateien

Mit den Anweisungen zur Dateibearbeitung wird eine einheitliche Schnittstelle für alle vom EDT unterstützten Dateitypen (DVS-Dateien, Bibliothekselemente und POSIX-Dateien) geboten.
Diese Anweisungen sollten daher bevorzugt zur Dateibearbeitung benutzt werden. Die alten, für die Dateitypen uneinheitlichen Anweisungen werden nur noch aus Kompatibilitätsgründen unterstützt.

@**CL**OSE	Veranlasst das Rückschreiben der aktuellen Arbeitsdatei auf Platte oder Band, das Schließen der geöffneten SAM-, ISAM- oder POSIX-Datei oder des Bibliothekselements und das Löschen der Arbeitsdatei.	F-Modus L-Modus
@**C**OPY (Format 1)	Liest eine existierende SAM-, ISAM- oder POSIX-Datei oder ein Bibliothekselement komplett in die aktuelle Arbeitsdatei ein. Die Arbeitsdatei braucht dabei nicht leer zu sein. Nach dem Einlesen wird die Datei bzw. das Bibliothekselement wieder geschlossen.	F-Modus L-Modus
@**D**ELETE (Format 3)	Löscht Dateien oder Elemente einer Bibliothek.	F-Modus L-Modus
@**OP**EN (Format 1)	Öffnet eine existierende SAM-, ISAM- oder POSIX-Datei oder ein Bibliothekselement und liest sie bzw. es in die aktuelle Arbeitsdatei ein oder erzeugt eine Datei neu und öffnet diese zur Bearbeitung.	F-Modus L-Modus
@**SHOW** (Format 1)	Gibt das Inhaltsverzeichnis einer Bibliothek oder eine Liste von Dateien aus dem BS2000-Katalog oder aus einem POSIX-Verzeichnis aus.	F-Modus L-Modus
@**W**RITE (Format 1)	Erzeugt eine SAM-, ISAM- oder POSIX-Datei oder ein Bibliothekselement neu und schreibt den Inhalt der aktuellen Arbeitsdatei in die neue Datei oder überschreibt eine existierende Datei mit dem Inhalt der aktuellen Arbeitsdatei oder schreibt den Inhalt der aktuellen Arbeitsdatei in eine mit @OPEN (Format 1) geöffnete Datei zurück. Eine geöffnete Datei bleibt bei @WRITE geöffnet, der Inhalt der Arbeitsdatei bleibt erhalten.	F-Modus L-Modus

8.3 Alte Anweisungen zum Bearbeiten von SAM- und ISAM-Dateien

Diese Anweisungen zur Bearbeitung von SAM- und ISAM-Dateien werden nur noch aus Kompatibilitätsgründen unterstützt. Es sollten stattdessen die Anweisungen aus dem Abschnitt „Bearbeiten von Dateien“ auf Seite 200 verwendet werden.

@**EL**IM	Löscht Sätze einer ISAM-Datei. Werden alle Sätze gelöscht, bleibt - im Gegensatz zu @UNSAVE - der Dateiname im Katalog bestehen.	F-Modus L-Modus
@**F**ILE	Stellt für @GET, @READ, @WRITE (Format 2), @SAVE, @OPEN (Format 2) und @ELIM einen Dateinamen als Standardwert ein. Man kann sowohl einen Dateinamen voreinstellen, der nur für die aktuelle Arbeitsdatei gilt (expliziter lokaler @FILE-Eintrag), als auch einen Dateinamen, der für alle Arbeitsdateien gilt (globaler @FILE-Eintrag).	F-Modus L-Modus
@**G**ET	Liest eine ISAM-Datei von Platte oder Band ganz oder teilweise in die aktuelle Arbeitsdatei ein.	F-Modus L-Modus
@**OP**EN (Format 2)	Öffnet eine ISAM-Datei zur Bearbeitung direkt auf der Platte. Sie kann bereits existieren, vor dem Öffnen neu erzeugt werden oder als Kopie einer existierenden SAM- oder ISAM-Datei entstehen. Das Öffnen von ISAM-Dateien zur realen Bearbeitung ist nur in der Arbeitsdatei 0 möglich. Diese muss leer sein oder eine mit @OPEN (Format 2) real geöffnete Datei enthalten.	F-Modus L-Modus
@**REA**D	Liest eine SAM-Datei ganz oder teilweise von Platte oder Band in die aktuelle Arbeitsdatei ein.	F-Modus L-Modus
@**SA**VE	Schreibt den Inhalt der aktuellen Arbeitsdatei ganz oder teilweise als ISAM-Datei auf Platte.	F-Modus L-Modus
@**U**NSAVE	Löscht eine BS2000-Datei und ihren Katalogeintrag.	F-Modus L-Modus
@**W**RITE (Format 2)	Schreibt den Inhalt der aktuellen Arbeitsdatei ganz oder teilweise als SAM-Datei auf Platte oder Band.	F-Modus L-Modus

8.4 Alte Anweisungen zum Bearbeiten von POSIX-Dateien

Diese Anweisungen zur Bearbeitung von POSIX-Dateien werden nur noch aus Kompatibilitätsgründen unterstützt. Es sollten stattdessen die Anweisungen aus dem Abschnitt „Bearbeiten von Dateien“ auf Seite 200 verwendet werden.

@**XC**OPY	Liest eine POSIX-Datei, die im POSIX-Dateisystem abgelegt ist, in die aktuelle Arbeitsdatei ein.	F-Modus L-Modus
@**X**OPEN	Öffnet eine POSIX-Datei, die im POSIX-Dateisystem abgelegt ist, und liest sie in die aktuelle Arbeitsdatei ein.	F-Modus L-Modus
@**XW**RITE	Schreibt den Inhalt der aktuellen Arbeitsdatei in eine POSIX-Datei. Die Arbeitsdatei bleibt dabei erhalten.	F-Modus L-Modus

8.5 Wechseln oder Positionieren der Arbeitsdatei

Die folgenden Anweisungen, die hauptsächlich im F-Modus Verwendung finden, dienen dazu, den gewünschten Ausschnitt einer Arbeitsdatei zur Bearbeitung auf den Bildschirm zu bringen.

+	Positioniert in der Arbeitsdatei vorwärts (Richtung Dateiende). Es kann um eine bestimmte Anzahl von Zeilen vorwärts positioniert werden oder zu einem Satz mit bestimmten Satzmarkierungen.	F-Modus
-	Positioniert in der Arbeitsdatei rückwärts (Richtung Dateianfang). Es kann um eine bestimmte Anzahl von Zeilen rückwärts positioniert werden oder zu einem Satz mit bestimmten Satzmarkierungen.	F-Modus
++	Positioniert an das Ende der Arbeitsdatei bzw. auf den letzten Satz mit bestimmten Satzmarkierungen.	F-Modus
- -	Positioniert an den Anfang der Arbeitsdatei bzw. auf den ersten Satz mit bestimmten Satzmarkierungen.	F-Modus
<	Positioniert in der Arbeitsdatei horizontal nach links, d.h. das Datenfenster kann spaltenweise nach links (in Richtung Datensatzanfang) verschoben werden.	F-Modus

>	Positioniert in der Arbeitsdatei horizontal nach rechts, d.h. das Datenfenster kann spaltenweise nach rechts (in Richtung Datensatzende und darüber hinaus) verschoben werden.	F-Modus
<<	Positioniert in der Arbeitsdatei horizontal zum Satzanfang, d.h. das Datenfenster wird spaltenweise nach links bis zum Anfang des Datensatzes verschoben.	F-Modus
#	siehe @SETF-Anweisung	F-Modus
@**EN**D	Bewirkt im L-Modus, dass die aktuelle Arbeitsdatei verlassen wird. Es wird wieder in die Arbeitsdatei zurückgeschaltet, in der die @PROC-Anweisung gegeben wurde, mit der die jetzt aktuelle Arbeitsdatei eingestellt wurde. Im F-Modus bewirkt @END das Beenden des EDT-Laufs.	F-Modus L-Modus
@**O**N (Format 3)	Bewirkt, dass alle Sätze, in denen ein Treffer festgestellt wird, mit der angegebenen Satzmarkierung gekennzeichnet werden. Im F-Modus wird das Arbeitsfenster auf den ersten Treffersatz positioniert.	F-Modus L-Modus
@**PRO**C (Format 1)	Schaltet im L-Modus in eine andere Arbeitsdatei um. Diese Arbeitsdatei wird damit zur aktuellen Arbeitsdatei.	L-Modus
@**SETF**	Positioniert mit oder ohne Wechsel der aktuellen Arbeitsdatei das Arbeitsfenster für eine Arbeitsdatei gleichzeitig vertikal und horizontal. Im F-Modus darf diese Anweisung mit # abgekürzt werden (sofern sie mit Operanden angegeben wurde).	F-Modus L-Modus
0..22	Bewirkt den Wechsel in eine andere Arbeitsdatei.	F-Modus
$0..$22	Bewirkt den Wechsel in eine andere Arbeitsdatei.	F-Modus

8.6 Behandlung der Zeilennummern

Die folgenden Anweisungen ermöglichen es, Zeilennummern und Schrittweite innerhalb einer Arbeitsdatei den aktuellen Erfordernissen anzupassen. Ferner können Zelennummern in Variablen abgelegt oder Sätze fortlaufend nummeriert werden.

@	Siehe @SET (Format 6).	F-Modus L-Modus
@+	Die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die nächste aktuelle Zeile umgeschaltet.	L-Modus
@-	Die aktuelle Zeilennummer wird um die aktuelle Schrittweite vermindert oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die vorangehende aktuelle Zeile umgeschaltet.	L-Modus
@PAR	Definiert mit dem Operanden INCREMENT die aktuelle Schrittweite.	F-Modus L-Modus
@**R**ENUMBER	Die in der Arbeitsdatei befindlichen Zeilen werden neu nummeriert. Es kann sowohl die Zeilennummer angegeben werden, die die erste Zeile der Arbeitsdatei erhalten soll als auch die Schrittweite, mit der neu nummeriert werden soll.	F-Modus L-Modus
@**SEQ**UENCE (Format 1)	Bewirkt, dass der EDT in jede Zeile eines zusammenhängenden Zeilenbereichs eine Zahl schreibt. In die erste Zeile des Zeilenbereichs wird eine vorgegebene, maximal 8-stellige Zahl geschrieben (eventuell mit führenden Nullen), die auch die Stellenzahl aller folgenden Zahlen festlegt. Alle folgenden Zahlen sind jeweils die Summe aus der vorhergehenden Zahl und einer vorgegebenen Schrittweite.	F-Modus L-Modus
@**SEQ**UENCE (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.	F-Modus L-Modus
@**SEQ**UENCE (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 als Dualzahl und prüft, ob die Dualzahlen eine aufsteigende Folge bilden.	F-Modus L-Modus

@SET (Format 3)	Weist einer Zeilennummervariablen einen Wert zu. Dieser Wert kann entstehen aus: der Angabe einer Zeilennummer, dem Wert einer Ganzzahlvariablen, der Angabe einer Zeilennummer als Zeichenfolge oder dem Binärwert der ersten 4 Byte einer Zeichenfolge.	F-Modus L-Modus
@SET (Format 6)	Definiert die aktuelle Zeilennummer und die aktuelle Schrittweite oder stellt frühere Werte für die Zeilennummer und die Schrittweite wieder her.	F-Modus L-Modus

8.7 Erzeugen, Einfügen und Ändern von Texten

Die folgenden Anweisungen finden Verwendung, wenn in größeren Bereichen nach dem gleichen Schema Änderungen am Text durchgeführt werden sollen. Innerhalb von EDT-Prozeduren können sie zur Automatisierung häufig wiederkehrender Änderungen eingesetzt werden.

@**CO**LUMN	Fügt in bestehenden Arbeitsdateizeilen bzw. Zeichenfolgevariablen ab der angegebenen Spaltenposition Text ein oder ersetzt diesen. Zusätzlich werden Leerzeichen am Zeilenende gelöscht.	F-Modus L-Modus
@**CONV**ERT	Konvertiert in Zeilenbereichen Klein- zu Großbuchstaben oder Groß- zu Kleinbuchstaben.	F-Modus L-Modus
@**CR**EATE (Format 1)	Erzeugt eine Zeile mit dem angegebenen Inhalt.	F-Modus L-Modus
@**CR**EATE (Format 2)	Weist einer Zeichenfolgevariablen eine Zeichenfolge zu.	F-Modus L-Modus
@**CR**EATE (Format 3)	Liest von der Datensichtstation oder von SYSDTA eine Zeichenfolge und erzeugt eine Zeile mit deren Inhalt.	F-Modus L-Modus
@**CR**EATE (Format 4)	Liest von der Datensichtstation oder von SYSDTA eine Zeichenfolge und erzeugt eine Zeichenfolgevariable mit deren Inhalt.	F-Modus L-Modus
@**O**N (Format 6)	Sucht nach einer Zeichenfolge und ersetzt die Trefferzeichenfolge durch den angegebenen Text.	F-Modus L-Modus
@**O**N (Format 7)	Sucht nach einer Zeichenfolge und fügt vor oder nach der Trefferzeichenfolge Text ein oder ersetzt diesen.	F-Modus L-Modus
@**PRE**FIX	Stellt jeder Zeile bzw. Zeichenfolgevariable des angegebenen Bereiches eine Zeichenfolge voran.	F-Modus L-Modus

@**SDF**TEST	Prüft, ob ein Zeilenbereich syntaktisch korrekte SDF-Kommandos bzw. syntaktisch korrekte SDF-Anweisungen enthält.	F-Modus L-Modus
@**SEP**ARATE	Bricht eine Zeile oder einen Zeilenbereich in mehrere Zeilen um. Die Umbruchstelle kann durch ein Satztrennzeichen oder durch eine Spaltenposition angegeben werden.	F-Modus L-Modus
@**SORT**	Sortiert zusammenhängende Zeilenbereiche in der aktuellen Arbeitsdatei aufsteigend oder absteigend. Durch die Angabe eines Spaltenbereichs kann der für die Sortierung relevante Teil des Satzes eingeschränkt werden.	F-Modus L-Modus
@**SU**FFIX	Fügt an das Ende jeder Zeile bzw. Zeichenfolgevariablen des angegebenen Bereiches eine Zeichenfolge an.	F-Modus L-Modus
@**T**ABS (Format 3)	Expandiert Softwaretabulatoren in Arbeitsdateien und Zeichenfolgevariablen, falls ein Tabulatorzeichen und entsprechende Tabulatorpositionen definiert sind (siehe @TABS, Format 2).	F-Modus L-Modus
@**U**PDATE	Diese Anweisung wird nur noch im Kompatibilitäts-Modus unterstützt.	L-Modus

8.8 Kopieren und Übertragen von Zeilen

Mit den folgenden Anweisungen können größere Bereiche des Textes kopiert oder verschoben werden, wenn die intuitive Arbeitsweise mit Kurzanweisungen im F-Modus zu umständlich wäre.

@**C**OPY (Format 2)	Kopiert Zeilen der aktuellen oder einer anderen Arbeitsdatei bzw. Zeichenfolgevariable in die aktuelle Arbeitsdatei.	F-Modus L-Modus
@**M**OVE	Überträgt Zeilen der aktuellen oder einer anderen Arbeitsdatei bzw. Zeichenfolgevariablen in die aktuelle Arbeitsdatei und löscht sie an ihren ursprünglichen Positionen.	F-Modus L-Modus
@**O**N (Format 4)	Kopiert alle mit der angegebenen Satzmarkierung markierten Sätze der zu durchsuchenden Zeilenbereiche in die angegebene Arbeitsdatei.	F-Modus L-Modus
@**O**N (Format 5)	Sucht nach einer Zeichenfolge und kopiert die Trefferzeilen in die angegebene Arbeitsdatei.	F-Modus L-Modus

8.9 Löschen von Arbeitsdateien, Zeilen, Texten und Satzmarkierungen

Die folgenden Anweisungen bieten verschiedene Möglichkeiten Sätze, Teile von Sätzen oder ganze Arbeitsdateien zu löschen.

@**D**ELETE (Format 1)	Löscht Zeilen und Zeichenfolgevariablen ganz oder teilweise.	F-Modus L-Modus
@**D**ELETE (Format 2)	Löscht Arbeitsdateien vollständig.	F-Modus L-Modus
@**D**ELETE (Format 4)	Löscht Satzmarkierungen.	F-Modus L-Modus
@**DR**OP	Löscht die angegebenen Arbeitsdateien vollständig.	L-Modus
@**O**N (Format 8)	Löscht die Trefferzeichenfolge in dem durchsuchten Bereich.	F-Modus L-Modus
@**O**N (Format 9)	Sucht nach einer Zeichenfolge und löscht den Inhalt einer Arbeitsdateizeile bzw. Zeichenfolgevariablen vor oder nach der Trefferzeichenfolge.	F-Modus L-Modus
@**O**N (Format 10)	Sucht nach einer Zeichenfolge und löscht die Arbeitsdateizeilen bzw. den Inhalt der Zeichenfolgevariablen, die den Suchbegriff enthalten.	F-Modus L-Modus

8.10 Vergleichen von Arbeitsdateien

Die beiden Anweisungen zum Vergleich von Arbeitsdateien unterscheiden sich im Wesentlichen im Layout der Ausgabe und in der Möglichkeit nur Teilbereiche einer Arbeitsdatei vergleichen zu können.

@**COM**PARE (Format 1)	Vergleicht zwei Arbeitsdateien ganz oder teilweise miteinander. Das Ergebnis des Vergleichs kann wahlweise in eine Arbeitsdatei, auf SYSOUT oder auf SYSLST ausgegeben werden.	F-Modus L-Modus
@**COM**PARE (Format 2)	Vergleicht die Inhalte zweier Arbeitsdateien zeilenweise. Das Vergleichsergebnis legt der EDT in einer Arbeitsdatei ab. Wahlweise ist auch eine Ausgabe des Ergebnisses nach SYSLST, im L-Modus auch nach SYSOUT möglich.	F-Modus L-Modus

8.11 Wechseln des Arbeitsmodus oder des Betriebsmodus

Die folgenden Anweisungen ermöglichen den Wechsel zwischen L-Modus und F-Modus bzw. zwischen Unicode-Modus und Kompatibilitäts-Modus.

@DIALOG	Schaltet im Dialogbetrieb in den Bildschirmdialog um.	F-Modus L-Modus
@EDIT (Format 1)	Schaltet im Dialogbetrieb vom L-Modus in den F-Modus um.	F-Modus L-Modus
@EDIT (Format 2)	Schaltet im L-Modus des Dialogbetriebs den Eingabestrom auf Terminal-Eingabe um. Es wird mit WRTRD gelesen und die aktuelle Zeilennummer als Eingabeaufforderung ausgegeben. Bei Eingabe der Anweisung im F-Modus wird zunächst in den L-Modus gewechselt.	F-Modus L-Modus
@EDIT (Format 3)	Schaltet im L-Modus des Dialogbetriebs den Eingabestrom auf Eingabe von SYSDTA um. Es wird mit RDATA gelesen. Bei Eingabe der Anweisung im F-Modus wird zunächst in den L-Modus gewechselt.	F-Modus L-Modus
@MODE	Schaltet zwischen Unicode-Modus und Kompatibilitäts-Modus um.	F-Modus L-Modus

8.12 Ausgabe von Zeilen und Informationen

Die folgenden Anweisungen dienen dazu, Daten bzw. Informationen auszugeben. In den meisten Fällen kann man sich entscheiden, ob die Ausgabe in eine Arbeitsdatei oder nach SYSOUT bzw. SYSLST erfolgen soll.

[n] **#**	Variante der #-Anweisung (siehe dort). Es wird die nt-letzte ausgeführte Anweisung erneut in der Anweisungszeile ausgegeben.	F-Modus
#	Gibt eine der letzten vom EDT bereits ausgeführte Anweisung erneut in der Anweisungszeile aus.	F-Modus
@FSTAT	Gibt eine Liste von Dateien aus dem BS2000-Katalog wahlweise in eine Arbeitsdatei oder nach SYSOUT bzw. SYSLST aus.	F-Modus L-Modus

@**LIM**ITS	Gibt die niedrigste und höchste vergebene Zeilennummer sowie die Anzahl der Zeilen für die aktuelle Arbeitsdatei aus.	F-Modus L-Modus
@**L**IST	Gibt Bereiche einer Arbeitsdatei oder von Zeichenfolgevariablen auf SYSLST oder über den Drucker aus.	F-Modus L-Modus
@**LOG**	Schaltet die Protokollierung der Eingaben im Stapelbetrieb und im Dialog ein bzw. aus und steuert deren Umfang.	F-Modus L-Modus
@**O**N (Format 1)	Sucht nach einer Zeichenfolge und gibt den Inhalt jeder Zeile bzw. Zeichenfolgevariablen aus, in der ein Treffer festgestellt wird. Im Dialog erfolgt die Ausgabe nach SYSOUT und im Stapelbetrieb nach SYSLST.	F-Modus L-Modus
@**O**N (Format 2)	Sucht nach einer Zeichenfolge und gibt die Zeilennummern bzw. die Namen der Zeichenfolgevariablen sowie die Nummern der Spalten aus, in denen die Trefferzeichenfolgen beginnen.	F-Modus L-Modus
@**PA**GE	Bewirkt einen Seitenvorschub auf SYSLST.	F-Modus L-Modus
@**P**RINT	Gibt den Inhalt der angegebenen Zeilenbereiche bzw. der Zeichenfolgevariablen aus. Im Dialogbetrieb erfolgt die Ausgabe nach SYSOUT, im Stapelbetrieb nach SYSLST.	F-Modus L-Modus
@**PRO**C (Format 2)	Gibt die Nummer der aktuellen Arbeitsdatei, die Nummern aller freien Arbeitsdateien bzw. die Nummern aller belegten Arbeitsdateien aus.	L-Modus
@**SHOW** (Format 1)	Gibt das Inhaltsverzeichnis einer Bibliothek bzw. eine Liste von Dateien aus dem BS2000-Katalog oder aus einem POSIX-Verzeichnis aus.	F-Modus L-Modus
@**SHOW** (Format 2)	Gibt eine Liste der von XHCS unterstützten Zeichensätze aus. Im Dialogbetrieb werden zusätzlich die von der Datensichtstation unterstützten Zeichensätze gekennzeichnet.	F-Modus L-Modus
@**ST**ATUS	Gibt Einstellungen des EDT und der Systemumgebung sowie die Werte von Zeilennummer- und Ganzzahlvariablen aus.	F-Modus L-Modus
@**SHIH**	Gibt den Anweisungspuffer des EDT aus.	F-Modus L-Modus
@**TM**ODE	Gibt Informationen über den Prozess aus, unter dem der EDT abläuft. Die Informationen werden als Meldung ausgegeben.	F-Modus L-Modus

8.13 Unterbrechen oder Beenden des EDT

Die folgenden Anweisungen unterbrechen oder beenden den EDT bzw. führen Systemkommandos aus, ohne den EDT zu verlassen.

@**EN**D	Im L-Modus wird die aktuelle Arbeitsdatei verlassen. Es wird wieder in die Arbeitsdatei zurückgeschaltet, in der die @PROC-Anweisung gegeben wurde, mit der die jetzt aktuelle Arbeitsdatei eingestellt wurde. Im F-Modus bewirkt @END das Beenden des EDT-Laufs bzw. des Bildschirmdialogs nach @DIALOG.	F-Modus L-Modus
@**EX**EC	Beendet die EDT-Sitzung und lädt und startet das angegebene Programm.	F-Modus L-Modus
@**H**ALT	Beendet den EDT-Lauf, den Bildschirmdialog nach @DIALOG bzw. den EDT als Unterprogramm mit oder ohne Übergabe eines Textes an das aufrufende Programm.	F-Modus L-Modus
@**LO**AD	Beendet die EDT-Sitzung und lädt das angegebene Programm.	F-Modus L-Modus
@**RET**URN	Beendet in EDT-Prozeduren die Abarbeitung der Prozedur und kehrt an die Stelle ihres Aufrufs zurück. Wird die Anweisung @RETURN außerhalb einer EDT-Prozedur gegeben, wird der EDT-Lauf bzw. der Bildschirmdialog nach @DIALOG beendet.	F-Modus L-Modus
@**SY**STEM	Unterbricht (wie K2) den EDT-Lauf oder bringt ein Betriebssystemkommando zur Ausführung, ohne dass der EDT-Lauf unterbrochen wird.	F-Modus L-Modus

8.14 Ablaufsteuerung in EDT-Prozeduren

Die folgenden Anweisungen werden innerhalb von EDT-Prozeduren zur Steuerung des Ablaufs und zur Programmierung von Schleifen und Verzweigungen verwendet.

@CONTINUE	Führt keine Aktion aus. Die Anweisung wird verwendet, um in EDT-Prozeduren eine Zeile zu erzeugen, die über @GOTO angesprungen werden kann.	L-Modus
@GOTO	Führt in einer @DO-Prozedur einen unbedingten Sprung zu der angegeben Zeile aus.	@PROC
@IF (Format 1)	Prüft in EDT-Prozeduren und im L-Modus, ob zuvor EDT- oder DVS-Fehler aufgetreten sind. Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.	L-Modus
@IF (Format 2)	Vergleicht in EDT-Prozeduren Zeichenfolgen, Zeilennummern oder Ganzzahlen miteinander. Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.	L-Modus
@IF (Format 3)	Prüft in EDT-Prozeduren, ob der EDT bei der letzten Ausführung von @ON einen Treffer festgestellt hat bzw. ob die aktuelle Arbeitsdatei leer ist. Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.	L-Modus
@IF (Format 4)	Prüft in EDT-Prozeduren, welche Auftrags- bzw. Benutzerschalter ein - oder ausgeschaltet sind. Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.	L-Modus
@IF (Format 5)	Ermittelt in EDT-Prozeduren oder im L-Modus den aktuell eingestellten Betriebsmodus. Abhängig vom Ergebnis wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.	L-Modus
@RESET	Setzt EDT- und DVS-Fehlerschalter zurück.	F-Modus L-Modus

8.15 Verwalten und Ausführen von EDT-Prozeduren

Mit den folgenden Anweisungen werden EDT-Prozeduren gestartet und mit Parametern versorgt. Weitere Anweisungen dienen zum Wechsel zwischen verschiedenen Prozedurebenen sowie zur Initialisierung oder Veränderung von Variablen.

@DO (Format 1)	Startet eine @DO-Prozedur, d.h. die in der angegebenen Arbeitsdatei stehenden Textzeilen und EDT-Anweisungen werden abgearbeitet.	F-Modus L-Modus
@DO (Format 2)	Schaltet die Protokollierung der gelesenen Anweisungen (siehe den Operanden PRINT in Format 1 der @DO-Anweisung) an einer beliebigen Stelle innerhalb der Prozedur ein oder aus.	@PROC
@END	Im L-Modus wird die aktuelle Arbeitsdatei verlassen. Es wird wieder in die Arbeitsdatei zurückgeschaltet, in der die @PROC-Anweisung gegeben wurde, mit der die jetzt aktuelle Arbeitsdatei eingestellt wurde. Im F-Modus bewirkt @END das Beenden des EDT-Laufs bzw. des Bildschirmdialogs nach @DIALOG.	F-Modus L-Modus
@INPUT (Format 1)	Startet eine @INPUT-Prozedur aus einer beliebigen Datei. Die Anweisungen bzw. Datensätze aus der Datei werden sequentiell abgearbeitet.	F-Modus L-Modus
@INPUT (Format 2)	Startet eine @INPUT-Prozedur aus einer SAM- oder ISAM-Datei. Die Anweisungen bzw. Datensätze aus der Datei werden sequentiell abgearbeitet. Dieses Format wird nur noch aus Kompatibilitätsgründen unterstützt und sollte nicht mehr verwendet werden.	F-Modus L-Modus
@NOTE	Führt keine Aktion aus. Die Anweisung wird verwendet, um in EDT-Prozeduren Kommentare einzufügen.	L-Modus
@PARAMS	Definiert symbolische Parameter, die innerhalb einer @DO-Prozedur benutzt werden.	@PROC
@PROC (Format 1)	Schaltet in eine andere Arbeitsdatei um. Diese Arbeitsdatei wird damit zur aktuellen Arbeitsdatei.	L-Modus
@SET (Format 1)	Weist einer Ganzzahlvariablen einen Wert zu.	F-Modus L-Modus
@SET (Format 2)	Weist einer Zeichenfolgevariablen ein Wert zu.	F-Modus L-Modus

@SET (Format 4)	Fügt den Inhalt einer Ganzzahlvariablen, den Namen einer Zeichenfolgevariablen oder den Inhalt einer Zeilennummervariablen ab einer bestimmten Spalte in abdruckbarer Form in eine Arbeitsdateizeile oder Zeichenfolgevariable ein.	F-Modus L-Modus
@SET (Format 5)	Legt Datum oder Uhrzeit ab einer gewünschten Spalte in einer Zeichenfolgevariablen oder in einer Arbeitsdateizeile ab.	F-Modus L-Modus

8.16 Aufruf eines Anwenderprogramms

Die folgenden Anweisungen erlauben das Nachladen und Starten von Routinen, die vom Benutzer oder von Drittanbietern geschrieben wurden, um die Funktionalität des EDT zu erweitern.

@RUN	Aufruf einer Anwenderroutine. Diese Anweisung unterscheidet sich von der gleichnamigen Anweisung im Kompatibilitäts-Modus.	F-Modus L-Modus
@UNLOAD	Entlädt Module, die mit @USE geladen wurden.	F-Modus L-Modus
@USE	Definiert Benutzeranweisungen, indem man ein Benutzeranweisungssymbol und die zugehörige Anweisungsroutine festlegt.	F-Modus L-Modus

8.17 Arbeiten mit Jobvariablen

Die folgenden Anweisungen dienen der Bearbeitung von Jobvariablen.

@ERAJV	Löscht Einträge von Jobvariablen aus dem Katalog.	F-Modus L-Modus
@GETJV	Gibt den Wert einer Jobvariablen am Bildschirm aus, schreibt ihn in eine Arbeitsdatei oder weist ihn einer Zeichenfolgevariablen zu.	F-Modus L-Modus
@SETJV	Trägt eine Jobvariable in den Katalog ein bzw. weist ihr einen Wert zu.	F-Modus L-Modus
@STAJV	Gibt Eigenschaften von Jobvariablen am Bildschirm oder in eine Arbeitsdatei aus.	F-Modus L-Modus

8.18 Arbeiten mit S-Variablen

Die folgenden Anweisungen dienen der Bearbeitung von S-Variablen.

@GETLIST	Schreibt Elemente einer S-Listenvariablen in die aktuelle Arbeitsdatei.	F-Modus L-Modus
@GETVAR	Gibt den Wert einer S-Variablen am Bildschirm aus, schreibt ihn in eine Arbeitsdatei oder weist ihn einer Variaben zu.	F-Modus L-Modus
@SETLIST	Weist einer S-Listenvariablen Elemente zu. Dabei werden Werte aus Zeilen der aktuellen Arbeitsdatei oder aus Zeichenfolgevariablen übernommen.	F-Modus L-Modus
@SETVAR	Deklariert eine S-Variable bzw. weist einer S-Variablen einen Wert zu.	F-Modus L-Modus

9 Anweisungen des EDT (alphabetisch)

Die folgenden Abschnitte enthalten die ausführlichen Beschreibungen aller Anweisungen des EDT in alphabetischer Reihenfolge.

Für Anweisungen, die Sonderzeichen enthalten, bedeutet alphabetisch die durch den Zeichensatz EBCDIC.DF.04 vorgegebene Ordnung. Das Anweisungssymbol @ im Anweisungsnamen wird nicht berücksichtigt.

9.1 @< – Datenfenster nach links positionieren

Mit der Anweisung < wird in der Arbeitsdatei horizontal nach links positioniert, d.h. das Datenfenster kann spaltenweise nach links (in Richtung Datensatzanfang) verschoben werden.

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

Operation	Operanden	F-Modus
@<	[n]	

n Anzahl der Spalten, um die das Arbeitsfenster nach links verschoben werden soll. Es werden für n Werte zwischen `0` und `32768` akzeptiert. Wenn der angegebene Wert `n` größer als die aktuelle Spaltenposition ist, wird auf die erste Spalte positioniert.

Wird `n` weggelassen, positioniert der EDT um die aktuelle Zeilenlänge des Datenfensters (abhängig von der verwendeten Datensichtstation, deren Einstellung mittels @VDT-Anweisung und der Sichtbarkeit der Zeilennummernanzeige) nach links.

Befindet sich der EDT im `EDIT-LONG`-Modus, wird die Anweisung zwar akzeptiert und bearbeitet (erkennbar an der Veränderung der Zustandsanzeige), die Verschiebung wird aber erst sichtbar, wenn der `EDIT-LONG`-Modus verlassen wird.

Beispiel

Das Datenfenster beginnt ab Spalte 10 (siehe Zustandsanzeige).

```
   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 ···············································································


<9·····························································0001.00:00010(00)
```

Das Datenfenster soll um 9 Spalten nach links verschoben werden.

```
   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 ···············································································


·····························································0001.00:00001(00)
```

Durch das Verschieben beginnt das Datenfenster auf Spalte 1.

9.2 @<< – Datenfenster zum Satzanfang positionieren

Mit der Anweisung << wird in der Arbeitsdatei horizontal zum Satzanfang positioniert, d.h. das Datenfenster wird spaltenweise nach links bis zum Anfang des Datensatzes verschoben.
Nach dem Verschieben beginnt der dargestellte Ausschnitt der Datensätze auf Spalte 1.
Die Spaltennummer wird in der Zustandsanzeige des Arbeitsfensters ausgegeben.

Operation	Operanden	F-Modus
@<<		

Befindet sich der EDT im EDIT-LONG-Modus, wird die Anweisung zwar akzeptiert und bearbeitet (erkennbar an der Veränderung der Zustandsanzeige), die Verschiebung wird aber erst sichtbar, wenn der EDIT-LONG-Modus verlassen wird.

Beispiel

Das Datenfenster beginnt ab Spalte 10 (siehe Zustandsanzeige).

```
   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 ···············································································



<<·····························································0001.00:00010(00)
```

Das Datenfenster wird bis Spalte 1 nach links verschoben.

```
   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 ···············································································



·····························································0001.00:00001(00)
```

Durch das Verschieben beginnt das Datenfenster auf Spalte 1.

9.3 @+ – Erhöhen der aktuellen Zeilennummer

Mit der Anweisung @+ wird die aktuelle Zeilennummer um die aktuelle Schrittweite erhöht oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die nächste aktuelle Zeile umgeschaltet.

Operation	Operanden	L-Modus
@+	[:[text]]	

text
: EDT-Anweisung oder eine Dateneingabe, die nach der Erhöhung der aktuellen Zeilennummer um die aktuelle Schrittweite ausgeführt bzw. in die neue aktuelle Zeile eingefügt wird. Die Zeichenfolge wird so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu Abschnitt „L-Modus“ auf Seite 131).

  Der Operand text beginnt unmittelbar hinter dem Zeichen ':', d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

  Wird text nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Wird kein Operand angegeben wird nur die aktuelle Zeile verändert.

Eine indirekte Operandenangabe ist für diese Anweisung nicht erlaubt.

9.4 + – Datenfenster vorwärts positionieren

Mit der Anweisung + wird in der Arbeitsdatei vorwärts (Richtung Dateiende) positioniert. Es kann um eine bestimmte Anzahl von Zeilen vorwärts positioniert werden oder zu einem Satz mit bestimmten Satzmarkierungen.

Operation	Operanden	F-Modus
+ DUE		
+ F3		
+	{ n / ([m[,...]]) }	

Wenn die Anweisung ohne Operanden mit DUE oder einer Funktionstaste ungleich F3 abgeschickt wird, wird um die Anzahl der im Datenfenster sichtbaren Datensätze vorwärts geblättert. Dabei werden etwa eingeblendete Zeilenlineale oder durch Meldungen verdeckte Zeilen berücksichtigt.

Wenn die Anweisung ohne Operanden mit F3 abgeschickt wird, wird zum nächsten Satz mit irgendeiner Satzmarkierung (1..9) positioniert. Die Anweisung + F3 ist also äquivalent zu +() DUE (siehe unten).

n
: Anzahl der Zeilen, um die vorwärts geblättert werden soll. Es werden für n Werte zwischen 0 und 99999999 akzeptiert, jedoch wird höchstens soweit vorwärts geblättert, bis der letzte Datensatz der Arbeitsdatei in der ersten Bildschirmzeile zu sehen ist.

  Der Wert von n bestimmt die Anzahl der Sätze, um die das Datenfenster vorwärts positioniert wird, unabhängig von der aktuell eingestellten Schrittweite der Zeilennummernanzeige oder Lücken in der Nummerierung der Sätze.

m
: Ist eine der möglichen Satzmarkierungen (1..9), zu der positioniert werden soll. Es können mehrere Satzmarkierungen, durch Komma getrennt, angegeben werden. Es wird vorwärts zum nächsten Satz positioniert, der mit einer der angegebenen Satzmarkierungen gekennzeichnet ist - dieser wird in der ersten Bildschirmzeile des Datenfensters angezeigt. Markierungen mit Sonderfunktionen (siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden hier nicht ausgewertet.

  Ist m nicht angegeben, wird zum nächsten Satz mit irgendeiner Satzmarkierung (1..9) positioniert.

Hinweis

Wenn die Anweisung mit F3 abgeschickt wird, ist darauf zu achten, dass nur solche Kurzanweisungen gleichzeitig eingegeben werden, die mit F3 gesendet werden dürfen (siehe Abschnitt „Kurzanweisungen im F-Modus“ auf Seite 113). Andernfalls wird schon bei der Analyse der Kurzanweisungen abgebrochen und die Anweisung + nicht mehr ausgeführt.

Beispiel

Im oberen Datenfenster, das durch die Anweisung @SPLIT verkürzt wurde, ist ein Zeilenlineal eingeblendet. Die letzte Zeile des Datenfensters ist durch die Meldung EDT0901 verdeckt.

```
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN<·····················
   2.00 DUCK     DONALD    WALTSTREET 8   DISNEYLAND<·························
   3.00 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG<·····················
%  EDT0901 NO MATCH IN RANGE
+·····························································0001.00:00001(00)
   1.00 ·····································································


·····························································0000.00:00001(04)
```

Es soll mit + um die Datenfensterlänge vorwärts geblättert werden.

```
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   4.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN<·····················
   5.00 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN<·····················
   6.00 ·····································································
   7.00 ·····································································
·····························································0004.00:00001(00)
   1.00 ·····································································


·····························································0000.00:00001(04)
```

Es wird zur ersten vorher nicht sichtbaren Zeile (4.00) vorwärts geblättert.

Ein Beispiel zum Positionieren zu einem Satz mit Satzmarkierung findet man im Beispiel zur @ON-Anweisung Format 4.

9.5 ++ – Zum letzten (markierten) Satz der Arbeitsdatei positionieren

Mit der Anweisung ++ wird an das Ende der Arbeitsdatei bzw. auf den letzten Satz mit bestimmten Satzmarkierungen positioniert.

Operation	Operanden	F-Modus
++ DUE		
++ F3		
++	([m[,...]])	

Wenn die Anweisung ohne Operanden mit DUE oder einer Funktionstaste ungleich F3 abgeschickt wird, wird die Arbeitsdatei so positioniert, dass der letzte Satz der Arbeitsdatei in der letzten Bildschirmzeile des Datenfensters angezeigt wird (im Gegensatz dazu positioniert `+99999999` so, dass der letzte Satz der Arbeitsdatei in der ersten Bildschirmzeile des Datenfensters angezeigt wird).

Wenn die Anweisung ohne Operanden mit F3 abgeschickt wird, wird zum letzten Satz positioniert, der mit irgendeiner Satzmarkierung (`1..9`) gekennzeichnet ist. Die Anweisung ++F3 ist also äquivalent zu ++()DUE.

m
: Ist eine der möglichen Satzmarkierungen (`1..9`), zu der positioniert werden soll. Es können mehrere Satzmarkierungen, durch Komma getrennt, angegeben werden. Es wird vorwärts zum letzten Satz positioniert, der mit einer der angegebenen Satzmarkierungen gekennzeichnet ist - dieser wird in der ersten Bildschirmzeile des Datenfensters angezeigt. Markierungen mit Sonderfunktionen (siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden hier nicht ausgewertet.

  Ist m nicht angegeben, wird zum letzten Satz mit irgendeiner Satzmarkierung (`1..9`) positioniert.

Hinweis

Wenn die Anweisung mit F3 abgeschickt wird, ist darauf zu achten, dass nur solche Kurzanweisungen gleichzeitig eingegeben werden, die mit F3 gesendet werden dürfen (siehe Abschnitt Kurzanweisungen). Andernfalls wird schon bei der Analyse der Kurzanweisungen abgebrochen und die Anweisung ++ nicht mehr ausgeführt.

9.6 $0..$22 – Wechseln der Arbeitsdatei

Mit dieser Anweisung wechselt der EDT in eine andere Arbeitsdatei.

Operation	Operanden	F-Modus
$0..$22		

Der EDT zeigt die mit der Anweisung $0..$22 ausgewählte Arbeitsdatei in dem Arbeitsfenster an, in dem die Anweisung eingegeben wurde. Die Zeilenposition und die Spaltenposition werden auf die Werte eingestellt, die in der neu eingestellten Arbeitsdatei vorher gültig waren. Wurde die Arbeitsdatei vorher noch nicht benutzt, werden die Standardwerte eingestellt.

Mit der Anweisung @SETF kann man ebenfalls die Arbeitsdatei wechseln und gleichzeitig auf eine beliebige Zeile und Spalte positionieren.

Beispiel

```
   1.00 Mit dieser Anweisung wechselt der EDT in eine<·····························
   2.00 andere Arbeitsdatei<·····················································
   3.00 ··························································································
$7 ·····························································0001.00:00001(00)
```

Die Anweisung $7 wird eingegeben, um in die Arbeitsdatei 7 zu wechseln.

```
·····························································0000.00:00001(07)
```

Die Arbeitsdatei 7 wird im Arbeitsfenster angezeigt (siehe Statusanzeige).

9.7 @– – Herabsetzen der aktuellen Zeilennummer

Mit der Anweisung @- wird die aktuelle Zeilennummer um die aktuelle Schrittweite vermindert oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die vorangehende aktuelle Zeile umgeschaltet.

Operation	Operanden	L-Modus
@–	[:[text]]	

text
: EDT-Anweisung oder eine Dateneingabe, die nach der Verminderung der aktuellen Zeilennummer um die aktuelle Schrittweite ausgeführt bzw. in die neue aktuelle Zeile eingefügt wird. Die Zeichenfolge wird so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu Abschnitt „L-Modus“ auf Seite 131).

  Der Operand text beginnt unmittelbar hinter dem Zeichen ':', d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

  Wird text nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Wird kein Operand angegeben wird nur die aktuelle Zeile verändert.

Eine indirekte Operandenangabe ist für diese Anweisung nicht erlaubt.

9.8 – – Datenfenster rückwärts positionieren

Mit der Anweisung - wird in der Arbeitsdatei rückwärts (Richtung Dateianfang) positioniert. Es kann um eine bestimmte Anzahl von Zeilen rückwärts positioniert werden oder zu einem Satz mit bestimmten Satzmarkierungen.

Operation	Operanden	F-Modus
– DUE		
– F3		
–	{ n / ([m[,...]]) }	

Wenn die Anweisung ohne Operanden mit DUE oder einer Funktionstaste ungleich F3 abgeschickt wird, wird um die Anzahl der im Datenfenster sichtbaren Datensätze rückwärts geblättert. Dabei werden etwa eingeblendete Zeilenlineale oder durch Meldungen verdeckte Zeilen berücksichtigt.

Wenn die Anweisung ohne Operanden mit F3 abgeschickt wird, wird rückwärts zum nächsten Satz mit irgendeiner Satzmarkierung (1..9) positioniert. Die Anweisung -F3 ist also äquivalent zu -()DUE (siehe unten).

n — Anzahl der Zeilen, um die rückwärts geblättert werden soll. Es werden für n Werte zwischen 0 und 99999999 akzeptiert, jedoch wird höchstens soweit rückwärts geblättert, dass der erste Datensatz der Arbeitsdatei in der ersten Bildschirmzeile zu sehen ist.

Der Wert von n bestimmt die Anzahl der Sätze, um die das Datenfenster rückwärts positioniert wird, unabhängig von der aktuell eingestellten Schrittweite der Zeilennummernanzeige oder Lücken in der Nummerierung der Sätze.

m — Ist eine der möglichen Satzmarkierungen (1..9), zu der positioniert werden soll. Es können mehrere Satzmarkierungen, durch Komma getrennt, angegeben werden. Es wird rückwärts zum nächsten Satz positioniert, der mit einer der angegebenen Satzmarkierungen gekennzeichnet ist - dieser wird in der ersten Bildschirmzeile des Datenfensters angezeigt. Markierungen mit Sonderfunktionen (siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden hier nicht ausgewertet.

Ist m nicht angegeben, wird rückwärts zum nächsten Satz mit irgendeiner Satzmarkierung (1..9) positioniert.

Hinweis

Wenn die Anweisung mit F3 abgeschickt wird, ist darauf zu achten, dass nur solche Kurzanweisungen gleichzeitig eingegeben werden, die mit F3 gesendet werden dürfen (siehe Abschnitt Kurzanweisungen). Andernfalls wird schon bei der Analyse der Kurzanweisungen abgebrochen und die Anweisung - nicht mehr ausgeführt.

Beispiel

Im Datenfenster ist die Schrittweite 0.1 eingestellt.

```
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   10.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN<·····················
   10.10 DUCK     DONALD    WALTSTREET 8   DISNEYLAND<·························
   10.20 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG<·····················
   10.30 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN<·····················
   10.40 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN<·····················
   10.50 ·····························································
   10.60 ·····························································
   10.70 ·····························································



-3·····························································0010.00:00001(01)
```

Es soll mit -3 zurück geblättert werden.

```
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
    9.70 ÅNGSTRØM ANDERS    STERNWARTE     STOCKHOLM<··························
    9.80 BASLER   MARIO     SÄNBENER STR.1 80321 MUENCHEN<·····················
    9.90 BAYER    ALOIS     OTTOSTR.4      80123 MUENCHEN<·····················
   10.00 BERGER   ADALBERT  HOCHSTR.10     81234 MUENCHEN<·····················
   10.10 DUCK     DONALD    WALTSTREET 8   DISNEYLAND<·························
   10.20 GROOT    GUNDULA   HAFERSTR.16    89123 AUGSBURG<·····················
   10.30 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN<·····················
   10.40 STIWI    MANUELA   POSTWEG 3      80123 MUENCHEN<·····················
   10.50 ·····························································
   10.60 ·····························································



·····························································0009.70:00001(01)
```

Es wird 3 Sätze zurück positioniert (auf Zeile 9.70).

9.9 – – – Zum ersten (markierten) Satz der Arbeitsdatei positionieren

Mit der Anweisung -- wird an den Anfang der Arbeitsdatei bzw. auf den ersten Satz mit bestimmten Satzmarkierungen positioniert.

Operation	Operanden	F-Modus
– –DUE		
– –F3		
– –	([m[,...]])	

Wenn die Anweisung ohne Operanden mit DUE oder einer Funktionstaste ungleich F3 abgeschickt wird, wird auf den ersten Datensatz der Arbeitsdatei positioniert, d.h. der erste Satz der Arbeitsdatei wird in der ersten Zeile des Datenfensters angezeigt.

Wenn die Anweisung ohne Operanden mit F3 abgeschickt wird, wird zum ersten Satz positioniert, der mit irgendeiner Satzmarkierung (1..9) gekennzeichnet ist. Die Anweisung – –F3 ist also äquivalent zu – –()DUE.

m
: Ist eine der möglichen Satzmarkierungen (1..9), zu der positioniert werden soll. Es können mehrere Satzmarkierungen, durch Komma getrennt, angegeben werden. Es wird rückwärts zum ersten Satz positioniert, der mit einer der angegebenen Satzmarkierungen gekennzeichnet ist - dieser wird in der ersten Bildschirmzeile des Datenfensters angezeigt. Markierungen mit Sonderfunktionen (siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden hier nicht ausgewertet.

  Ist m nicht angegeben, wird zum ersten Satz mit irgendeiner Satzmarkierung (1..9) positioniert.

Hinweis

Wenn die Anweisung mit F3 abgeschickt wird, ist darauf zu achten, dass nur solche Kurzanweisungen gleichzeitig eingegeben werden, die mit F3 gesendet werden dürfen (siehe Abschnitt Kurzanweisungen). Andernfalls wird schon bei der Analyse der Kurzanweisungen abgebrochen und die Anweisung -- nicht mehr ausgeführt.

9.10 @> – Datenfenster nach rechts positionieren

Mit der Anweisung > wird in der Arbeitsdatei horizontal positioniert, d.h. das Datenfenster kann spaltenweise nach rechts (in Richtung Datensatzende und darüber hinaus) verschoben werden.

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

Operation	Operanden	F-Modus
@>	[n]	

n Anzahl der Spalten, um die das Arbeitsfenster nach rechts verschoben werden soll. Es werden für `n` Werte zwischen `0` und `32768` akzeptiert. Es wird jedoch höchstens so weit nach rechts positioniert, dass die letzte Spalte der Bildschirmzeile die vom EDT maximal erlaubte Spaltenposition eines Satzes (32768) anzeigt. Dies gilt unabhängig davon, ob die Arbeitsdatei überhaupt Sätze mit dieser Länge enthält.

Wird `n` weggelassen, positioniert der EDT um die aktuelle Zeilenlänge des Datenfensters (abhängig von der verwendeten Datensichtstation, deren Einstellung mittels @VDT-Anweisung und der Sichtbarkeit der Zeilennummernanzeige) nach rechts.

Befindet sich der EDT im `EDIT-LONG`-Modus wird die Anweisung zwar akzeptiert und bearbeitet (erkennbar an der Veränderung der Zustandsanzeige), die Verschiebung wird aber erst sichtbar, wenn der `EDIT-LONG`-Modus verlassen wird.

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····· ·····················································0001.00:00001(00)
```

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 ........................................................................




...............................................................0001.00:00010(00)
```

Durch das Verschieben beginnt das Datenfenster auf Spalte 10.

9.11 @: – Vereinbaren eines Anweisungssymbols

Mit der Anweisung @: wird ein neues Anweisungssymbol definiert.

Operation	Operanden	F-Modus, L-Modus
@:	spec	

spec Sonderzeichen für das neue Anweisungssymbol.

Ist der Operand spec keines der zulässigen Sonderzeichen, wird @: mit der Fehlermeldung EDT3952 abgewiesen.

Das aktuelle Bereichssymbol (siehe @RANGE) darf nicht für den Operanden spec verwendet werden und wird mit der Fehlermeldung EDT4315 abgewiesen.

Bei dieser Anweisung muss auch im F-Modus das bisher gültige Anweisungssymbol vorangestellt werden.

Beim Start des EDT ist @ das aktuelle Anweisungssymbol.

Achtung

Wird für den Operanden spec eines der Sonderzeichen <, > (nur im F-Modus), +, −, $, %, * oder ? (im F- und im L-Modus) verwendet, so ist die Eindeutigkeit der Anweisungen unter Umständen nicht mehr gewährleistet und es kann zu unerwünschten Verhaltensweisen kommen.

Wird für spec das Sonderzeichen : verwendet, hat man keine Chance mehr, diese Einstellung wieder rückgängig zu machen, da von diesem Moment an eine Folge von Doppelpunkten am Zeilenanfang als Folge von Anweisungssymbolen interpretiert wird.

Beispiel

```
3.     @print ------------------------------------------------------------ (1)
1.0000 Diese Anweisung ermoeglicht dem Benutzer die Vereinbarung eines
2.0000 neuen Anweisungssymbols.
3.     @:! ------------------------------------------------------------- (2)
3.     @print ------------------------------------------------------------ (3)
4.     !print ------------------------------------------------------------ (4)
1.0000 Diese Anweisung ermoeglicht dem Benutzer die Vereinbarung eines
2.0000 neuen Anweisungssymbols.
3.0000 @print
4.     !:@ -------------------------------------------------------------- (5)
4.
```

(1) Mit @PRINT wird der Inhalt der Arbeitsdatei ausgegeben.

(2) Als neues Anweisungssymbol wird ! vereinbart.

(3) @PRINT wird jetzt nicht als Anweisung sondern als Text interpretiert.

(4) Mit !PRINT wird der Inhalt der Arbeitsdatei ausgegeben.

(5) Als Anweisungssymbol wird wieder @ vereinbart.

9.12 # – Ausgeben der letzten Anweisung

Mit der Anweisung # kann man eine der letzten vom EDT bereits ausgeführten Anweisungen erneut in der Anweisungszeile ausgeben lassen. 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. Für n sind Werte zwischen 1 und 2048 zulässig.

Die Anweisung # oder 1# gibt die letzte vom EDT bereits ausgeführte Anweisung in der Anweisungszeile erneut aus. Mit 2# wird die zweite davor liegende gezeigt. Wird die #-Anweisung mehrfach nacheinander eingegeben, wird jedes Mal 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 jeder Eingabe, ungleich # oder einer leeren Eingabe wird wieder am Ende des Puffers aufgesetzt.

Ist die anzuzeigende Anweisung länger als die Kommandozeile, werden bis zu 3 Kommandozeilen angezeigt. Kann die Anweisung auch dann noch nicht vollständig dargestellt werden, wird sie ohne Meldung verkürzt.

War bei Eingabe einer Anweisung ein anderer als der aktuelle Kommunikationszeichensatz eingestellt, wird diese Anweisung zur Ausgabe in der Anweisungszeile entsprechend konvertiert. Sind einzelne Zeichen dabei nicht konvertierbar, wird die Meldung EDT5453 ausgegeben, die Anweisung aber ausgegeben, wobei die nicht konvertierbaren Zeichen durch Fragezeichen '?' ersetzt werden.

Der Anweisungspuffer enthält maximal 2048 Anweisungen, unabhängig von der Länge der einzelnen Anweisungen.

Mindestens ein Zeichen muss in der Anweisungszeile überschrieben, geändert bzw. hinzugefügt werden, wenn sie als Anweisung abgeschickt werden soll.

Es wird nicht berücksichtigt, ob eine Anweisung im oberen oder unteren Teil eines geteilten Bildschirmes eingegeben wurde. Die Speicherung im Anweisungspuffer erfolgt in der Reihenfolge von oben nach unten unabhängig von der Arbeitsdatei, auf die die Anweisung angewendet wurde. Anweisungen innerhalb einer Anweisungsfolge (durch ';' getrennte Anweisungen) werden einzeln abgelegt.

Wird eine **#**-Anweisung innerhalb einer Anweisungsfolge eingegeben, wird nach der Abarbeitung von # die Verarbeitung abgebrochen und die zuletzt ausgeführte Anweisung ausgegeben. Nach # stehende Anweisungen in der Anweisungsfolge werden nicht mehr ausgeführt.

Hinweis

Die Anweisung # mit nachfolgenden Operanden stellt im F-Modus eine Abkürzung der Anweisung @SETF dar und dient zum Positionieren des Datenfensters. Diese Anweisung sollte mit der hier beschriebenen nicht verwechselt werden.

Mit der Anweisung @SHIH (Show Input History) kann man den gesamten Anweisungspuffer des EDT in eine Arbeitsdatei ausgeben. Einzelne Anweisungen können dann mit der Kurzanweisung K in die Anweisungszeile kopiert werden.

9.13 @AUTOSAVE – Automatisches Sichern

Mit der Anweisung @AUTOSAVE wird ein automatisches, zeitgesteuertes Sichern von Arbeitsdateien ein- oder ausgeschaltet.

Operation	Operanden	F-Modus / L-Modus
@AUTOSAVE	{ [**ID**=name] [[,] **T**IME=n] [**ON**] / **O**FF }	

name — Frei wählbarer Identifikator für die Sicherungsdateien, der in die Namensbildung für die Sicherungsdatei eingeht (Standardwert: `EDT`).

n — Zeitdifferenz in Minuten (`0..255`) zwischen einer manuellen oder automatischen Sicherung und der nächsten automatischen Sicherung (Standardwert ist `5`).

Bei `TIME=0` wird nach jedem Dialogschritt gesichert.

ON — Die automatische Sicherung wird eingeschaltet (Standardwert).

OFF — Die automatische Sicherung wird ausgeschaltet. Alle Sicherungsdateien werden gelöscht.

Die Anweisung @AUTOSAVE ist nur im Dialogbetrieb wirksam, d.h. nur im Dialogbetrieb wird eine Datensicherung durchgeführt. Die Anweisung wird im Stapelbetrieb ohne Meldung ignoriert.

Bei Beginn des EDT-Laufs ist die automatische Sicherung ausgeschaltet.

Unter Umständen ist die Sicherung sehr langer Sätze (nahe `32768`) nicht möglich. In diesem Fall wird der betroffene Satz nur in der maximal möglichen Länge abgespeichert und die Warnung `EDT2405` ausgegeben.

Eine Sicherung findet bei jedem Einschalten der automatischen Sicherung statt oder nach einem Dialogschritt, also vor der nächsten Eingabeaufforderung, wenn die automatische Sicherung eingeschaltet ist und die beim Einschalten der automatischen Sicherung definierte Zeit seit der letzten automatischen Sicherung verstrichen ist. Gesichert werden jeweils alle nicht leeren Arbeitsdateien, deren Inhalt zum Zeitpunkt der automatischen Sicherung nicht auch bereits in Form einer Plattendatei existiert (entweder explizit vom Benutzer angelegt oder durch die automatische Sicherung in Form einer Sicherungsdatei). Real geöffnete ISAM-Dateien werden nicht 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 automatische Sicherung eingeschaltet wurde. Die Angabe nn ist die Nummer der aktuellen Arbeitsdatei.

Eine zugehörige Sicherungsdatei wird gelöscht, wenn:

– die Arbeitsdatei keine Sätze mehr enthält, (z.B. @DELETE)

– der Inhalt der Arbeitsdatei als Datei oder Bibliothekselement explizit gesichert wird. Die möglichen Anweisungen sind: @WRITE, @SAVE, @XWRITE und @CLOSE.

Alle Sicherungsdateien werden gelöscht bei der Anweisung @AUTOSAVE OFF, bei Beendigung des EDT-Laufs durch eine der Anweisungen @HALT, @END, @EXEC oder @LOAD oder bei Rückkehr zum Hauptprogramm.

Die Sicherungsdateien bleiben erhalten bei abnormaler Beendigung oder bei Verlassen des EDT durch K2 oder der Anweisung @SYSTEM ohne Rückkehr.

Hinweis

Der Inhalt und der Zeichensatz einer einzelnen Arbeitsdatei kann restauriert werden, indem in einer leeren Arbeitsdatei folgende Anweisung abgesetzt wird:

```
@COPY FILE=S.name.jjjj-mm-tt.hhmmss.SAVEnn,KEY=LINENUMBER
```

9.14 @BLOCK – Blockmodus einstellen

Die Anweisung @BLOCK schaltet den geblockten Ein-Ausgabemodus (BLOCK-Modus) des EDT ein bzw. aus.

Bei eingeschaltetem BLOCK-Modus kann man mit einer einzigen Eingabe über die Datensichtstation im L-Modus mehrere Zeilen erstellen oder mehrere Anweisungen eingeben, die sequentiell abgearbeitet werden oder beides gemischt. Dabei sind die einzelnen Zeilen bzw. Anweisungen durch das LZE-Zeichen zu trennen.

Operation	Operanden	F-Modus / L-Modus
@BLOCK **@BK**	[{ **O**N / **OF**F }]	

ON Schaltet den BLOCK-Modus ein (Standardwert).

OFF Schaltet den BLOCK-Modus aus.

Bei Beginn des EDT-Laufs ist der BLOCK-Modus standardmäßig eingeschaltet.

Im BLOCK-Modus können maximal so viele Zeichen in einem Block eingegeben werden, wie auf einer Bildschirmseite darstellbar sind.

Enthält der eingegebene Block eine fehlerhafte Anweisung, wird diese zusammen mit der aktuellen Zeilennummer und der entsprechenden Fehlermeldung zum Zeitpunkt ihrer Ausführung ausgegeben. Danach wird der Rest des Blocks ausgeführt.

Enthält bei geblockter Eingabe der eingegebene Block eine Anweisung @BLOCK OFF, werden die nachfolgenden Anweisungen oder Dateneingaben dieses Blocks ignoriert.

Im Stapelbetrieb wird die Anweisung @BLOCK ignoriert.

Beim Aufruf von @DO-Prozeduren wird der BLOCK-Modus auf OFF gesetzt. Wird die @DO-Prozedur wieder verlassen, wird der BLOCK-Modus wieder in den Zustand versetzt, den er vor Aufruf der @DO-Prozedur hatte.

9.15 @CHECK (Format 1) – Zeilen prüfen

Diese Anweisung ermöglicht die Protokollierung jeder Zeile, die in einer Arbeitsdatei oder in einer Zeichenfolgevariablen durch eine Anweisung aufgebaut oder verändert wird. Die betreffende Zeile wird im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST ausgegeben. Ferner kann mit der Anweisung @CHECK die Überprüfung der Zeilenlänge (Anzahl Zeichen pro Zeile) gesteuert werden, wobei die Tabulatorexpansion berücksichtigt wird.

Operation	Operanden	L-Modus
@CHECK	[{ **ON** / **OF**F }] [,] [col]	

ON Schaltet den CHECK-Modus ein (Standardwert). Bei eingeschaltetem CHECK-Modus wird jede Zeile nach SYSOUT ausgegeben, die in einer Arbeitsdatei oder in einer Zeichenfolgevariablen durch eine der folgenden Anweisungen aufgebaut oder verändert wird: @COLUMN, @COPY (Format 2), @CREATE, @MOVE, @ON (Formate 7 bis 10), @PREFIX, @SET (Formate 2, 4 und 5), @SEPARATE, @SEQUENCE (Formate 1 und 2), @SUFFIX.

OFF Schaltet den CHECK-Modus aus.

col Gibt die Anzahl der Zeichen pro Zeile für die Zeilenlängenprüfung an. Insbesondere werden auch Überschreitungen der vorgegebenen Zeilenlänge als Folge von eventuell stattfindenden Tabulatorexpansionen angezeigt. Der EDT überprüft die Anzahl der Zeichen in jeder Zeile, die neu eingegeben oder durch eine der folgenden Anweisungen aufgebaut wird: @+, @–, @IF, @SET (Format 6). Die Überprüfung der Anzahl Zeichen pro Zeile wird unabhängig vom aktuellen CHECK-Modus durchgeführt.

Ist eine Zeile länger als col, wird diese Zeile zwar angelegt, aber der EDT macht mit der Meldung EDT2901 darauf aufmerksam, dass die vorgegebene Anzahl Zeichen pro Zeile überschritten wurde. Der standardmäßig vorgegebene Wert von col entspricht dem Maximalwert von 32768 Zeichen pro Zeile, der kleinstmögliche Wert von col ist 1 Zeichen pro Zeile.

Der Wert für col kann auch durch die Anweisung @TABS verändert werden.

Beim Start des EDT ist der CHECK-Modus ausgeschaltet.

Die Anweisung @CHECK ist nur im L-Modus wirksam. Ein vorübergehender Wechsel in den F-Modus schaltet den CHECK-Modus aus. Der aktuelle Wert von col bleibt dabei unverändert.

Die Angabe von ON oder OFF hat keinen Einfluss auf den aktuellen Wert von col. Deshalb wird der aktuelle Wert von col nicht verändert, wenn nur ON oder OFF angegeben wird. Soll col auf den Standardwert 32768 zurückgesetzt werden, so muss dies durch explizite Angabe des Standardwerts geschehen (@CHECK [ON,]32768 oder @CHECK OFF,32768).

9.16 @CHECK (Format 2) – Zeilen auf Konvertierbarkeit prüfen

Mit diesem Format der Anweisung @CHECK kann geprüft werden, ob der angegebene Bereich der aktuellen Arbeitsdatei bzw. von Zeichenfolgevariablen verlustfrei in den angegebenen Ziel-Zeichensatz konvertiert werden kann.

Operation	Operanden	F-Modus, L-Modus
@**CH**ECK	[{ lines / svars } [,...] [:cols[:]]] [,] **C**ODE = { name / ***E**DT / ***F**ILE } [,**M**ARK[=m]] [,**L**ENGTH = { int / ***F**ILE }]	

lines — Einer oder mehrere Zeilenbereiche, die überprüft werden sollen. Enthält der angegebene Bereich keine oder nur Leerzeilen ist das Ergebnis der Prüfung positiv (Leerzeilen können immer verlustfrei geschrieben werden).

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, die überprüft werden sollen.

cols — Zusammenhängender Spaltenbereich in der aktuellen Arbeitsdatei bzw. in den angegebenen Zeichenfolgevariablen, der überprüft werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile bzw. Zeichenfolgevariable überprüft.

CODE= — Spezifiziert direkt oder symbolisch den Zeichensatz, für den die Überprüfung stattfinden soll.

name — Name des Zeichensatzes, für den die Überprüfung stattfinden soll. Der angegebene Zeilenbereich bzw. Bereich von Zeichenfolgevariablen wird probeweise in diesen Zeichensatz konvertiert. Je nach eingestellter Option werden fehlerhafte Zeilen markiert oder auf SYSOUT ausgegeben.

*EDT Der Zeichensatz, der für die aktuelle Arbeitsdatei eingestellt ist, soll für die Prüfung benutzt werden. Da eine Arbeitsdatei keine Sätze mit Zeichen enthalten kann, die im Zeichensatz der Arbeitsdatei ungültig sind, ist die Angabe von *EDT nur sinnvoll, wenn nur auf Überschreitung der Maximallänge geprüft werden soll (siehe Operand LENGTH).

*FILE Der Zeichensatz, der für die geöffnete Datei oder für das geöffnete Bibliothekselement im Katalog eingetragen ist, bzw. der bei der Anweisung @OPEN im Operanden CODE spezifiziert worden ist, soll für die Prüfung benutzt werden. Falls weder eine Datei noch ein Bibliothekselement geöffnet ist, wird die Angabe von *FILE mit der Fehlermeldung EDT5467 abgewiesen. Falls für die Datei der Zeichensatz *NONE eingetragen ist, findet die Prüfung gegen den Zeichensatz EDF03IRV statt.

MARK= Die Zeilen, die sich nicht verlustfrei in den angegebenen Zeichensatz konvertieren lassen, sollen mit einer Satzmarkierung versehen werden. Diese Option ist nur erlaubt, wenn ausschließlich Zeilenbereiche in der aktuellen Arbeitsdatei überprüft werden. Wird MARK nicht spezifiziert, werden die Zeilen bzw. Zeichenfolgevariablen, die sich nicht verlustfrei konvertieren lassen, nach SYSOUT ausgegeben.

m Satzmarkierung (1..9), mit der die Zeilen markiert werden sollen, die sich nicht verlustfrei in den angegebenen Zeichensatz konvertieren lassen. Wird m nicht angegeben, wird die Satzmarkierung 1 verwendet.

LENGTH= Es wird eine maximale Länge (in Bytes) spezifiziert, die bei der Konvertierung der Zeilen bzw. Zeichenfolgevariablen nicht überschritten werden darf. Da bei der Konvertierung in einen Unicode-Zeichensatz manche Zeichen durch mehrere Bytes codiert werden, kann eine Zeile bei der Konvertierung eventuell länger werden. Zeilen bzw. Zeichenfolgevariablen, die bei der Konvertierung die angegebene maximale Länge überschreiten, werden daher ebenfalls als fehlerhaft betrachtet.

int Legt explizit eine maximale Länge fest (1..32768).

*FILE Die maximale Länge wird aus dem Katalogeintrag der geöffneten Datei oder des geöffneten Bibliothekselements berechnet. Es wird der Wert gewählt, der unter Berücksichtigung anderer relevanter Dateieigenschaften (z.B. des Dateityps und des Satzformats) garantiert, dass alle überprüften Sätze verlustfrei in die Datei geschrieben werden können. Wenn weder eine Datei noch ein Bibliothekselement geöffnet ist, wird der Wert 32768 angenommen.

Wird weder lines noch svars angegeben, wird die gesamte aktuelle Arbeitsdatei überprüft.

Eine Zeile bzw. Zeichenfolgevariable wird als fehlerhaft im Sinne der von @CHECK durchgeführten Überprüfung betrachtet, wenn sie nach der Konvertierung entweder länger als der bei LENGTH spezifizierte Wert wird oder wenn sie Zeichen enthält, die bei der Konvertierung in den angegebenen Zeichensatz auf ein eventuell spezifiziertes Ersatzzeichen (siehe Anweisung @PAR SUBSTITUTION-CHARACTER) abgebildet werden müssten.

Findet der EDT während der Überprüfung keine fehlerhaften Zeilen bzw. Zeichenfolgevariablen oder werden nur Längenüberschreitungen moniert (siehe unten), kann der Anwender sicher sein, dass eine Konvertierung in den angegebenen Zeichensatz auch dann möglich ist, wenn kein Ersatzzeichen spezifiziert wurde. Andernfalls kann sich der Anwender entscheiden, ob er ein Ersatzzeichen spezifiziert (wenn dies nicht schon geschehen ist) und den dann entstehenden Informationsverlust in Kauf nimmt, oder ob er an den betroffenen Zeilen bzw. Zeichenfolgevariablen Änderungen durchführt, um eine verlustfreie Konvertierung zu ermöglichen.

Zeilen bzw. Zeichenfolgevariablen, bei denen eine Längenüberschreitung moniert wird, werden beim Schreiben in eine Datei, eine Jobvariable oder eine S-Variable abgeschnitten, falls die überprüfte Länge in diesem Fall physikalisch relevant ist (z.B. bei Dateien mit fester Satzlänge). Die Kürzung erfolgt in jedem Fall an einer gültigen Grenze zwischen zwei Zeichen. Die wirklich geschriebene Länge kann also um maximal 3 Byte kürzer sein als die überprüfte Länge. Wenn das Abschneiden nicht akzeptabel ist, muss der Anwender die monierten Zeilen bzw. Zeichenfolgevariablen selbst sinnvoll aufteilen oder kürzen.

Zusätzlich zur Markierung bzw. Ausgabe der fehlerhaften Zeilen bzw. Zeichenfolgevariablen wird eine Meldung ausgegeben, in der das Ergebnis der Prüfung zusammengefasst wird. Falls nur ungültige Zeichen gefunden wurden, wird die Meldung EDT5453 ausgegeben, falls nur Längenüberschreitungen gefunden wurden, die Meldung EDT5462. Falls sowohl ungültige Zeichen als auch Längenüberschreitungen vorkommen wird die Meldung EDT5456 ausgegeben.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

9.17 @CLOSE – Zurückschreiben und Schließen einer Datei

Mit @CLOSE wird eine geöffnete Datei evtl. zurück geschrieben, anschließend geschlossen, und die Arbeitsdatei wird gelöscht.

Operation	Operanden	F-Modus, L-Modus
@CLOSE	[**N**OWRITE | **C**ODE = name | ***E**DT | ***F**ILE]	

NOWRITE Die Arbeitsdatei wird gelöscht und nicht zurück geschrieben. Die geöffnete Datei bzw. das Bibliothekselement wird unverändert geschlossen. Bei einer real geöffneten Datei in der Arbeitsdatei 0 (siehe @OPEN, Format 2) ist `NOWRITE` wirkungslos.

CODE= Der Operand steuert, in welchem Zeichensatz die Arbeitsdatei geschrieben werden soll. Bei einer real geöffneten Datei in der Arbeitsdatei 0 ist der Operand wirkungslos.

Ist der Operand nicht angegeben und unterscheidet sich der Zeichensatz der SAM-Datei, der ISAM-Datei, des Bibliothekselements oder der beim Öffnen der POSIX-Datei verwendeten Zeichensatz von dem der Arbeitsdatei, dann wird im Stapelbetrieb die Meldung `EDT5457` ausgegeben, es wird nicht geschrieben und die Datei bleibt geöffnet. Im Dialogbetrieb wird die Abfrage

```
%  EDT0915 CONVERT TO FILE CCS (&00)? REPLY (Y=YES; N=NO)?
```

ausgegeben. Bei der Antwort `Y` wird vor dem Schreiben in den Zeichensatz der Datei umcodiert. Bei der Antwort `N` wird der Zeichensatz der Arbeitsdatei verwendet.

name Zeichensatz, der beim Schreiben verwendet werden soll. Es muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).

*FILE Die Arbeitsdatei wird vor dem Schreiben in den Zeichensatz der existierenden SAM-Datei oder ISAM-Datei, des Bibliothekselements oder den beim Öffnen der POSIX-Datei verwendeten Zeichensatz umcodiert. War dieser Zeichensatz `*NONE`, wird `EDF03IRV` verwendet.

*EDT Beim Schreiben wird der Zeichensatz der Arbeitsdatei verwendet, unabhängig davon, ob eine eventuell existierende Datei einen anderen Zeichensatz hat.

Die Anweisung @CLOSE wird mit der Fehlermeldung EDT5177 abgewiesen, wenn in der aktuellen Arbeitsdatei keine Datei oder kein Bibliothekselement mit @OPEN bzw. @XOPEN geöffnet ist.

Nach dem Schließen wird für SAM-Dateien, ISAM-Dateien und Bibliothekselemente der beim Schreiben verwendete Zeichensatz in den Katalog eingetragen.

Wird die Arbeitsdatei vor dem Schreiben umcodiert und enthält sie Zeichen, die im Zeichensatz der zu schreibenden Datei ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht geschrieben, sie bleibt geöffnet und die Fehlermeldung EDT5453 wird ausgegeben. Der Benutzer kann dann ein Ersatzzeichen definieren oder den Zeichensatz für das Schreiben verändern und @CLOSE erneut ausführen.

Enthält die Arbeitsdatei Zeilen, die für die zu schreibende Datei zu lang sind (z.B. bei fester Satzlänge der Datei) oder entstehen durch die Konvertierung solche Sätze (bei Unicode-Zeichensätzen möglich), dann wird der Schreibvorgang mit der Meldung EDT5444 abgebrochen.

Beim Schreiben von ISAM-Dateien wird der ISAM-Schlüssel aus der Zeilennummer gebildet, wenn beim Öffnen der Datei KEY=LINENUMBER oder KEY=IGNORE angegeben wurde. Wurde beim Öffnen KEY=DATA angegeben, wird der ISAM-Schlüssel aus dem Datenbereich übernommen. In diesem Fall muss der Benutzer sicherstellen, dass die Reihenfolge der Arbeitsdateisätze der Reihenfolge der ISAM-Schlüssel entspricht, da sonst das Schreiben mit der Meldung EDT4208 (DVS-Fehlercode 0AAB) abgewiesen wird.

Die Definition von Sekundärschlüsseln einer ISAM-Datei in einem Sekundärindex bleibt nach @CLOSE erhalten, es sei denn, die Schlüsselfelder im Datenbereich wurden inkonsistent geändert. In diesem Fall wird die Meldung EDT5246 ausgegeben und der Sekundärindex wird gelöscht.

Wird während der Verarbeitung einer geöffneten ISAM-Datei der Zeichensatz von dem oder in den Zeichensatz UTF16 geändert, bzw. geschieht dies implizit durch entsprechende Angabe des CODE-Operanden, kann die Datei nicht zurück geschrieben werden, da sich die Länge des Schlüsselfeldes dadurch ändern würde. In diesem Fall wird die @CLOSE-Anweisung mit der Fehlermeldung EDT5468 abgewiesen.

Nach dem Schließen einer SAM- oder ISAM-Datei mit @CLOSE gibt der EDT normalerweise den nicht benötigten Speicherplatz der Datei frei. Die Freigabe kann durch Setzen des Auftragsschalters 7 verhindert werden.

Wurde beim Öffnen einer ISAM-Datei zur realen Bearbeitung (siehe @OPEN, Format 2) die aktuelle Versionsnummer angegeben, der Operand AS jedoch nicht, dann wird nach @CLOSE die aktualisierte (um 1 erhöhte) Versionsnummer angezeigt.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Hinweis

Bei der Ausführung von @CLOSE wird implizit ein @DELETE (Format 2) für die aktuelle Arbeitsdatei ausgeführt. Dabei werden verschiedene Eigenschaften der Arbeitsdatei auf ihren Initialwert zurückgesetzt.

Beispiel

```
@OPEN FILE=DATEI1 ------------------------------------------------------ (1)
<EDT-Anweisungen> ------------------------------------------------------ (2)
@CLOSE CODE=UTFE ------------------------------------------------------- (3)
```

(1) Die Datei DATEI1 wird geöffnet und in dem Zeichensatz der Datei in die aktuelle Arbeitsdatei eingelesen.

(2) Die aktuelle Arbeitsdatei wird bearbeitet.

(3) Die aktuelle Arbeitsdatei wird in den Zeichensatz UTFE umcodiert und in die Datei DATEI1 zurück geschrieben. Im BS2000-Katalog erhält DATEI1 das Attribut CODED-CHARACTER-SET=UTFE. Die aktuelle Arbeitsdatei wird gelöscht.

9.18 @CODENAME (Format 1) – Zeichensatz für Arbeitsdateien und Zeichenfolgevariablen einstellen

Mit Format 1 der Anweisung @CODENAME kann der vom EDT für eine Arbeitsdatei bzw. eine Zeichenfolgevariable verwendete Zeichensatz eingestellt werden. Die bei @CODENAME getroffene Festlegung hat Vorrang vor der impliziten Wahl eines Zeichensatzes durch den EDT, z.B. aufgrund des Eintrags im Dateikatalog.

Operation	**Operanden**	**F-Modus, L-Modus**
@CODENAME	name [, { **L**OCAL / **G**LOBAL / $0...$22 / #S0...#S20 }] [, **F**ORCE = { **Y**ES / **N**O }]	

name — Name des einzustellenden Zeichensatzes. Der Zeichensatzname muss in XHCS bekannt sein, andernfalls wird die Anweisung mit der Meldung `EDT4980` abgewiesen. Der angegebene Zeichensatz wird für eine oder mehrere Arbeitsdateien bzw. für die Zeichenfolgevariable eingestellt, die über die weiteren Operanden bestimmt werden.

LOCAL — Es wird der angegebene Zeichensatz für die gerade aktuelle Arbeitsdatei eingestellt. Falls die Arbeitsdatei nicht leer und nicht der Operand `FORCE=YES` gegeben wurde, werden die enthaltenen Daten in den neuen Zeichensatz umcodiert.

GLOBAL — Es wird der angegebene Zeichensatz global für alle Arbeitsdateien des EDT eingestellt. Dabei werden, falls nicht der Operand `FORCE=YES` gegeben wurde, alle nicht leeren Arbeitsdateien umcodiert. Die Einstellung gilt nicht für Zeichensätze von Zeichenfolgevariablen.

Der Operand wird hauptsächlich verwendet, wenn der EDT über die alte Unterprogrammschnittstelle (V16-Format) als Unterprogramm aufgerufen wurde. Wenn der EDT erkennt, dass für alle Arbeitsdateien der gleiche Zeichensatz eingestellt ist, wird dieser Zeichensatz an der alten Unterprogrammschnittstelle im globalen Kontrollblock `EDTPARG` eingetragen. Falls das aufrufende Programm diesen Eintrag auswertet, empfiehlt es sich also, mit dem Operanden `GLOBAL` für alle Arbeitsdateien den gleichen Zeichensatz einzustellen (und diesen auch nachträglich nicht mehr zu ändern).

Das Verhalten an der alten Unterprogrammschnittstelle, falls kein globaler Zeichensatz eingestellt ist, sowie weitere Details zur Verwendung von Zeichensätzen an der Unterprogrammschnittstelle entnehme man dem Handbuch Unterprogramm-Schnittstellen [1].

$0..$22 Es wird der angegebene Zeichensatz für die angegebene Arbeitsdatei eingestellt. Falls die Arbeitsdatei nicht leer und nicht der Operand FORCE=YES gegeben wurde, werden die enthaltenen Daten in den neuen Zeichensatz umcodiert.

#S0..#S20 Es wird der angegebene Zeichensatz für die angegebene Zeichenfolgevariable eingestellt. Falls nicht der Operand FORCE=YES gegeben wurde, werden die in der Zeichenfolgevariablen enthaltenen Daten in den neuen Zeichensatz umcodiert.

FORCE=

NO Die Daten der angegebenen Arbeitsdatei bzw. Zeichenfolgevariablen werden in den angegebenen Zeichensatz umcodiert.

YES Die Angabe von FORCE=YES ist nur für 7-Bit- und 8-Bit-Zeichensätze erlaubt und führt dann zum Umetikettieren der angegebenen Arbeitsdatei bzw. Zeichenfolgevariablen, d.h. der Inhalt der Arbeitsdatei bzw. der Zeichenfolgevariablen bleibt unverändert (als Bytefolge), wird aber in dem angegebenen Zeichensatz neu interpretiert. Diese Funktion dient dazu, Fehler bei der Zuweisung von Zeichensätzen - etwa durch fehlerhafte Einträge im DVS-Katalog - zu korrigieren. Bei Unicode-Zeichensätzen wird die Anweisung mit der Meldung EDT5494 abgewiesen.

Wird die Arbeitsdatei umcodiert und enthält sie Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @CODENAME-Anweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben. Bei Angabe von GLOBAL werden in diesem Fall bereits erfolgreich umcodierte Arbeitsdateien nicht zurück gewandelt.

Wenn einer Arbeitsdatei oder einer Zeichenfolgevariablen mit @CODENAME ein Zeichensatz zugewiesen wurde, gilt diese Zuweisung bis sie explizit durch eine erneute @CODENAME-Anweisung geändert wurde oder bis die Arbeitsdatei (z.B. mit @DELETE, Format 2) bzw. die Zeichenfolgevariable vollständig gelöscht wurde. Das bedeutet, dass alle Daten, die in die Arbeitsdatei bzw. Zeichenfolgevariable kopiert oder eingelesen werden, in diesen Zeichensatz umcodiert werden (siehe Abschnitt „Zeichensätze in Arbeitsdateien“ auf Seite 55).

Wenn in Arbeitsdatei 0 eine ISAM-Datei mit @OPEN (Format 2) real geöffnet ist, wird jede @CODENAME-Anweisung, die sich explizit oder implizit für diese Arbeitsdatei auswirkt, mit der Meldung EDT5452 abgewiesen.

Für eine geöffnete existierende Datei, die im Katalog den Zeichensatz *NONE eingetragen hat, bewirkt ein Aufruf der @CODENAME-Anweisung, dass beim Zurückschreiben durch @CLOSE oder @WRITE immer der beim Schreiben verwendete Zeichensatz explizit im Katalog eingetragen wird.

9.19 @CODENAME (Format 2) – Kommunikations-Zeichensatz einstellen

Mit Format 2 der Anweisung @CODENAME kann der vom EDT im Dialogbetrieb verwendete Kommunikations-Zeichensatz eingestellt werden. Die bei @CODENAME getroffene Festlegung hat Vorrang vor der impliziten Wahl eines Zeichensatzes durch den EDT, z.B. aufgrund der verwendeten Datensichtstation.

Operation	Operanden	F-Modus, L-Modus
@CODENAME	[{ name / ***A**UTO } ,**TERM**INAL]	

name — Name des einzustellenden Zeichensatzes. Der Zeichensatzname muss in XHCS bekannt sein. Der angegebene Zeichensatz muss von der verwendeten Datensichtstation unterstützt werden, andernfalls wird die Anweisung mit der Meldung `EDT5487` abgewiesen.

Die explizite Angabe eines Kommunikations-Zeichensatzes setzt gleichzeitig dessen automatische Wahl durch den EDT außer Kraft (siehe Abschnitt „Kommunikationszeichensatz“ auf Seite 54).

*AUTO — Stellt die automatische Wahl des Kommunikations-Zeichensatzes durch den EDT ein (siehe Abschnitt „Kommunikationszeichensatz“ auf Seite 54).

Wird kein Operand angegeben, wird als Kommunikationszeichensatz der mit `/MODIFY-TERMINAL-OPTIONS` eingestellte Zeichensatz verwendet. Dies ist auch die Anfangseinstellung beim Start des EDT.

Es wird der angegebene Zeichensatz als Kommunikations-Zeichensatz für den Datenaustausch mit der Datensichtstation eingestellt, der Operand beeinflusst nicht direkt die Codierung in den Arbeitsdateien (siehe Abschnitt „Zeichensätze“ auf Seite 48). Allerdings erhalten leere Arbeitsdateien ohne Zeichensatz, die erstmals durch direkte Terminal-Eingaben des Benutzers mit Daten gefüllt werden, dann zunächst diesen Zeichensatz. Insbesondere wenn die automatische Wahl des Kommunikationszeichensatzes eingestellt ist und eine moderne Unicode-fähige Emulation vorliegt, erhalten so gefüllte Arbeitsdateien immer den Zeichensatz `UTFE`, was nur durch vorherige explizite Zuweisung eines Arbeitsdatei-Zeichensatzes verhindert werden kann.

Im Stapelbetrieb wird die Anweisung ignoriert.

9.20 @COLUMN – Text einfügen und Leerzeichen am Zeilenende löschen

Die @COLUMN-Anweisung verändert den Inhalt von bestehenden Arbeitsdateizeilen bzw. von Zeichenfolgevariablen.

Im ersten Schritt wird ab der angegebenen Spalte Text eingefügt bzw. vorhandener Text ersetzt. Anschließend werden von rechts nach links beginnend mit dem letzten Zeichen der Arbeitsdateizeile bzw. Zeichenfolgevariablen alle Leerzeichen (bis zum ersten Nicht-Leerzeichen) gelöscht. Eine Arbeitsdateizeile, die nur aus Leerzeichen besteht, bleibt als Leerzeile in der Arbeitsdatei erhalten. Eine Zeichenfolgevariable, die nur Leerzeichen enthält, wird durch das Löschen zu einer leeren Zeichenfolgevariablen.

Operation	**Operanden**	**F-Modus, L-Modus**
@COLUMN	col **ON** { lines / svars } [,...] [{ **C**HANGE / **I**NSERT }] [:] string	

col — Spalte, ab der Text ersetzt oder eingefügt werden soll.

lines — Einer oder mehrere Zeilenbereiche, in denen Text eingefügt bzw. ersetzt werden soll. Es werden nur existierende Zeilen bearbeitet.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen Text eingefügt bzw. ersetzt werden soll.

CHANGE — Die mit `string` angegebene Zeichenfolge ersetzt ab der Spalte `col` den bestehenden Text (Standardwert).

INSERT — Die mit `string` angegebene Zeichenfolge wird ab der Spalte `col` eingefügt.

: — Ist zwingend anzugeben, wenn weder `CHANGE` noch `INSERT` angegeben ist, um die Bereichsangabe in eindeutiger Weise von `string` zu trennen.

string — Zeichenfolge, die in jeder Zeile jedes angegebenen Zeilenbereiches ab der angegebenen Spalte eingefügt wird bzw. vorhandenen Text ersetzt. Die Angabe einer leeren Zeichenfolge ist ebenfalls gestattet.

Die Zeichenfolge wird in den Zeichensatz der Arbeitsdatei bzw. der Zeichenfolgevariablen konvertiert. Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), anderenfalls wird die @COLUMN-Anweisung abgewiesen und die Fehlermeldung `EDT5453` ausgegeben.

Wenn die Spalte, ab der Text eingefügt werden soll, hinter dem bisherigen Zeilenende liegt, werden die dazwischen liegenden Spalten mit Leerzeichen aufgefüllt.

Das Ersetzen bzw. Einfügen findet nicht statt, wenn eine Arbeitsdateizeile bzw. eine Zeichenfolgevariable dadurch die maximale Länge von 32768 Zeichen überschreitet. Stattdessen wird in diesem Falle die Meldung `EDT5474` ausgegeben. Der EDT prüft nicht, ob die Zeilenlänge durch das Löschen von Leerzeichen am Zeilenende wieder einen zulässigen Wert annehmen würde.

Beim Auftreten von Fehlern während der Verarbeitung (`EDT5453` oder `EDT5474`) wird die Anweisung abgebrochen. Evtl. zu diesem Zeitpunkt schon erfolgreich modifizierte Zeilen und/oder Zeichenfolgevariablen bleiben modifiziert.

Wird die Anweisung mit [K2] unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Man kann diese Anweisung verwenden, um Leerzeichen an den Zeilenenden zu löschen, ohne die Zeilenlängen zu kennen. Wird in der @COLUMN-Anweisung als Ersetzungstext eine leere Zeichenfolge angegeben, so findet keine Textersetzung in den Zeilen statt. Der von rechts nach links verlaufende Löschvorgang bewirkt das Löschen der Leerzeichen an den Zeilenenden.

Beispiel

```
    1.00 126790<............................................................
    2.00 348<...............................................................
    3.00 ...................................................................



column 3 on 1:2 ...........................................0001.00:00001(01)
```

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'·····································0001.00:00001(01)
```

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 ··································································
```

9.21 @COMPARE (Format 1) – Vergleichen zweier Arbeitsdateien

Dieses Format der Anweisung @COMPARE bewirkt, dass der EDT zwei Arbeitsdateien ganz oder teilweise miteinander vergleicht. Das Ergebnis des Vergleichs kann wahlweise in eine Arbeitsdatei, auf SYSOUT oder auf SYSLST ausgegeben werden.

Operation	Operanden	F-Modus / L-Modus
@COMPARE	[procnr1] :lines1 **W**ITH [procnr2] :lines2 [,[int1] [(int2)] [**L**IST [line [(inc)]]]]	

procnr1, procnr2
: Nummern der miteinander zu vergleichenden Arbeitsdateien (0..22). Ein Vergleich von Bereichen aus der gleichen Arbeitsdatei (procnr1 gleich procnr2) ist gestattet. Ist eine der Vergleichsdateien die Arbeitsdatei 0 und ist in ihr eine Datei durch @OPEN, Format 2 real geöffnet, so wird @COMPARE mit der Fehlermeldung EDT4935 abgewiesen. Wird procnr1 oder procnr2 nicht angegeben, wird für den jeweiligen Operanden der Wert der aktuellen Arbeitsdatei verwendet.

lines1, lines2
: Zeilenbereiche, die miteinander verglichen werden sollen. Der Operand lines1 definiert den Zeilenbereich in der ersten Arbeitsdatei (procnr1). Der Operand lines2 definiert den Zeilenbereich in der zweiten Arbeitsdatei (procnr2). Beide Zeilenbereiche müssen nicht leer sein, sonst wird die Anweisung mit dem Fehler EDT4932 abgewiesen.

int1, int2
: Über int1 und int2 kann beeinflusst werden, wie tolerant sich der EDT im Falle von ungleichen Zeilen verhält. Findet der EDT innerhalb von int1 Zeilen nicht mindestens int2 aufeinander folgende Zeilen, die in beiden Dateien gleich sind, bricht er den Vergleich ab.

  Mit int2 wird zusätzlich festgelegt, wie viel aufeinander folgende Zeilen aus der einen Arbeitsdatei mit der entsprechenden Anzahl aufeinander folgender Zeilen aus der anderen Arbeitsdatei mindestens übereinstimmen müssen, damit der EDT die aus diesen Zeilen bestehenden Bereiche als gleich betrachtet.

  Für int1 und int2 gilt: int2 ≤ int1 ≤ 65535. Der Standardwert für int1 ist 10, für int2 ist er 1.

LIST — Bestimmt, wohin der EDT das Ergebnis des Vergleichs ausgibt.

Wird `LIST` ohne line angegeben, gibt der EDT das Ergebnis des Vergleichs auf `SYSLST` aus. Dabei gibt der EDT von jeder Zeile, für die er keine Übereinstimmung feststellt, außer der Zeilennummer auch die ersten 51 Zeichen des Zeileninhalts aus.

Wird `LIST` mit line angegeben, schreibt der EDT das Ergebnis des Vergleichs in die aktuelle Arbeitsdatei, falls diese nicht eine der Vergleichsdateien ist. Andernfalls wird die @COMPARE-Anweisung mit der Meldung `EDT4909` abgewiesen. Die Zeilennummernvergabe kann über die Operanden `line` und `inc` beeinflusst werden (siehe unten). Es werden lediglich die Nummern der Zeilen ausgegeben, für die keine Übereinstimmung vorliegt. Die Ausgabe des Zeileninhalts unterbleibt.

Wird `LIST` nicht angegeben, gibt der EDT das Ergebnis im Dialogbetrieb nach `SYSOUT` und im Stapelbetrieb nach `SYSLST` aus. Das Ausgabeformat ist das gleiche wie bei Ausgabe in die aktuelle Arbeitsdatei.

line — Die Nummer der Zeile in der aktuellen Arbeitsdatei, 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 `SYSOUT` benutzt.

inc — Schrittweite, aus der die auf line folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch line gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

Der EDT codiert vor dem Vergleich jede Zeile intern nach `UTF16` um und vergleicht die entstandenen Zeilen als Bytefolge. Die Zeilen sind gleich, wenn sowohl Zeileninhalt wie Zeilenlänge dieser Bytefolge gleich sind. Die Zeilennummern werden beim Datenvergleich nicht berücksichtigt. Wenn beide Arbeitsdateien im gleichen Zeichensatz vorliegen, ist dieses Verfahren äquivalent zu einem byteweisen Vergleich der Originalzeilen.

Die Ausgabe des Vergleichsergebnisses muss ggf. in einen geeigneten Zeichensatz umcodiert werden. Bei Ausgabe nach `SYSOUT` oder `SYSLST` ist dies der Zeichensatz, der für `SYSOUT` bzw. `SYSLST` eingestellt ist. Bei Ausgabe in die aktuelle Arbeitsdatei ist dies der für diese Arbeitsdatei eingestellte Zeichensatz. Wenn für die aktuelle Arbeitsdatei kein Zeichensatz eingestellt ist, erfolgt die Ausgabe im Zeichensatz der verglichenen Arbeitsdateien. Haben diese unterschiedliche Zeichensätze, erfolgt die Ausgabe im Zeichensatz `UTFE`.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Der EDT beginnt den Vergleich am Anfang der angegebenen Zeilenbereiche. Wenn der EDT ein ungleiches Zeilenpaar findet, versucht er durch vorwärts lesen innerhalb beider Dateien den nächsten Block von `int2` gleichen Zeilen zu finden.

Dabei liest der EDT in jeder Datei maximal `int1` Zeilen. Findet der EDT innerhalb dieses Bereichs `int2` aufeinander folgende gleiche Zeilen, richtet er für den weiteren Vergleich die beiden Dateien an diesen Zeilenpaaren aus, andernfalls bricht er den Vergleich ab.

Durch geeignete Wahl von `int2` kann zudem erreicht werden, dass beim Vergleich mehrere aufeinander folgende Zeilen als Einheit betrachtet werden. Dies kann etwa für Adressdateien nützlich sein, wenn eine Adressangabe aus mehreren Zeilen besteht (siehe Beispiel 2).

Wird für `int2` der gleiche Wert wie für `int1` gewählt, kann der EDT in den beiden Dateien keine übereinstimmenden Zeilenbereiche finden, wenn mindestens ein Zeilenpaar nicht identisch ist.

Die Ergebnisse des Vergleichs teilt der EDT in Form von kommentierten Listen von Zeilennummern mit.

Dabei wird die Gleichheit von Zeilen bzw. von Zeilenbereichen nicht gesondert ausgewiesen. Die Bereiche, für die in der jeweils anderen Datei ein übereinstimmender Bereich gefunden wurde, werden vom EDT miteinander identifiziert (sofern sie mindestens `int2` Zeilen umfassen), bilden also sozusagen Paare übereinstimmender Bereiche. Diese Paarbildung ist, wenn man leere Bereiche zulässt, auch für die nicht übereinstimmenden Bereiche möglich, denn zwei benachbarte Paare übereinstimmender Bereiche werden notwendigerweise durch genau ein Paar nicht übereinstimmender Bereiche getrennt, wobei einer der trennenden Bereiche leer sein kann.

Der EDT generiert eine kommentierte Liste für jedes Paar nicht übereinstimmender Bereiche in Abhängigkeit von deren Größe und Lage:

Wenn in einem Paar nicht übereinstimmender Bereiche beide nicht leer sind, aber höchstens `int1 - int2` Zeilen umfassen, werden die Bereiche als unterschiedlich ausgewiesen:

```
NON-MATCHING LINES
  ln       ln
   .        .
   .        .
   .       ln
  ln
```

Wenn in einem Paar nicht übereinstimmender Bereiche beide nicht leer sind und ein Bereich mehr als `int1 - int2` Zeilen umfasst, wird die Ausgabe der obigen Liste entsprechend verkürzt und der Vergleich mit der Meldung

```
NOTHING SEEMS TO MATCH
```

abgebrochen.

Wenn in einem Paar nicht übereinstimmender Bereiche einer leer ist und der andere höchstens `int1 - int2` Zeilen umfasst, wird der nicht leere Bereich als zusätzlich vorhandener Bereich ausgewiesen:

```
EXTRA LINES IN 1ST FILE
  ln
   .
   .
   .
  ln
```

Falls der nicht leere Bereich in der zweiten Datei liegt, erfolgt die Ausgabe analog mit geänderter Überschrift:

```
EXTRA LINES IN 2ND FILE
```

Wenn der nicht leere Bereich mehr als `int1 - int2` Zeilen umfasst und nicht am Ende des zu vergleichenden Zeilenbereichs liegt, wird die Ausgabe der obigen Liste entsprechend verkürzt und der Vergleich mit der Meldung

```
NOTHING SEEMS TO MATCH
```

abgebrochen. Liegt der nicht leere Bereich von mehr als int1 - int2 Zeilen am Ende des zu vergleichenden Zeilenbereichs wird stattdessen die Meldung

```
REACHED LIMIT ON 1ST FILE
```

bzw.

```
REACHED LIMIT ON 2ND FILE
```

ausgegeben, wobei die Bezeichnung `1ST` bzw. `2ND` sich auf die Datei bezieht, die den leeren Bereich enthält.

Werden in beiden Dateien bis zum Ende des jeweiligen zu vergleichenden Zeilenbereichs keine nicht übereinstimmenden Bereiche von mehr als `int1 - int2` Zeilen gefunden, wird zusätzlich der Hinweis

```
REACHED LIMIT ON BOTH FILES
```

ausgegeben, falls am Ende der zu vergleichenden Zeilenbereiche ein Paar nicht übereinstimmender Bereiche liegt. Liegt hingegen am Ende der zu vergleichenden Zeilenbereiche ein Paar übereinstimmender Bereiche wird die Meldung

```
REACHED LIMIT ON BOTH FILES AT SAME TIME
```

ausgegeben. Wenn die beiden zu vergleichenden Zeilenbereiche vollständig übereinstimmen ist dies die einzige ausgegebene Meldung.

Beispiel 1

```
   1.     @PROC 1
   1.     @COPY FILE=PROC-DATEI.1 ---------------------------------------- (1)
   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.     @COPY FILE=PROC-DATEI.2 ---------------------------------------- (2)
   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) ------------------------------- (3)
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) ------------------------------- (4)
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) ------------------------------- (5)
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.
```

(1) Die SAM-Datei `PROC-DATEI.1` wird in die Arbeitsdatei 1 eingelesen.

(2) Die SAM-Datei `PROC-DATEI.2` wird in die Arbeitsdatei 2 eingelesen.

(3) Lassen sich bei der Betrachtung von jeweils 5 Zeilen in den beiden Dateien nicht mindestens 2 aufeinander folgende gleiche Zeilenpaare finden, soll der Vergleich abgebrochen werden. Der Vergleich wird bis zum Ende beider Dateien durchgeführt.

(4) Das unter (3) gegebene @COMPARE wird leicht modifiziert noch einmal gegeben. Es müssen jetzt mindestens 3 aufeinander folgende gleiche Zeilenpaare gefunden werden. Diesmal bricht der EDT den Vergleich ab.

(5) Das unter (4) 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.

Beispiel 2

Es wird vorausgesetzt, dass die Arbeitsdateien schon mit den entsprechenden Daten gefüllt sind.

```
   5.     @PROC 1
  21.     @PRINT ------------------------------------------------------ (1)
   1.0000 Donald Duck
   2.0000 Am Dorfteich 11
   3.0000 12345 Entenhausen
   4.0000 -
   5.0000 Dagobert Duck
   6.0000 Schlossallee 1a
   7.0000 12345 Entenhausen
   8.0000 -
   9.0000 Daisy Duck
  10.0000 Am Dorfteich 12
  11.0000 12345 Entenhausen
  12.0000 -
  13.0000 Gustav Gans
  14.0000 Im Wiesengrund 10
  15.0000 12345 Entenhausen
  16.0000 -
  17.0000 Gustav Gans
  18.0000 Schmale Gasse 7
  19.0000 12345 Entenhausen
  20.0000 -
  21.     @END
   5.     @PRINT ------------------------------------------------------ (2)
```

```
   1.0000 Gustav Gans
   2.0000 Im Wiesengrund 10
   3.0000 12345 Entenhausen
   4.0000 -
   5.     @COMPARE 0:& WITH 1:&, 9999(4) ------------------------------ (3)
EXTRA LINES IN 2ND FILE
                1.0000
                2.0000
                3.0000
                4.0000
                5.0000
                6.0000
                7.0000
                8.0000
                9.0000
               10.0000
               11.0000
               12.0000
EXTRA LINES IN 2ND FILE
               17.0000
               18.0000
               19.0000
               20.0000
REACHED LIMIT ON BOTH FILES
    5.    @ON 2 CHANGE '10' TO '13'
    5.    @COMPARE 0:& WITH 1:&, 9999(4) ------------------------------ (4)
NON-MATCHING LINES
   1.0000      1.0000
   2.0000      2.0000
   3.0000      3.0000
   4.0000      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
               17.0000
               18.0000
               19.0000
               20.0000
```

```
REACHED LIMIT ON BOTH FILES
    5.
```

(1) Arbeitsdatei 1 wird ausgegeben. Es handelt sich um eine Adressendatei.

(2) Nach Rückkehr aus Arbeitsdatei 1 wird Arbeitsdatei 0 ausgegeben. Sie enthält eine Adresse (4 Zeilen), nach der in der Arbeitsdatei 1 gesucht werden soll.

(3) Die Suche wird durch einen Vergleich der beiden Arbeitsdateien realisiert. Durch Wahl von `9999` für `int1` (siehe Operandenbeschreibung) wird ein Abbruch des Vergleichs verhindert. Durch Wahl von `4` für `int2` wird erreicht, dass nur Bereiche mit `4` übereinstimmenden Zeilen als gleich gewertet werden. Die Ausgabe weist `EXTRA LINES` vor und nach einem übereinstimmenden Bereich aus, also ist die Adresse in Arbeitsdatei 1 enthalten.

(4) Nach Modifikation der Hausnummer wird erneut gesucht. Da keine `4` aufeinander folgenden übereinstimmenden Zeilen gefunden wurden, werden nur `NON-MATCHING LINES` ausgewiesen. Die Adresse ist also nicht enthalten.

Vergleicht man mit @COMPARE 0:& WITH 1:&, 9999(1) statt mit @COMPARE 0:& WITH 1:&, 9999(4) erhält man sowohl für übereinstimmende Hausnummer wie für nicht übereinstimmende Hausnummer die gleiche Ausgabe, weil der EDT sich bei der ersten übereinstimmenden Zeile (`12345 Entenhausen`) neu synchronisiert.

9.22 @COMPARE (Format 2) – Vergleichen zweier Arbeitsdateien zeilenweise

Mit Format 2 der Anweisung @COMPARE können die Inhalte zweier Arbeitsdateien zeilenweise verglichen werden. Das Vergleichsergebnis legt der EDT in einer Arbeitsdatei ab. Diese wird vor dem Ablegen des Ergebnisses gelöscht. Wahlweise ist auch eine Ausgabe des Ergebnisses nach SYSLST, im L-Modus auch nach SYSOUT möglich.

Operation	Operanden	F-Modus / L-Modus
@COMPARE	{ [procnr1] **W**ITH procnr2 / procnr1 } [**L**IST [procnr3]] [,procnr4]	

procnr1 Nummer der Arbeitsdatei, die verglichen werden soll. Ist procnr1 nicht angegeben, so wird die aktuelle Arbeitsdatei mit procnr2 verglichen.

procnr2 Nummer der Arbeitsdatei, mit der verglichen wird. Ist procnr2 nicht angegeben, so wird procnr1 mit der aktuellen Arbeitsdatei verglichen.

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 L-Modus desDialogbetriebs nach SYSOUT ausgegeben, im Stapelbetrieb nach SYSLST ausgegeben und im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei 9 eine Datei geöffnet ist, wird die Meldung EDT5189 ausgegeben und die Anweisung nicht ausgeführt.

procnr3 Arbeitsdatei, in der das detaillierte Vergleichsergebnis abgelegt wird, falls ein solches erstellt wird (siehe unten). Die Vergabe der Zeilennummern erfolgt nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Die Arbeitsdatei wird vor der Verwendung gelöscht. War eine Datei in dieser Arbeitsdatei geöffnet, wird diese implizit und ohne zurück zu schreiben geschlossen (@CLOSE NOWRITE).

procnr4 Die Angabe einer Arbeitsdatei als Hilfsdatei wird nur noch aus Kompatibilitätsgründen erlaubt. Eine hier spezifizierte Arbeitsdatei wird vom EDT nicht mehr verwendet.

Die Arbeitsdateien procnr1 und procnr2 müssen voneinander verschieden sein, andernfalls wird die @COMPARE-Anweisung mit der Meldung EDT5499 abgewiesen. Die Arbeitsdatei procnr3 kann mit procnr1 oder procnr2 identisch sein, man bekommt dann aber kein detailliertes Ergebnis (siehe unten).

Sind alle zu vergleichenden Zeilen gleich bzw. ungleich, wird nur die Meldung EDT0291 bzw. EDT0290 ausgegeben. Ein detailliertes Vergleichsergebnis wird in diesem Fall nicht ausgegeben.

Muss ein detailliertes Ergebnis nach procnr3 ausgegeben werden, wird nach Ende des Vergleichs die Meldung EDT0297 ausgegeben. Ist in diesem Fall eine der beiden Vergleichsdateien die Arbeitsdatei, in der das Ergebnis stehen soll, wird die Meldung EDT5350 ausgegeben, die Ausgabe des detaillierten Vergleichsergebnisses unterbleibt dann.

Ist eine der Vergleichsdateien die Arbeitsdatei 0, darf keine ISAM-Datei durch @OPEN (Format 2) real geöffnet sein, andernfalls wird die @COMPARE-Anweisung mit der Meldung EDT4935 abgewiesen.

Um in EDT-Prozeduren das Vergleichsergebnis abfragen zu können, wird zusätzlich zu den Meldungen EDT0290 und EDT0297 der EDT-Fehlerschalter gesetzt der mit der @IF-Anweisung abgefragt werden kann (siehe @IF-Anweisung):

	EDT-Fehlerschalter	**Arbeitsdatei procnr3**
EDT0291	nicht gesetzt	leer
EDT0290	gesetzt	leer
EDT0297	gesetzt	nicht leer

Wenn alle aufgeführten Fälle unterschieden werden sollen, muss vor dem Vergleichen mit @COMPARE sowohl der EDT-Fehlerschalter mit @RESET zurückgesetzt als auch die Arbeitsdatei procnr3 gelöscht werden.

Der EDT codiert vor dem Vergleich jede Zeile intern nach UTF16 um und vergleicht die entstandenen Zeilen als Bytefolge. Die Zeilen sind gleich, wenn sowohl Zeileninhalt wie Zeilenlänge dieser Bytefolge gleich sind. Die Zeilennummern werden beim Datenvergleich nicht berücksichtigt. Wenn beide Arbeitsdateien im gleichen Zeichensatz vorliegen, ist dieses Verfahren äquivalent zu einem byteweisen Vergleich der Originalzeilen.

Die Ausgabe des Vergleichsergebnisses muss ggf. in einen geeigneten Zeichensatz umcodiert werden. Bei Ausgabe nach SYSOUT oder SYSLST ist dies der Zeichensatz, der für SYSOUT bzw. SYSLST eingestellt ist. Bei Ausgabe in eine Arbeitsdatei ist dies der Zeichensatz der verglichenen Arbeitsdateien. Haben diese unterschiedliche Zeichensätze, erfolgt die Ausgabe im Zeichensatz UTFE.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Das Format der Ausgabe ist für die Ausgabe in eine Arbeitsdatei bzw. nach SYSLST oder SYSOUT identisch:

```
LINE#( 1)          FILENAME: DAT.270104
         LINE#( 0) FILENAME: VERGL.1
```

Es wird eine Überschriftenzeile ausgegeben, in der die Spalten, die den verglichenen Arbeitsdateien zugeordnet werden, mit LINE#... und der Nummer der jeweiligen Arbeitsdatei (in Klammern) gekennzeichnet sind. Zusätzlich wird, sofern vorhanden, der Name einer geöffneten Datei bzw. eines geöffneten Bibliothekselements oder ein lokaler @FILE-Eintrag ausgegeben.

```
0007.10  KUNDE-100     SORT
0007.20  KUNDE-200     PERCON
        0007.30  KUNDE-700     FDDRL

0010.00          $KUNDE-900    LMS
        0010.00  $KUNDE-900    LMSCONV
```

Für Zeilen, die nur in einer Arbeitsdatei vorkommen, werden die Zeilennummern und der Inhalt der Sätze (ggf. um 17 Zeichen gekürzt) ausgegeben. Dabei gibt die Stellung der Zeilennummer in Spalte 1 oder in Spalte 2 unter der Überschrift LINE#... an, in welcher der beiden Arbeitsdateien der Satz steht. Dies gilt sinngemäß auch für Sätze unterschiedlichen Inhalts. Diese kommen mit dem einen Inhalt nur in der ersten Arbeitsdatei vor, mit dem anderen Inhalt nur in der zweiten Arbeitsdatei und stehen dabei für gewöhnlich untereinander.

```
0008.00=0010.00
0018.00=0020.00
```

Für Zeilen gleichen Inhalts werden die identifizierten Zeilennummern in der Form 0001.00=0006.00 ausgegeben. Sind mehrere aufeinander folgende Sätze gleich (Bereich gleicher Sätze), wird nur das erste und letzte Zeilennummernpaar des Bereiches angegeben (Näheres siehe Beispiel).

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 ·····························································
```

```
1·····························································0001.00:00001(02)
```

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 ·····························································
```

```
@compare 2 with 1 list 3;  3···································0001.00:00001(01)
```

Arbeitsdatei 2 wird mit Arbeisdatei 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
·····························································0000.10:00001(03)
```

Das Vergleichsergebnis der Arbeitsdateien 1 und 2 ist in der Arbeitsdatei 3 abgelegt.

9.23 @CONTINUE – Leere Anweisung

Die Anweisung @CONTINUE führt keine Aktion aus. Sie wird verwendet, um in EDT-Prozeduren eine Zeile zu erzeugen, die über @GOTO angesprungen werden kann. Sie kann auch zur Kommentierung von EDT-Prozeduren verwendet werden. Die Anweisung @NOTE bietet die gleiche Funktionalität wie @CONTINUE.

Operation	Operanden	L-Modus
@CONTINUE	[comment]	

comment Der Operand `comment` kann beliebigen Text als Kommentar enthalten.

Neben der Kommentierung ist eine häufige Anwendung dieser Anweisung die Definition einer letzten Zeile innerhalb einer EDT-Prozedur, die in einer @GOTO-Anweisung oder in einer @IF-Anweisung als Sprungziel angegeben werden kann. Diese Konstruktion wird benötigt, wenn eine EDT-Prozedur in einer äußeren Schleife mit einem Schleifenzähler aufgerufen wird (z.B. @DO 5,!=%,$), und ein @IF ... RETURN zu einem nicht gewollten Abbruch der äußeren Schleife führen würde. Stattdessen verzweigt man an das Ende der Prozedur, 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' ---------------------- (1)
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 ------------------------------------------------- (2)
2.01   @END
6.     @DO 1,!=1,$ ------------------------------------------------- (3)
1.0000 MIT DEM EDT
2.0000 KANN MAN
4.0000 PROZEDUR UM PROZEDUR
6.
```

(1) Hier wird @CONTINUE zur Kommentierung verwendet.

(2) Hier wird @CONTINUE benötigt, da es eine letzte Zeile in der Prozedur geben muss, die angesprungen werden kann.

(3) Über @DO mit Schleifenzähler wird die in Arbeitsdatei 1 befindliche Prozedur ausgeführt und dabei auf Arbeitsdatei 0 angewendet.

9.24 @CONVERT – Konvertieren von Klein- bzw. Großbuchstaben

Mit der Anweisung @CONVERT können in Zeilenbereichen Klein- in Großbuchstaben oder Groß- in Kleinbuchstaben konvertiert werden.

Operation	Operanden	F-Modus, L-Modus
@CONVERT	[[{ lines / svars } [,...]] **T**O =] { **UP**PER / **LOW**ER }	

lines Einer oder mehrere Zeilenbereiche, in denen konvertiert werden soll. Es werden nur existierende Zeilen bearbeitet.

svars Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen konvertiert werden soll.

UPPER Alle Kleinbuchstaben werden in Großbuchstaben konvertiert.

LOWER Alle Großbuchstaben werden in Kleinbuchstaben konvertiert.

Es muss entweder `UPPER` oder `LOWER` angegeben werden. Wird weder `lines` noch `svars` angegeben, wird in der gesamten aktuellen Arbeitsdatei konvertiert.

Die Information, welche Zeichen im Zeichensatz der aktuellen Arbeitsdatei bzw. der eichenfolgevariablen Klein- bzw. Großbuchstaben sind, wird bei XHCS abgefragt. Alle anderen Zeichen bleiben unverändert.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

9.25 @COPY (Format 1) – Einlesen einer Datei

Mit @COPY (Format 1) wird eine existierende Datei komplett in die aktuelle Arbeitsdatei eingelesen. Die Arbeitsdatei braucht dabei nicht leer zu sein. Die Arbeitsdateiposition, an der die Datei eingefügt werden soll, kann angegeben werden. Nach dem Einlesen wird die Datei wieder geschlossen.

Wenn in diesem Abschnitt von Datei die Rede ist, dann kann dies eine SAM-Datei, eine ISAM-Datei, ein Bibliothekselement oder eine POSIX-Datei sein.

Operation	**Operanden**	**F-Modus, L-Modus**
@COPY	**L**IBRARY=path1 ([**E**LEMENT=] elname [(vers)][,eltype]) **E**LEMENT=elname [(vers)][,eltype] **F**ILE = { path2 / *linkname } [,**K**EY = { **L**INENUMBER / **D**ATA / **I**GNORE }] **P**OSIX - FILE = xpath [,**C**ODE = name] [[,] { **B**EFORE / **A**FTER } line]	

LIBRARY= Ein Bibliothekselement soll eingelesen werden. Dieses wird durch explizite Angabe des Bibliotheksnamens und der Elementbezeichnung bestimmt.

path1 Name der Bibliothek.

elname Name des Elements.

vers Version des gewünschten Elements (siehe Handbuch LMS [14]). Wird `vers` nicht angegeben oder wird `*STD` angegeben, wird die höchste vorhandene Version des Elementes gewählt.

eltype Typ des Elements. Zulässige Typangaben sind `S`,`M`, `P`, `J`, `D`, `X`, `*STD` und freie Typnamen mit entsprechendem Basistyp. Wird `eltype` nicht angegeben, wird der mit @PAR ELEMENT-TYPE voreingestellte Wert verwendet. Die zulässigen Elementtypen und deren Bedeutung sind im Kapitel „Dateibearbeitung“ auf Seite 137 beschrieben.

ELEMENT= Ein Bibliothekselement soll eingelesen werden. Dieses wird durch die Elementbezeichnung ohne Angabe des Bibliotheknamens bestimmt. Es wird implizit die mit @PAR LIBRARY voreingestellte Bibliothek verwendet (sofern @PAR LIBRARY spezifiziert wurde, andernfalls wird die Fehlermeldung EDT5181 ausgegeben).

Die Operanden elname, vers und eltype haben die gleiche Bedeutung wie bei expliziter Angabe der Bibliothek (siehe oben).

FILE= Eine BS2000-Datei soll eingelesen werden.

path2 Name der BS2000-Datei (voll qualifizierter Dateiname), die eingelesen werden soll.

*linkname Dateikettungsname der BS2000-Datei, die eingelesen werden soll. Der Dateiname und die Dateiattribute sind in der Task File Table abgelegt. Der Dateikettungsname darf nicht mit den Spezial-Dateinamen *BY-PROGRAM vereinbart worden sein. Dies führt zum Fehler EDT4923. Ist der Dateikettungsname nicht definiert, wird die Anweisung mit dem Fehler EDT5480 abgewiesen.
Ist der Dateikettungsname mit dem Spezial-Dateinamen *DUMMY vereinbart worden, wird dies wie eine existierende leere Datei behandelt.

KEY= Legt für ISAM-Dateien den Ort fest, an dem der ISAM-Schlüssel in der Arbeitsdatei abgelegt wird. Für andere Dateitypen wird der Operand ignoriert.

LINENUMBER
Der ISAM-Schlüssel wird als Zeilennummer in der Arbeitsdatei abgelegt. Bereits existierende Zeilen mit den gleichen Zeilennummern werden überschrieben. Die Operanden BEFORE oder AFTER dürfen nicht angegeben werden. Kann der ISAM-Schlüssel nicht als Zeilennummer interpretiert werden, weil die Schlüsselposition vom Standard abweicht, die Schlüssellänge zu groß ist oder die Schlüssel nicht numerisch sind, wird die Meldung EDT5459 ausgegeben und die Datei nicht eingelesen.

DATA Der ISAM-Schlüssel wird Bestandteil des Datenbereichs der Arbeitsdatei.

IGNORE Der ISAM-Schlüssel wird nicht in der Arbeitsdatei abgelegt. Dies ist der Standardwert. Weicht die Schlüsselposition vom Standard ab, wird die Meldung EDT5466 ausgegeben und die Datei nicht eingelesen.

POSIX-FILE= Es soll eine POSIX-Datei eingelesen werden.

xpath Pfadname der POSIX-Datei, die in die aktuelle Arbeitsdatei eingelesen werden soll.
Der Operand xpath kann auch als Zeichenfolgevariable angegeben werden. Er *muss* als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

CODE= Es wird festgelegt, welcher Zeichensatz für die POSIX-Datei angenommen wird. Da POSIX-Dateien innerhalb des POSIX-Dateisystems kein Zeichensatz zugeordnet werden kann, ist hier eine Angabe des Anwenders erforderlich.

Wird `CODE` nicht angegeben, wird für die Datei der mit @PAR CODE eingestellte Zeichensatz angenommen.

- name — Zeichensatz der einzulesenden POSIX-Datei. Als `name` muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).
- EBCDIC — Das Schlüsselwort `EBCDIC` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `EDF041` unterstützt.
- ISO — Das Schlüsselwort `ISO` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `ISO88591` unterstützt.

BEFORE Die Datei wird vor der angegebenen Arbeitsdateizeile eingefügt. Der Operand darf nicht gleichzeitig mit `KEY=LINENUMBER` angegeben werden.

AFTER Die Datei wird nach der angegebenen Arbeitsdateizeile eingefügt. Der Operand darf nicht gleichzeitig mit `KEY=LINENUMBER` angegeben werden.

line Zeilennummer, vor oder nach der eingefügt wird.

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar bzw. kann die Datei nicht erfolgreich eingelesen werden, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Wird weder `BEFORE` noch `AFTER` angegeben und ist der Operand `KEY` ungleich `LINENUMBER`, dann wird die Datei nach der letzten Zeile der aktuellen Arbeitsdatei eingefügt.

Die Zeilennummern der Arbeitsdatei werden beim Einlesen von ISAM-Dateien mit Operand `KEY=LINENUMBER` aus dem ISAM-Schlüssel der Datei gebildet, in allen anderen Fällen nach dem Verfahren Einfügen zwischen zwei Zeilen (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz `*NONE`, erhält die Arbeitsdatei den Zeichensatz der einzulesenden Datei. Ist dieser `*NONE`, erhält die Arbeitsdatei den Zeichensatz `EDF03IRV`.

Hat die Arbeitsdatei bereits einen Zeichensatz, dann werden die einzulesenden Sätze vom Zeichensatz der Datei in den Zeichensatz der Arbeitsdatei konvertiert. Enthält die einzulesende Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht eingelesen und die Fehlermeldung `EDT5453` ausgegeben.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird das Lesen mit der Meldung EDT5454 abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Beispiel

```
@COPY LIBRARY=MACLIB(ELEMENT=XYZ,M) AFTER 12.3
```

Das Element mit dem Namen XYZ, der höchsten vorhandenen Version und dem Elementtyp M (Makro) aus der Bibliothek MACLIB wird vollständig in die aktuelle Arbeitsdatei nach der Zeile 0012.3000 eingefügt.

```
@PAR LIBRARY=DATEN
@COPY ELEMENT=PERSONAL(@),D
```

Das Element mit dem Namen PERSONAL, der höchsten vorhandenen Version und dem Elementtyp D (Textdaten) aus der Bibliothek DATEN wird nach der letzten Zeile der aktuellen Arbeitsdatei eingefügt.

```
@COPY FILE=DATEI.ISAM,KEY=LINENUMBER
```

Die ISAM-Datei DATEI.ISAM wird in die Arbeitsdatei eingelesen. Die Zeilennummern der Arbeitsdatei werden aus dem ISAM-Schlüssel gebildet. Eventuell schon vorhandene Zeilen werden dabei überschrieben.

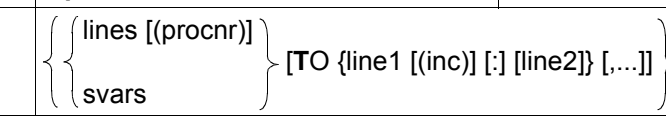
```
@COPY POSIX-FILE=/home/user1/test/data,CODE=UTF8
```

Die POSIX-Datei data in dem Verzeichnis /home/user1/test mit dem Zeichensatz UTF8 wird nach der letzten Zeile der aktuellen Arbeitsdatei eingefügt. Die Datei wird dabei in den Zeichensatz der Arbeitsdatei konvertiert.

9.26 @COPY (Format 2) – Kopieren von Zeilen oder Zeichenfolgevariablen

Mit der Anweisung @COPY werden Datensätze der aktuellen oder einer anderen Arbeitsdatei oder Inhalte von Zeichenfolgevariablen in die aktuelle Arbeitsdatei kopiert.

Der Zeilenbereich der Quell-Arbeitsdatei, der die zu kopierenden Datensätze enthält, bzw. der Bereich von Zeichenfolgevariablen wird der Einfachheit halber im Folgenden als Sendebereich bezeichnet. Der Zeilenbereich der aktuellen Arbeitsdatei, in den die Datensätze der Quell-Arbeitsdatei kopiert werden, wird als Empfangsbereich bezeichnet.

Operation	**Operanden**	**F-Modus, L-Modus**
@COPY		

lines — Zusammenhängender Zeilenbereich, der in die aktuelle Arbeitsdatei kopiert werden soll. Symbolische Zeilennummern in `lines` beziehen sich auf die Zeilennummern der aktuellen Arbeitsdatei, auch wenn die Zeilen aus einer anderen Arbeitsdatei kopiert werden.

procnr — Nummer der Quell-Arbeitsdatei, aus der die Zeilen kopiert werden (`0..22`). Ist `procnr` nicht angegeben, werden die Zeilen aus der aktuellen Arbeitsdatei kopiert. Eine aktive Arbeitsdatei darf nicht angegeben werden.
Wird kein `TO` Operand angegeben, darf `procnr` nicht die aktuelle Arbeitsdatei sein.

svars — Bereich von Zeichenfolgevariablen, deren Inhalte in die aktuelle Arbeitsdatei kopiert werden sollen.

TO... — Mit den auf `TO` folgenden Operanden werden der oder die Empfangsbereiche definiert. Ist kein Empfangsbereich angegeben, dann werden die Zeilennummern der Quell-Arbeitsdatei in die aktuelle Arbeitsdatei übernommen.
Ist die Quell-Arbeitsdatei die aktuelle Arbeitsdatei oder werden Zeichenfolgevariable kopiert, dann muss `TO...` angegeben werden. Ist in diesen Fällen kein Empfangsbereich angegeben, dann wird die @COPY-Anweisung mit der Fehlermeldung `EDT3218` abgewiesen.

line1 — Nummer der ersten Zeile des Empfangsbereiches.

inc — Schrittweite, aus der die auf `line1` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line1` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

: — Die Operanden `line1` und `line2` sollten durch : voneinander getrennt werden, falls `inc` nicht angegeben wird.

line2 — Legt die größtmögliche Zeilennummer des Empfangsbereiches fest, bis zu der ein Kopieren von Datensätzen erlaubt wird.

Es findet somit kein Kopieren in Zeilen der aktuellen Arbeitsdatei statt, deren Zeilennummer größer als `line2` ist. Dies gilt auch dann, wenn dadurch nicht alle Datensätze des Sendebereiches in den Empfangsbereich kopiert werden können.
Ist `line2` nicht angegeben, so wird durch die @COPY-Anweisung kein Maximalwert für die Zeilennummern des Empfangsbereiches festgelegt.

Mit der @COPY-Anweisung können mehrere Sendebereiche mit jeweils mehreren Empfangsbereichen durch Komma getrennt angegeben werden. Die Anzahl der Sende- und Empfangsbereiche ist nur durch die maximal erlaubte Länge einer EDT-Anweisung begrenzt.

Wenn sich der Sende- und der Empfangsbereich überlappen, wird der Sendebereich zeilenweise kopiert. Dies kann dazu führen, dass ein Datensatz zunächst in eine Zeile und anschließend wieder aus dieser Zeile kopiert wird, wenn diese sowohl im Sende- als auch im Empfangsbereich liegt. Auf diese Weise kann der Sendebereich bzw. Teile davon mehrfach in den Empfangsbereich kopiert werden.

Bereits in der aktuellen Arbeitsdatei existierende Zeilen mit gleichen Zeilennummern werden beim Kopieren überschrieben.

Wenn eine Zeile angelegt wird, deren Nummer größer als die bisherige höchste Zeilennummer ist, wird die aktuelle Zeilennummer verändert.

Ist die aktuelle Arbeitsdatei leer und hat sie den Zeichensatz `*NONE`, dann erhält sie beim Kopieren den Zeichensatz der Quell-Arbeitsdatei bzw. der ersten angegebenen Zeichenfolgevariablen. Hat die aktuelle Arbeitsdatei einen Zeichensatz, dann werden die zu kopierenden Zeilen bzw. die Inhalte der Zeichenfolgevariablen in den Zeichensatz der aktuellen Arbeitsdatei konvertiert. Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @COPY-Anweisung abgewiesen und die Fehlermeldung `EDT5453` ausgegeben.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Da es die obige Syntax erlaubt, den TO-Operanden wegzulassen, ist es nicht immer möglich, eindeutig zwischen einem Empfangsbereich und einem nächsten Sendebereich zu unterscheiden. In diesen Fällen entscheidet sich der EDT für die Interpretation als Empfangsbereich. So wird z.B. in der Eingabe

```
@COPY 2-3(1) TO 7,1(1)
```

die Angabe 1(1) als zweiter Empfangsbereich interpretiert (die 1 in Klammern wird als Schrittweite interpretiert), während die Angabe 1(0) an dieser Stelle als nächster Sendebereich interpretiert worden wäre (die 0 kann keine Schrittweite sein und wird als Arbeitsdateinummer interpretiert). Will der Anwender in diesem Beispiel die Interpretation als Sendebereich erzwingen, könnte er z.B.

```
@COPY 2-3(1) TO 7,1-1(1)
```

eingeben, dann gibt es keine Mehrdeutigkeiten.

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)···············0001.00:00001(00)
```

Mit den drei @COPY-Anweisungen sollen die Zeile 1 in die Zeile 7, die Zeile 2 in die Zeile 5 und der Zeilenbereich von Zeilennummer 1 bis 3 in den Zeilenbereich ab Zeile 30.1 mit der expliziten Schrittweite 5 kopiert werden.

```
   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 der @COPY-Anweisung 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................................................0001.00:00001(00)
```

Mit dieser Anweisung wird der Zeilenbereich von Zeilennummer 1 bis 2 in den Zeilenbereich 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 mit der impliziten Schrittweite 0.1 in die Zeile 1.6 kopiert. Entsprechend wird Zeile 1.6 in 1.7, ... , Zeile 1.9 in 2.0 (dabei wird der Inhalt der Zeile 2 überschrieben) und abschließend Zeile 2.0 in 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............................................0001.00:00001(00)
```

Der Zeilenbereich 3 bis 5 soll in den Zeilenbereich 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. Die beiden neu angelegten Zeilen liegen im angegebenen Sendebereich. Daher wird anschließend die Zeile 4.1 in die Zeile 4.3, die Zeile 4.2 in 4.4, ... , die Zeile 4.8 in 5.0 kopiert. Dabei wird der Inhalt der Zeile 5 überschrieben. Die Zeilen 4.9 und 5.0 werden nicht mehr kopiert, da die größtmögliche Zeilennummer des Empfangsbereiches 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 ·····························································
·····························0001.00:00001(00)
```

9.27 @CREATE (Format 1) – Erzeugen einer Zeile

Mit dem Format 1 der @CREATE-Anweisung wird eine Zeile mit dem angegebenen Inhalt erzeugt.

Operation	**Operanden**	**F-Modus, L-Modus**
@**CR**EATE	line [:] [string[,...]] [,**C**ODE=name]	

line — Die Zeilennummer in der aktuellen Arbeitsdatei, die neu erzeugt werden soll. Existiert diese Zeile bereits, dann wird sie vollständig überschrieben.

: — Muss angegeben werden, wenn `line` nicht eindeutig von `string` zu trennen ist.

string — Eine oder mehrere Zeichenfolgen, die in der angegebenen Reihenfolge verkettet und als Zeile eingefügt werden sollen.

name — Zeichensatz, der für die aktuelle Arbeitsdatei eingestellt werden soll, falls diese leer ist und den Zeichensatz `*NONE` hat.

Die in `string` angegebenen Zeichenfolgen werden im ersten Schritt miteinander verkettet. Haben alle beteiligten Zeichenfolgen denselben Zeichensatz, so erhält auch das Zwischenergebnis diesen Zeichensatz. Sind Zeichenfolgen mit unterschiedlichen Zeichensätzen beteiligt, so erhält das Zwischenergebnis den Zeichensatz `UTFE`.

Überschreitet das Zwischenergebnis nach dem Konvertieren die maximale Länge von 32768 Zeichen, wird auf die maximale Länge abgeschnitten und die Meldung `EDT2400` ausgegeben.

Ist der `CODE`-Operand nicht angegeben und ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz `*NONE`, so wird das Zwischenergebnis ohne Konvertieren in die Zeile eingefügt. Für die aktuelle Arbeitsdatei wird der Zeichensatz eingestellt, in dem das Zwischenergebnis vorliegt.

Ist der `CODE`-Operand angegeben und ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz `*NONE`, dann wird das Zwischenergebnis vor dem Einfügen in den Zeichensatz `name` konvertiert. Für die aktuelle Arbeitsdatei wird dann genau dieser Zeichensatz eingestellt.

Ist der `CODE`-Operand angegeben und hat die aktuelle Arbeitsdatei bereits einen Zeichensatz, der von diesem verschieden ist, dann wird die @CREATE-Anweisung nicht ausgeführt und die Meldung `EDT5458` ausgegeben.

Ist der `CODE`-Operand nicht angegeben und hat die aktuelle Arbeitsdatei bereits einen Zeichensatz, dann wird das Zwischenergebnis vor dem Einfügen in den Zeichensatz der Arbeitsdatei konvertiert.

Enthält die einzufügende Zeichenfolge Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @CREATE-Anweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Ist string nicht angegeben, dann wird die Zeile als Leerzeile (Zeile der Länge 0) angelegt. Für die aktuelle Arbeitsdatei wird der Zeichensatz EDF041 eingestellt, falls diese leer ist und den Zeichensatz *NONE hat und der CODE-Operand nicht angegeben wurde.

Die @CREATE-Anweisung verändert die aktuelle Zeilennummer nicht. Dies gilt auch dann, wenn neue Zeilen hinter dem Ende der bestehenden Arbeitsdatei angelegt werden.

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'·····················0001.00:00001(00)
```

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'·····························0001.00:00001(00)
```

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···············0001.00:00001(00)
```

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 an der Datensichtstation 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 ···············0001.00:00001(00)
```

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 ·····························································
```

9.28 @CREATE (Format 2) – Zeichenfolge einer Zeichenfolgevariablen zuweisen

Mit dem Format 2 der @CREATE-Anweisung wird einer Zeichenfolgevariablen eine Zeichenfolge zugewiesen.

Operation	**Operanden**	**F-Modus, L-Modus**
@**CR**EATE	svarex [:] [sting[,...]] [,**C**ODE=name]	

svarex — Zeichenfolgevariable, die neu angelegt werden soll.

: — Muss angegeben werden, wenn `svarex` nicht eindeutig von `string` zu trennen ist.

string — Eine oder mehrere Zeichenfolgen, die in der angegebenen Reihenfolge verkettet und einer Zeichenfolgevariablen zugewiesen werden sollen.

name — Zeichensatz, der für die angegebene Zeichenfolgevariable eingestellt werden soll.

Die in `string` angegebenen Zeichenfolgen werden im ersten Schritt miteinander verkettet. Haben alle beteiligten Zeichenfolgen denselben Zeichensatz, so erhält auch das Zwischenergebnis diesen Zeichensatz. Sind Zeichenfolgen mit unterschiedlichen Zeichensätzen beteiligt, so erhält das Zwischenergebnis den Zeichensatz `UTFE`.

Überschreitet das Zwischenergebnis nach dem Konvertieren die maximale Länge von 32768 Zeichen, wird auf die maximale Länge abgeschnitten und die Meldung `EDT2400` ausgegeben

Ist der `CODE`-Operand nicht angegeben, so werden der Zeichenfolgevariablen der Inhalt und der Zeichensatz des Zwischenergebnisses zugewiesen.

Ist der `CODE`-Operand angegeben, so wird der Zeichenfolgevariablen dieser Zeichensatz zugewiesen und das Zwischenergebnis wird vor der Zuweisung an die Zeichenfolgevariable in den Zeichensatz `name` konvertiert. Enthält die einzufügende Zeichenfolge Zeichen, die im Zeichensatz `name` nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @CREATE-Anweisung abgewiesen und die Fehlermeldung `EDT5453` ausgegeben.

Ist weder string noch der `CODE`-Operand angegeben, dann wird die Zeichenfolgevariable mit einem Leerzeichen und dem Zeichensatz `EDF041` neu angelegt. Ist `string` nicht angegeben, aber der `CODE`-Operand, dann wird die Zeichenfolgevariable mit dem Zeichensatz des `CODE`-Operanden und mit einem Leerzeichen in diesem Zeichensatz neu angelegt.

Beispiel

```
@READ 'SRC.EDF041' ------------------------------------------------- (1)
@CREATE #S01:1,CODE=UTF16 ------------------------------------------ (2)
@DELETE
@READ 'SRC.EDF045' ------------------------------------------------- (3)
@CREATE #S02:3,CODE=UTF16 ------------------------------------------ (4)
@CREATE #S03: #S01,#S02 -------------------------------------------- (5)
```

(1) Für die Arbeitsdatei wurde der Zeichensatz EDF041 eingestellt.

(2) Für die Zeichenfolgevariable #S01 wird mit dem Operanden CODE der Zeichensatz UTF16 eingestellt. Die erste Zeile der Arbeitsdatei wird von EDF041 nach UTF16 konvertiert und der Zeichenfolgevariablen #S01 zugewiesen.

(3) Für die Arbeitsdatei wurde der Zeichensatz EDF045 eingestellt.

(4) Für die Zeichenfolgevariable #S02 wird mit dem Operanden CODE der Zeichensatz UTF16 eingestellt. Die dritte Zeile der Arbeitsdatei wird von EDF045 nach UTF16 konvertiert und der Zeichenfolgevariablen #S02 zugewiesen.

(5) Für die Zeichenfolgevariable #S03 wird implizit der Zeichensatz UTF16 eingestellt, in dem die Zeichenfolgevariablen #S01 und #S02 vorliegen. Die Inhalte der Zeichenfolgevariablen #S01 und #S02 werden miteinander verkettet und der Zeichenfolgevariablen #S03 zugewiesen.

9.29 @CREATE (Format 3) – Zeichenfolge einlesen und Zeile erzeugen

Mit dem Format 3 der @CREATE-Anweisung wird von der Datensichtstation oder von SYSDTA eine Zeichenfolge gelesen und mit deren Inhalt eine Zeile erzeugt.

Operation	Operanden	L-Modus
@**CR**EATE	line **R**EAD [string[,...]] [,**C**ODE=name]	

line — Die Zeilennummer der aktuellen Arbeitsdatei, in die eine Zeichenfolge eingefügt werden soll. Existiert diese Zeile bereits, dann wird sie vollständig überschrieben

string — Eine oder mehrere Zeichenfolgen, die in der angegebenen Reihenfolge verkettet und an der Datensichtstation als Eingabeaufforderung ausgegeben werden sollen.

Ist string nicht angegeben, dann wird an der Datensichtstation keine Eingabeaufforderung ausgegeben.

name — Zeichensatz, der für die aktuelle Arbeitsdatei eingestellt werden soll, falls diese leer ist und den Zeichensatz *NONE hat.

Im Dialogbetrieb wird an der Datensichtstation die aus den Operanden gebildete Eingabeaufforderung ausgegeben und eine Zeichenfolge gelesen. Überschreitet die aus den Operanden gebildete Eingabeaufforderung die maximale Länge von 32763 Bytes, wird auf die maximale Länge abgeschnitten und die Fehlermeldung EDT2402 ausgegeben. Ist string nicht angegeben, wird eine Zeichenfolge von SYSDTA statt von der Datensichtstation gelesen.

Im Stapelbetrieb wird string ignoriert und eine Zeichenfolge immer von SYSDTA gelesen.

Die maximale Länge der gelesenen Zeichenfolge hängt vom Eingabe-Medium ab.

Hat die aktuelle Arbeitsdatei bereits einen Zeichensatz, so wird die gelesene Zeichenfolge vor dem Einfügen in den Zeichensatz der Arbeitsdatei konvertiert. Enthält die einzufügende Zeichenfolge Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @CREATE-Anweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Ist der CODE-Operand nicht angegeben und ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz *NONE, so wird die gelesene Zeichenfolge ohne Konvertieren in die Zeile eingefügt. Für die aktuelle Arbeitsdatei wird der Kommunikationszeichensatz als Zeichensatz eingestellt.

Ist der CODE-Operand angegeben und hat die aktuelle Arbeitsdatei bereits einen Zeichensatz, der von diesem verschieden ist, dann wird die @CREATE-Anweisung nicht ausgeführt und die Meldung EDT5458 ausgegeben.

Die Eingabe von F1 ohne Text an einer Datensichtstation bewirkt, dass die angegebene Zeile als Leerzeile (Zeile der Länge 0) angelegt wird. Eine leere Eingabe, die mit DUE oder einer anderen Funktionstaste gesendet wird, wird ignoriert und führt zur erneuten Ausgabe der Eingabeaufforderung.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Beispiel

```
   1.     @PROC 1
   1.     @CREATE 1 READ '*** NAME DES ERSTEN TEILNEHMERS ?',CODE=UTF16 - (1)
*** NAME DES ERSTEN TEILNEHMERS ? SCHÖLLER
   1.     @PROC 2 ------------------------------------------------------ (2)
   4.     @PRINT
   1.0000 MAIER
   2.0000 ZEILE WIRD ÜBERSCHRIEBEN
   3.0000 SCHMIDT
   4.     @CREATE 2 READ '*** MUELLER ODER MÜLLER ?' ------------------- (3)
*** MUELLER ODER MÜLLER ? MÜLLER
```

(1) Für die Arbeitsdatei 1, die leer ist und den Zeichensatz *NONE hat, wird mit dem Operanden CODE der Zeichensatz UTF16 vereinbart. An der Datensichtstation werden die Eingabeaufforderung '*** NAME DES ERSTEN TEILNEHMERS ?' ausgegeben und eine Zeichenfolge eingelesen. Diese wird in den Zeichensatz UTF16 konvertiert und in die erste Zeile der Arbeitsdatei geschrieben.

(2) Für die Arbeitsdatei 2 sei der Zeichensatz EDF041 eingestellt.

(3) An der Datensichtstation wird die Eingabeaufforderung '*** MUELLER ODER MÜLLER ?' ausgegeben und eine Zeichenfolge eingelesen. Diese wird in den Zeichensatz EDF041 der Arbeitsdatei konvertiert und in die zweite Zeile geschrieben. Der bestehende Inhalt der zweiten Zeile wird hierbei vollständig überschrieben.

9.30 @CREATE (Format 4) – Zeichenfolge einlesen und einer Zeichenfolgevariablen zuweisen

Mit dem Format 4 der @CREATE-Anweisung wird von der Datensichtstation oder von SYSDTA eine Zeichenfolge gelesen und einer Zeichenfolgevariablen zugewiesen.

Operation	Operanden	L-Modus
@CREATE	svarex **R**EAD [string[,...]] [,**C**ODE = name]	

svarex Zeichenfolgevariable, die neu angelegt werden soll.

string Eine oder mehrere Zeichenfolgen, die in der angegebenen Reihenfolge verkettet und an der Datensichtstation als Eingabeaufforderung ausgegeben werden sollen.

Ist string nicht angegeben, dann wird an der Datensichtstation keine Eingabeaufforderung ausgegeben.

name Zeichensatz, der für die angegebene Zeichenfolgevariable eingestellt werden soll.

Im Dialogbetrieb wird an der Datensichtstation die aus den Operanden gebildete Eingabeaufforderung ausgegeben und eine Zeichenfolge gelesen. Überschreitet die aus den Operanden gebildete Eingabeaufforderung die maximale Länge von 32763 Bytes, wird auf die maximale Länge abgeschnitten und die Fehlermeldung EDT2402 ausgegeben. Ist string nicht angegeben, wird eine Zeichenfolge von SYSDTA statt von der Datensichtstation gelesen.

Im Stapelbetrieb wird string ignoriert und eine Zeichenfolge immer von SYSDTA gelesen.

Die maximale Länge der gelesenen Zeichenfolge hängt vom Eingabe-Medium ab.

Ist der CODE-Operand nicht angegeben, dann werden der Zeichenfolgevariablen der Inhalt der Zeichenfolge und der Kommunikationszeichensatz als Zeichensatz zugewiesen.

Ist der CODE-Operand angegeben, so wird der Zeichenfolgevariablen dieser Zeichensatz zugewiesen und die gelesene Zeichenfolge vor dem Zuweisen in den Zeichensatz name konvertiert. Enthält die einzufügende Zeichenfolge Zeichen, die im Zeichensatz name nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @CREATE-Anweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Die Eingabe von F1 ohne Text an einer Datensichtstation bewirkt, dass die angegebene Zeichenfolgevariable als leere Zeichenfolgevariable neu angelegt wird. Eine leere Eingabe, die mit DUE oder einer anderen Funktionstaste gesendet wird, wird ignoriert und führt zur erneuten Ausgabe der Eingabeaufforderung.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

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 ---------------------- (1)
   2.     @ @SET #L2 = SUBSTR #S2 ---------------------------------------- (2)
   3.     @ @PRINT #L2
   4.     @END
   6.     @DO 1
*** WELCHE ZEILE SOLL DENN AUSGEGEBEN WERDEN *** 2 ---------------------- (3)
   2.0000 KEINER VERLAESST
   6.
```

(1) Über @CREATE...READ soll die Zeichenfolgevariable #S2 erzeugt werden. Zuvor wird aber der Text auf der Datensichtstation ausgegeben, der sich aus *** WELCHE und den Inhalten der Zeilen 4 und 5 sowie der Zeichenfolgevariablen #S1 ergibt.

(2) Die Eingabe wird interpretiert und in der Zeilennummervariablen #L2 abgelegt.

(3) Die über die Datensichtstation ausgegebene Anfrage wird beantwortet.

Beispiel 2

```
@CREATE #SO1 READ 'MUELLER ODER MÜLLER ?'
```

An der Datensichtstation wird die Eingabeaufforderung 'MUELLER ODER MÜLLER ?' ausgegeben und eine Zeichenfolge eingelesen. Der Zeichenfolgevariablen werden der Kommunikationszeichensatz als Zeichensatz und der Inhalt der gelesenen Zeichenfolge zugewiesen.

```
@CREATE #SO2 READ 'WOHNHAFT IN GÜNZBURG ODER DONAUWÖRTH ?',CODE=EDF041
```

Für die Zeichenfolgevariable #SO2 wird der Zeichensatz EDF041 vereinbart. An der Datensichtstation werden die Eingabeaufforderung ' WOHNHAFT IN GÜNZBURG ODER DONAUWÖRTH ?' ausgegeben und eine Zeichenfolge eingelesen. Diese wird in den Zeichensatz EDF041 konvertiert und der Zeichenfolgevariablen #SO2 zugewiesen.

9.31 @DELETE (Format 1) – Löschen von Zeilen und Zeichenfolgevariablen

Mit diesem Format der @DELETE-Anweisung können Zeilen der aktuellen Arbeitsdatei oder ein Bereich von Zeichenfolgevariablen ganz oder teilweise gelöscht werden.

Operation	Operanden	F-Modus, L-Modus
@DELETE	{ lines / svars } [:cols [:] [,...]	

lines — Der Zeilenbereich, der gelöscht werden soll.

svars — Der Bereich von Zeichenfolgevariablen, deren Inhalt gelöscht werden soll.

cols — Spaltenbereich in den angegebenen Zeilen oder Zeichenfolgevariablen, der gelöscht werden soll.

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte der Rest der Zeile bzw. Zeichenfolgevariablen gelöscht. Ist die erste Spaltenangabe größer als die Länge der Zeile bzw. Zeichenfolgevariablen, wird diese Zeile bzw. Zeichenfolgevariable nicht behandelt.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile gelöscht, eine Zeichenfolgevariable wird bei fehlender Spaltenangabe vollständig gelöscht (siehe unten).

Beim Löschen eines Zeilenbereichs werden stets nur die angegebenen Sätze gelöscht, darüber hinaus werden keinerlei Bereinigungen vorgenommen, selbst wenn die Arbeitsdatei nach dem Löschen keine Sätze mehr enthält.

Das *vollständige* Löschen einer Zeichenfolgevariablen setzt diese wieder in den Zustand, den sie vor der ersten Zuweisung eines Wertes hatte, d.h. sie enthält genau ein Leerzeichen im initialen Zeichensatz `EDF041`.

Wurde in der Arbeitsdatei 0 eine ISAM-Datei mit @OPEN (Format 2) real geöffnet, wird auch der entsprechende Bereich der ISAM-Datei gelöscht. Der Katalogeintrag der Datei bleibt aber in jedem Fall erhalten, auch wenn die Datei nach dem Löschen keine Sätze mehr enthält.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

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 ·····················································0001.00:00001(00)
```

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·····················································0002.10:00001(00)
```

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 ·····································································
·····························································0002.10:00001(00)
```

Der angegebene Bereich wurde gelöscht.

9.32 @DELETE (Format 2) – Vollständiges Löschen von Arbeitsdateien

Mit diesem Format der @DELETE-Anweisung können Arbeitsdateien vollständig gelöscht werden.

Operation	Operanden	F-Modus, L-Modus
@DELETE	[**A**LL | (procnr[,...])]	

ALL Alle Arbeitsdateien werden vollständig gelöscht.

procnr Die Arbeitsdateien mit den angegebenen Nummern (`0..22`) werden vollständig gelöscht.

Wird kein Operand angegeben, wird die aktuelle Arbeitsdatei vollständig gelöscht.

In der F-Modus-Anweisungszeile wird die Abkürzung `D` ohne Operanden mit einer Fehlermeldung abgewiesen, um ein unbeabsichtigtes Löschen der aktuellen Arbeitsdatei bei der Eingabe von `D`-Kurzanweisungen zu verhindern.

Ist unter den angegebenen Arbeitsdateien eine aktive Arbeitsdatei, so wird die Anweisung mit der Meldung `EDT5476` abgewiesen.

Das vollständige Löschen einer Arbeitsdatei macht diese zu einer leeren Arbeitsdatei. Insbesondere werden neben dem Löschen aller Sätze auch der Zeichensatz der Arbeitsdatei auf `*NONE` sowie andere arbeitsdateispezifische Einstellungen auf ihre Standardwerte gesetzt (siehe Abschnitt „Arbeitsdateien“ auf Seite 27). Außerdem wird eine vollständig gelöschte Datei (außer der aktuellen Arbeitsdatei) aus der Menge der belegten Arbeitsdateien (siehe @PROC USED) entfernt.

Die Arbeitsdateien werden ohne Rückfrage gelöscht, unabhängig von ihrem Inhalt. Sind in den zu löschenden Arbeitsdateien Dateien geöffnet, so werden diese ohne Rückschreiben implizit geschlossen. Dies gilt auch für mit @OPEN (Format 2) real geöffnete Dateien. Deren Sätze werden also nicht gelöscht.

9.33 @DELETE (Format 3) – Löschen von Dateien und Bibliothekselementen

Mit diesem Format der @DELETE-Anweisung können Dateien oder Elemente einer Bibliothek gelöscht werden.

Operation	Operanden	F-Modus, L-Modus
@DELETE	{ **L**IBRARY=path1 ([**E**LEMENT=] elname [(vers)][,eltype]) **E**LEMENT=elname [(vers)][,eltype] **F**ILE = path2 **P**OSIX-FILE = xpath }	

LIBRARY= Es soll ein Bibliothekselement gelöscht werden.

path1 Name der Bibliothek.

elname Name des Elements.

vers Version des zu löschenden Elements (siehe Handbuch LMS [14]). Wird `vers` nicht angegeben oder wird `*STD` angegeben, wird die höchste vorhandene Version des Elementes gelöscht.

eltype Typ des Elements. Zulässige Typangaben sind `S`, `M`, `P`, `J`, `D`, `X`, `R`, `C`, `H`, `L`, `U`, `F`, `*STD` und freie Typnamen mit entsprechendem Basistyp. Wird `eltype` nicht angegeben, wird der mit @PAR ELEMENT-TYPE voreingestellte Wert verwendet. Die zulässigen Elementtypen und deren Bedeutung sind im Abschnitt „Dateibearbeitung“ auf Seite 137 beschrieben.

ELEMENT= Das zu löschende Bibliothekselement wird durch seinen Namen ohne Angabe des Bibliotheknamens bestimmt. Es wird implizit die mit @PAR LIBRARY voreingestellte Bibliothek verwendet (sofern @PAR LIBRARY spezifiziert wurde – andernfalls wird die Fehlermeldung `EDT5181` ausgegeben).
Die Operanden `elname`, `vers` und `eltype` haben die gleiche Bedeutung wie bei expliziter Angabe der Bibliothek (siehe oben).

FILE= Es soll eine BS2000-Datei gelöscht werden.

path2 Der voll qualifizierte Dateiname einer BS2000-Datei, die gelöscht werden soll.

POSIX-FILE= Es soll eine POSIX-Datei gelöscht werden.

xpath Pfadname der POSIX-Datei, die gelöscht werden soll.

Der Operand `xpath` kann auch über eine Zeichenfolgevariable angegeben werden. Er muss über eine Zeichenfolgevariable angegeben werden, wenn er Sonderzeichen enthält, die in der EDT-Syntax eine spezielle Bedeutung haben (z.B. Leerzeichen oder Semikolon im F-Modus).

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Beispiel

```
@DELETE LIBRARY=PROGLIB(ELEMENT=TESTALT(VER2))
```

Die Version `VER2` des Bibliothekselementes `TESTALT` der Bibliothek `PROGLIB` mit dem Elementtyp `S` wird gelöscht.

9.34 @DELETE (Format 4) – Löschen von Satzmarkierungen

Mit diesem Format der @DELETE-Anweisung werden Satzmarkierungen (siehe Abschnitt „Satzmarkierungen“ auf Seite 46) gelöscht.

Operation	Operanden	F-Modus, L-Modus
@DELETE	**MA**RK [m[,...]]	

m
: Eine oder mehrere Satzmarkierungen (`1..9`), die in allen Sätzen der aktuellen Arbeitsdatei gelöscht werden sollen.

  Wird `m` nicht angegeben, werden alle Satzmarkierungen `1` bis `9` in den Sätzen der aktuellen Arbeitsdatei gelöscht.

Satzmarkierungen mit Sonderfunktion (Markierung `13`, `14`, `15`, siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden nicht gelöscht.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

9.35 @DELIMIT – Textbegrenzerzeichen vereinbaren

Mit der Anweisung @DELIMIT können Zeichen vereinbart werden, die beim Suchen mit @ON als Textbegrenzerzeichen wirken (siehe Abschnitt „Begrenzersymbole“ auf Seite 84).

Operation	Operanden	F-Modus, L-Modus
@DELIMIT	= [{ **R** | str1 | +|– str2 }]	

R — Als Textbegrenzerzeichen werden das Leerzeichen ␣ sowie die Zeichen `+.!*();-/,?:'="` vereinbart.

str1 — Zeichenfolge, die alle Zeichen enthält, die als Textbegrenzerzeichen vereinbart werden sollen.

str2 — Zeichenfolge, die die Zeichen enthält, die zusätzlich als Textbegrenzerzeichen vereinbart werden sollen (+), bzw. die nicht mehr als Textbegrenzerzeichen vereinbart sein sollen (-).

Wird kein Operand angegeben (@DELIMIT =), soll kein Zeichen mehr als Textbegrenzerzeichen vereinbart sein.

9.36 @DIALOG – Aufruf des Bildschirmdialogs

Mit der Anweisung @DIALOG kann der EDT bei Eingabe von SYSDTA (meist in BS2000-Systemprozeduren oder aus der Unterprogrammschnittstelle) in den Bildschirmdialog versetzt werden. Im Bildschirmdialog ist der bisherige Lesevorgang unterbrochen und der EDT liest seine Eingaben im F-Modus (oder im L-Modus nach Eingabe von @EDIT) von der Datensichtstation. Der Bildschirmdialog kann mit @HALT, @END, @RETURN oder K1 wieder verlassen werden. Der EDT setzt dann den unterbrochenen Lesevorgang fort.

Operation	Operanden	F-Modus, L-Modus
@DIALOG		

Die Anweisung wird im F-Modus ignoriert. Wird @DIALOG im L-Modus von einem andern Medium als SYSDTA eingegeben (z.B. wenn Anweisungen aus einer EDT-Prozedur gelesen werden oder wenn mit der Zeilennummer als Eingabeaufforderung von der Datensichtstation gelesen wird) oder wird sie im Stapelbetrieb aufgerufen, wird die Anweisung mit der Fehlermeldung EDT5400 bzw. EDT4920 abgewiesen.

Wurde der Bildschirmdialog aus einer BS2000-Systemprozedur aufgerufen, sind die Anweisungen @SYSTEM ohne Operanden und @EDIT ONLY gesperrt und werden mit der Meldung EDT4976 abgewiesen. Ein Wechsel in das Betriebssystem ist dann nur mit K2 möglich, sofern die BS2000-Systemprozedur nicht mit der Option INTERRUPT-ALLOWED=NO dagegen geschützt wurde (siehe Abschnitt „Zugriffsschutz“ auf Seite 103).

Mit @END, @HALT, @RETURN oder K1 wird der Bildschirmdialog beendet. Wenn der Aufruf von @DIALOG über die Unterprogramm-Schnittstelle oder aus einer BS2000-Prozedur erfolgte, wird mit der auf @DIALOG folgenden Anweisung fortgesetzt. Folgt an der Unterprogrammschnittstelle nach @DIALOG keine weitere Anweisung, bekommt das aufrufende Programm die Kontrolle. Wenn der Aufruf von SYSDTA (nach @EDIT ONLY) erfolgte, wird die nächste Eingabe von SYSDTA angefordert. In allen Fällen bleibt der EDT geladen und alle Einstellungen des EDT bleiben so, wie sie zum Zeitpunkt der Beendigung des Bildschirmdialogs waren.

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=(4,5) ------------------------------------------- (1)
/START-EDTU
@PROC 1 ----------------------------------------------------------------- (2)
@COPY FILE=&FILE1 ------------------------------------------------------- (3)
@PAR SCALE=ON ----------------------------------------------------------- (4)
@DIALOG ----------------------------------------------------------------- (5)
@SETF(1) ---------------------------------------------------------------- (6)
@WRITE FILE=&FILE2 ------------------------------------------------------ (7)
@HALT ------------------------------------------------------------------- (8)
/MODIFY-JOB-SWITCHES OFF=(4,5)
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/END-PROCEDURE
```

(1) Der Auftragsschalter 5 wird vor dem Laden des EDT gesetzt. Damit wird der L-Modus und das Lesen der Eingaben von `SYSDTA` eingestellt.

(2) Es wird in die Arbeitsdatei 1 umgeschaltet.

(3) Eine Datei soll eingelesen werden. Der Dateiname wird beim Ablauf der Prozedur angefordert.

(4) Die Spaltenzähleranzeige wird eingeschaltet.

(5) Der EDT soll in den F-Modus-Bildschirmdialog umschalten und das Arbeitsfenster am Bildschirm ausgegeben. Im Dialog können dann alle Anweisungen des F- und L-Modus eingegeben werden. Mit @END, @HALT oder @RETURN bzw. mit K1 wird der F-Modus-Bildschirmdialog beendet und mit der auf @DIALOG folgenden Anweisung fortgesetzt.

(6) 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.

(7) Die Arbeitsdatei 1 wird in eine SAM-Datei geschrieben. Der Dateiname wird beim Ablauf der Prozedur angefordert.

(8) Der EDT wird beendet.

```
/call-procedure name=proc.dialog
%/BEGIN-PROCEDURE LOGGING=A,PARAMETERS=YES(PROCEDURE-PARAMETERS
=(&FILE1=,&FILE2=),ESCAPE-CHARACTER='&')
%/ASSIGN-SYSDTA TO-FILE=*SYSCMD
%/MODIFY-JOB-SWITCHES ON=(4,5)
%/START-EDTU
%PROC 1
%@COPY FILE=&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 Habe nun, ach!Philosophie,<·····································
  2.00 Juristerei und Medizin,<·····································
  3.00 Und leider auch Theologie<·····································
  4.00 Durchaus studiert, mit heißem Bemühn.<·····································
  5.00 Da steh ich nun ich arm er Tor!<·····································
  6.00 Und bin so klug als wie zuvor<·····································
@halt·····0001.00:00001(01)
```

Entsprechend den unter (4) definierten Voreinstellwerten wird die Spaltenzähleranzeige ausgegeben. Mit @HALT wird der F-Modus-Bildschirmdialog wieder beendet und die 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-Bildschirmdialog werden ggf. noch weitere Meldungen ausgegeben.

```
%@SETF (1)
%@WRITE FILE=&FILE2
%&FILE2=bsp.dialog1
%@WRITE FILE=BSP.DIALOG1
%@HALT
%/MODIFY-JOB-SWITCHES OFF=(4,5)
%/ASSIGN-SYSDTA TO-FILE=*PRIMARY
%/END-PROCEDURE
/
```

9.37 @DO (Format 1) – Starten von EDT-Prozeduren aus Arbeitsdateien

Mit Format 1 der @DO-Anweisung wird eine @DO-Prozedur gestartet, d.h. die in der angegebenen Arbeitsdatei stehenden Textzeilen und EDT-Anweisungen werden abgearbeitet.
Einzelheiten zum Aufbau und zur Bearbeitung von EDT-Prozeduren findet man im Abschnitt „EDT-Prozeduren“ auf Seite 66.

Operation	Operanden	F-Modus, L-Modus
@DO	procnr [,] [(param [,...])] [spec] [=line1,line2 [, [-] line3]] [**P**RINT]	

procnr — Die Nummer der Arbeitsdatei (`1..22`), deren Inhalt der EDT abarbeiten soll. Ist die Arbeitsdatei leer, wird die Fehlermeldung `EDT4950` ausgegeben.

param — Parameter, die an die auszuführende Prozedur übergeben werden. Die Parameter müssen in der Prozedur mit @PARAMS definiert sein (siehe @PARAMS-Anweisung). Sie werden voneinander durch Komma getrennt.

Sind Parameter (auch leere) angegeben und enthält die Prozedur keine @PARAMS-Anweisung, wird die Anweisung mit der Meldung `EDT4944` abgewiesen. Sind zu viele Parameter angegeben wird die Fehlermeldung `EDT4963` bzw. `EDT4965` ausgegeben.

Es wird zwischen Stellungs- und Schlüsselwortparametern unterschieden. Bei Stellungsparametern wird nur der Wert des Parameters übergeben, bei Schlüsselwortparametern ein Ausdruck der Form `formal=value`, wobei `formal` das Schlüsselwort (ohne das einleitende &-Zeichen) ist, mit dem der Parameter in der @PARAMS-Anweisung definiert wurde (Einzelheiten zur Parameterübergabe sind weiter unten sowie bei der @PARAMS-Anweisung erläutert).

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. Wird ein Stellungsparameter nach einem Schlüsselwortparameter angegeben, wird die Anweisung mit der Meldung `EDT4948` abgewiesen. Wird ein Schlüsselwortparameter mehrfach angegeben, erfolgt die Meldung `EDT3911`.

Die mögliche Anzahl der Parameter ist durch die maximale Länge einer EDT-Anweisung begrenzt.

spec

Schleifenzähler. In der Prozedur kann er 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 Schleifenzählers (siehe Abschnitt „EDT-Prozeduren“ auf Seite 66).

Der Schleifenzähler muss eines der zugelassenen Sonderzeichen sein, sonst wird @DO mit der Fehlermeldung `EDT3952` abgewiesen. Um Fehler und unvorhersehbare Ergebnisse zu vermeiden, sollten folgende Zeichen nicht als Schleifenzähler gewählt werden:

`% $ ? * ( : # + - . < = > ’ ;`

Das Zeichen `'='` kann wegen der Syntax des Kommandos gar nicht angegeben werden. Wird die Prozedur im F-Modus gestartet, darf `';'` nicht verwendet werden.

Geeignete Zeichen für den Schleifenzähler sind:

`! { } [ ] | /`

Wird der Schleifenzähler nicht angegeben, gilt er als undefiniert. Wird die Operandenfolge `line1,line2,[-]line3` nicht angegeben, hat der Schleifenzähler den Wert 1.

=line1,line2,[–]line3

Eine Prozedur wird mehrmals durchlaufen (siehe Beispiel 3).

Vor dem ersten Durchlaufen weist der EDT `line1` dem Schleifenzähler als Anfangswert zu. Nach jedem Durchlauf erhöht oder vermindert (Minuszeichen vor `line3`) der EDT den Wert des Schleifenzählers um `line3`. Standardwert für `line3` ist 1. Solange der Schleifenzähler den Wert von `line2` 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 der Bearbeitung der letzten Zeile erfolgt (`REPEAT UNTIL`). Wird die letzte Zeile aufgrund einer @RETURN-Anweisung in der Prozedur nicht erreicht, findet die Prüfung nicht statt und die Prozedur wird nicht erneut durchlaufen.

Für `line1`, `line2` oder `line3` 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 Schleifenzähler wird innerhalb der Prozedur wie eine Zeilennummervariable behandelt. Eine Ersetzung des Schleifenzählers durch den aktuellen Wert findet daher auch nur dann statt, wenn der Schleifenzähler als Zeilennummer angesprochen wird, insbesondere also *nicht* in Literalen. Bei Angabe des Schleifenzählers in einer Anweisung, die eine Zeilennummer mit impliziter Schrittweite erlaubt, wird die implizite Schrittweite – wie bei Zeilennummervariablen – daher immer als `0.0001` angenommen (siehe Beispiel 6).

Da ein Sonderzeichen nur dann als Schleifenzähler betrachtet wird, wenn es an Stelle einer Zeilennummer verwendet und in der aufrufenden @DO-Anweisung auch als Schleifenzähler spezifiziert wird, kann bei geschachtelten äußeren Schleifen der gleiche Schleifenzähler mehrfach verwendet werden. Er nimmt dann stets die Werte an, die in der aufrufenden @DO-Anweisung angegeben wurden (siehe Beispiel 7). Ein Zugriff auf Schleifenzähler, die beim Aufruf in einer tieferen Ebene der Schachtelung angegeben wurden, ist nicht möglich, die dort verwendeten Symbole werden in der höheren Ebene nicht durch Zeilennummern ersetzt, was meist zu Syntaxfehlern führt.

Der Standardwert für `line1`, `line2` und `line3` ist 1.

PRINT

Jede Zeile der Prozedur soll vor ihrer Verarbeitung (mit expandierten Parametern) protokolliert werden. Die Ausgabe erfolgt im Dialogbetrieb nach `SYSOUT` und im Stapelbetrieb nach `SYSLST`.

Durch die Angabe von `PRINT` wird auch erreicht, dass alle Fehlermeldungen ausgegeben werden und der EDT-Fehlerschalter gesetzt wird. Normalerweise werden die beiden Meldungen `EDT0901` und `EDT4932` in Prozeduren nicht ausgegeben und der EDT-Fehlerschalter nicht gesetzt (siehe Abschnitt „Meldungstexte“ auf Seite 664).

Der Wert eines Stellungsparameters ergibt sich aus allen angegebenen Zeichen zwischen den Kommas bzw. den Klammern, einschließlich der Leerzeichen.
Befinden sich keine Zeichen zwischen den Kommas bzw. den Klammern, so ist der Wert des Stellungsparameters die leere Zeichenfolge.
Der Wert eines Schlüsselwortparameters ergibt sich aus allen angegebenen Zeichen zwischen dem Gleichheitszeichen und dem nachfolgenden Komma bzw. der Klammer, einschließlich der Leerzeichen.
Befinden sich keine Zeichen zwischen dem Gleichheitszeichen und dem nachfolgenden Komma bzw. der Klammer, so ist der Wert des Schlüsselwortparameters die leere Zeichenfolge.
Ein Parameter kann in Hochkommas eingeschlossen werden, diese werden nicht übernommen, wenn sie als erstes und letztes Zeichen des Parameterwerts auftreten und dazwischen entweder keine oder nur doppelte Hochkommas auftreten.

In allen anderen Fällen bleiben alle angegebenen Hochkommas Bestandteil des Parameters (siehe Beispiele bei der @PARAMS-Anweisung). Die Zeichen Komma und schließende Klammer können nur Bestandteil eines Parameters sein, wenn sie innerhalb einer durch Hochkomma eingeschlossenen Teilzeichenfolge des Parameterwertes vorkommen. Hochkommas müssen immer paarig in einem Parameterwert vorkommen. Ein einzelnes Hochkomma kann nicht in einem Parameterwert übergeben werden. Wurde zuvor mit @QUOTE einem anderen Zeichen die Funktion des Hochkommas zugeordnet, gilt dies **nicht** für die den Parameterwert einschließenden Hochkommas.
Wird ein Stellungsparameter nicht angegeben, erhält der Parameter als Wert die leere Zeichenfolge.
Wird ein Schlüsselwortparameter nicht angegeben, erhält er den Standardwert, der in der @PARAMS-Anweisung festgelegt wurde.

Die angegebenen Parameter werden bei der Parameter-Ersetzung in den Zeichensatz der Prozedur-Arbeitsdatei konvertiert. Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER). Anderenfalls wird die @DO-Anweisung abgewiesen und die Fehlermeldung `EDT5453` ausgegeben.

In Parametern können evtl. Unicode-Ersatzdarstellungen enthalten sein. Diese werden bei der Parameter-Ersetzung noch nicht expandiert. Dies geschieht erst bei der Ausführung der jeweiligen Prozedurzeile.

EDT-Prozeduren können durch K2 jederzeit unterbrochen werden.
Vom Betriebssystem aus kann man mit `/RESUME-PROGRAM` die Prozedur fortsetzen oder mit `/INFORM-PROGRAM` in den EDT zurückkehren und die Prozedur abbrechen.

Während der Abarbeitung einer Benutzeranweisung (siehe @USE) kann die Prozedur nicht mit `/INFORM-PROGRAM` abgebrochen werden.

Eine fehlerhafte Anweisung innerhalb der Prozedur führt nicht zum Abbruch.

Beispiel 1

```
1.     @SET #S0 = 'TEST VON PROZEDURDATEI 1'
1.     @PROC 1 ------------------------------------------------------ (1)
1.     @ @SET #S1 = #S0:1-4: ---------------------------------------- (2)
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 --------------------------------------------------------- (3)
1.     @DO 1 -------------------------------------------------------- (4)
  #S01 TEST
  #S02      VON
  #S03          PROZEDURDATEI 1
  #S04
  #S00 TEST VON PROZEDURDATEI 1
1.
```

(1) Es wird in die Arbeitsdatei 1 umgeschaltet.

(2) 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.

(3) Mit @END wird aus der Arbeitsdatei 1 zurückgekehrt.

(4) Die in Arbeitsdatei 1 stehende Prozedur wird aufgerufen.

Beispiel 2

```
1.     @PROC 2 ------------------------------------------------------ (1)
1.     @ @PARAMS &STRING -------------------------------------------- (2)
2.     @ @SET #S1 = '++++++++++'
3.     @ @SET #S2 = &STRING ----------------------------------------- (3)
4.     @ @PRINT #S2
5.     @END
1.     @DO 2(#S1) PRINT --------------------------------------------- (4)
1.     @SET #S1 = '++++++++++'
1.     @SET #S2 = #S1
1.     @PRINT #S2
  #S02 ++++++++++
1.     @DO 2('#S1') PRINT ------------------------------------------- (5)
1.     @SET #S1 = '++++++++++'
1.     @SET #S2 = #S1
1.     @PRINT #S2
  #S02 ++++++++++
1.     @DO 2(␣'#S1'␣) PRINT ----------------------------------------- (6)
1.     @SET #S1 = '++++++++++'
1.     @SET #S2 = ␣'#S1'␣
1.     @PRINT #S2
  #S02 #S1
1.
```

(1) Es wird in die Arbeitsdatei 2 umgeschaltet.

(2) Die erste in dieser Arbeitsdatei abgelegte Zeile ist eine @PARAMS-Anweisung. Damit kann innerhalb dieser Arbeitsdatei der Stellungsparameter `&STRING` angesprochen werden.

(3) `#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.

(4) 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.

(5) 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 (4).

(6) Der einzige Unterschied zu (5) 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 ------------------------------------------------------- (1)
  1.     @ @CREATE $+1: $,'*' ------------------------------------------ (2)
  2.     @END ---------------------------------------------------------- (3)
  2.     @DO 3,!=1,15 -------------------------------------------------- (4)
  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 **************** ------------------------------------------------ (5)
  2.
```

(1) Es wird in die Arbeitsdatei 3 umgeschaltet.

(2) Eine einzige EDT-Anweisung wird in die Arbeitsdatei 3 geschrieben.

(3) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(4) Die Arbeitsdatei 3 wird ausgeführt. Hierbei wird als Schleifenzähler 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.

(5) 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 ------------------------------------------------------- (1)
   1.     @ @PRINT !.-.$ ------------------------------------------------ (2)
   2.     @ @PRINT #S4 N
   3.     @END
   5.     @DO 4,!=$,%,-1 ------------------------------------------------ (3)
   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.
```

(1) Es wird in die Arbeitsdatei 4 umgeschaltet.

(2) Eine Zeilennummer wird über den Schleifenzähler `!` angesprochen.

(3) Die Arbeitsdatei 4 wird mehrmals ausgeführt. Beim 1. Durchlauf wird für `!` der Wert der höchsten vergebenen Zeilennummer angenommen. Bei jedem weiteren Durchlauf erniedrigt sich dieser Wert um 1 (drittes `line=-1`), bis der Schleifenzähler den Wert der niedrigsten vergebenen Zeilennummer (`%`) angenommen hat.

Beispiel 5

```
1.     @PROC 4 ------------------------------------------------------ (1)
1.     @READ 'PROC-DATEI.4' ----------------------------------------- (2)
6.     @PRINT
1.0000 @PARAMS &A, &OPTION=ALL
2.0000 ABCABCABCABC
3.0000 EFG
4.0000 @ON  1 CHANGE &OPTION 'ABC' TO '&A'
5.0000 @5: &A
6.     @END --------------------------------------------------------- (3)
1.     @DO 4 ('A','B',OPTION=R) ------------------------------------- (4)
6.     @PRINT
1.0000 ABCABCABCA,'B ----------------------------------------------- (5)
2.0000 EFG
5.0000  A,''B
6.
```

(1) Es wird in die Arbeitsdatei 4 umgeschaltet.

(2) Die SAM-Datei `PROC-DATEI.4` wird in die Arbeitsdatei 4 gelesen.

(3) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(4) Der Standardwert `ALL` des Schlüsselwortparameters `&OPTION` wird beim Aufruf durch `R` ersetzt. Folglich findet das Suchen und Ersetzten in Zeile 1 rückwärts statt.

(5) Beim Ausführen der Arbeitsdatei wurden Zeilen in die aktuelle Arbeitsdatei geschrieben. In Zeile 1 hat der EDT eines der 2 aufeinander folgenden Hochkommas unterdrückt. In Zeile 3 hat der EDT den Parameterwert unverändert übernommen.

Beispiel 6

```
1.     AAA
2.     BBB
3.     CCC
4.     @PROC 1
1.     @@COPY 1-3 TO ! ------------------------------------------------ (1)
2.     @END
4.     @DO 1,!=4,4 ---------------------------------------------------- (2)
5.0002 @DO 1,!=5.0,5.0 ------------------------------------------------ (3)
6.0002 @PRINT ------------------------------------------------------- (4)
1.0000 AAA
2.0000 BBB
3.0000 CCC
4.0000 AAA
4.0001 BBB
4.0002 CCC
5.0000 AAA
5.0001 BBB
5.0002 CCC
6.0002
```

(1) Die Prozedurdatei enthält eine Anweisung, bei der die Schrittweite implizit durch die Angabe der Zeilennummer des Zielbereichs festgelegt wird.

(2) Aufruf der Prozedur mit Schleifenzähler `4`.

(3) Aufruf der Prozedur mit Schleifenzähler `5.0`.

(4) Die Ausgabe zeigt, dass für den Schleifenzähler stets die implizite Schrittweite `0.0001` genommen wurde - wie dies auch bei Zeilennummervariablen der Fall ist.

Beispiel 7

```
1.     @PROC 1
1.     @   @DO 2, !=!,1,-1 ------------------------------------------- (1)
1.     @END
1.     @PROC 2
1.     @   @SET #L1 = ! ---------------------------------------------- (2)
2.     @   @SET #S1 = C #L1 ------------------------------------------ (3)
3.     @   @PRINT #S1 ------------------------------------------------ (4)
3.     @END
1.     @DO 1, !=1,4 -------------------------------------------------- (5)
  #S01    1.0000 ------------------------------------------------------ (6)
  #S01    2.0000
  #S01    1.0000
  #S01    3.0000
  #S01    2.0000
  #S01    1.0000
  #S01    4.0000
  #S01    3.0000
  #S01    2.0000
  #S01    1.0000
1.
```

(1) In Arbeitsdatei 1 wird eine @DO-Anweisung abgelegt. Der Schleifenzähler `!` wird hier in zwei Bedeutungen verwendet, einmal wird er neu definiert für den Aufruf der @DO-Prozedur in Arbeitsdatei 2, zum anderen wird als Anfangswert der aktuelle Wert des Schleifenzählers übernommen, der beim Aufruf von Arbeitsdatei 1 spezifiziert wurde. Dies ist möglich, weil die Ersetzung nur dort stattfindet, wo das Symbol `!` auch als Zeilennummer verwendet wird.

(2) In Arbeitsdatei 2 wird der Wert des dort aktuellen Schleifenzählers der Zeilennummervariablen `#L1` zugewiesen.

(3) Die Zeilennummervariable `#L1` wird abdruckbar in `#S1` abgelegt.

(4) Die Zeichenfolgevariable `#S1` wird ausgegeben.

(5) Beim Aufruf von Arbeitsdatei 1 lässt man den Schleifenzähler von `1` bis `4` laufen.

(6) Das Ergebnis zeigt: der innere Schleifenzähler (in Arbeitsdatei 2) läuft viermal rückwärts bis `1.0000` ausgehend von den Startwerten `1.0000` bis `4.0000`.

9.38 @DO (Format 2) – Protokollierung aus- oder einschalten

Mit diesem Format der @DO-Anweisung kann die Protokollierung der gelesenen Anweisungen (siehe den Operanden PRINT in Format 1 der @DO-Anweisung) 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 kann zur Fehlersuche in EDT-Prozeduren verwendet werden. Man kann z.B. feststellen, ob eine bestimmte Stelle einer Prozedur durchlaufen wird oder nicht.

Beispiel

```
1.     @PROC 5 ------------------------------------------------------ (1)
1.     @ @SET #S5 = 'A'
2.     @ @DO N ------------------------------------------------------ (2)
3.     @ @CREATE #S6: 'B'*6,#S5
4.     @ @CREATE #S7: #S6,'C',#S6
5.     @ @DO P
6.     @ @PRINT #S5.-#S7 -------------------------------------------- (3)
7.     @ @DELETE #S5.-#S7
8.     @END --------------------------------------------------------- (4)
1.     @DO 5 PRINT -------------------------------------------------- (5)
1.     @SET #S5 = 'A'
1.     @DO N
1.     @PRINT #S5.-#S7
  #S05 A
  #S06 BBBBBBA
  #S07 BBBBBBACBBBBBBA
1.     @DELETE #S5.-#S7
1.
```

(1) Es wird in die Arbeitsdatei 5 umgeschaltet.

(2) Die folgenden Zeilen der Prozedur werden nicht mehr protokolliert.

(3) Der EDT protokolliert die folgenden Zeilen der Prozedur vor der Verarbeitung.

(4) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(5) Die Prozedur in Arbeitsdatei 5 wird gestartet. Die Anweisungen sollen vor der Durchführung protokolliert werden.

9.39 @DROP – Löschen von Arbeitsdateien

Die Anweisung @DROP löscht die angegebenen Arbeitsdateien vollständig.

Operation	Operanden	L-Modus
@DROP	{ 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.

Die Anweisung @DROP darf nur eingeben werden, wenn die aktuelle Arbeitsdatei die Arbeitsdatei 0 ist. Innerhalb von @DO-Prozeduren ist @DROP ebenfalls nicht erlaubt.

Die Arbeitsdateien werden ohne Rückfrage gelöscht, unabhängig von ihrem Inhalt. Sind in den zu löschenden Arbeitsdateien Dateien mit @OPEN oder @XOPEN geöffnet, so werden diese implizit geschlossen.

Hinweis

Die Anweisung @DROP wirkt wie das Löschen jeder der angegebenen Arbeitsdateien mit @DELETE (Format 2) und entfernt die angegebenen Arbeitsdateien aus der Menge der belegten Arbeitsdateien (siehe @PROC-Anweisung).
Geöffnete Dateien bzw. Bibliothekselemente sollten vorher zurück geschrieben und geschlossen werden (siehe @CLOSE), andernfalls gehen alle Änderungen verloren.

Beispiel 1

```
   1.     @PROC USED ------------------------------------------------- (1)
<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 -------------------------------------------------- (2)
   1.     @PROC USED
<03>     1.0000 TO    3.0000
<05>     1.0000 TO    1.0000 ---------------------------------------- (3)
<08>     1.0000 TO    1.0000
<14>     1.0000 TO    1.0000
   1.     @DROP 8,5 ------------------------------------------------- (4)
   1.     @PROC USED
<03>     1.0000 TO    3.0000 ---------------------------------------- (5)
<14>     1.0000 TO    1.0000
   1.
```

(1) Die belegten Arbeitsdateien `1..22` sollen ausgegeben werden. Es sind in diesem Fall die Arbeitsdateien `3`, `5`, `8`, `10`, `14`.

(2) Die Arbeitsdatei `10` wird gelöscht und freigegeben.

(3) @PROC USED gibt aus, dass nur noch die Arbeitsdateien `3`, `5`, `8`, `14` belegt sind.

(4) Mit @DROP können auch mehrere Arbeitsdateien gelöscht und freigegeben werden, wie hier z.B. `5` und `8`.

(5) Jetzt bleiben nur noch die Arbeitsdateien `3` und `14` übrig.

Beispiel 2

```
   1.     @PROC USED ------------------------------------------------- (1)
<03>     1.0000 TO    3.0000
<14>     1.0000 TO    1.0000
   1.     @DROP ALL ------------------------------------------------- (2)
   1.     @PROC USED
% EDT0907 NO WORK FILES USED ------------------------------------------ (3)
   1.
```

(1) Alle belegten Arbeitsdateien sollen ausgegeben werden.

(2) Die Arbeitsdateien `1..22` werden gelöscht und freigegeben.

(3) Es ist keine der Arbeitsdateien `1` bis `22` mehr belegt.

9.40 @EDIT (Format 1) – Umschalten in den F-Modus

Das Format 1 der Anweisung @EDIT schaltet im Dialogbetrieb vom L-Modus in den F-Modus.

Operation	**Operanden**	**F-Modus, L-Modus**
@EDIT	**F**ULL [**SCREEN**]	

Im Stapelbetrieb und im F-Modus wird die Anweisung ignoriert. Innerhalb einer EDT-Prozedur wird sie mit der Meldung EDT4920 abgewiesen.

Wird @EDIT FULL SCREEN im Dialog innerhalb eines Anweisungsblocks (BLOCK-Modus) angegeben, werden die nachfolgenden Anweisungen ignoriert.

Hinweis

Die Anweisung @PAR EDIT-FULL hat eine andere Semantik als @EDIT FULL SCREEN, kann also nicht anstelle von @EDIT FULL SCREEN verwendet werden.

9.41 @EDIT (Format 2) – Einstellen der Eingabe von der Datensichtstation

Das Format 2 der Anweisung @EDIT schaltet im L-Modus des Dialogbetriebs den Eingabestrom auf Terminal-Eingabe um. Es wird mit WRTRD gelesen und die aktuelle Zeilennummer als Eingabeaufforderung ausgegeben.

Bei Eingabe im F-Modus wird zunächst in den L-Modus gewechselt. Im Stapelbetrieb wird nur die Protokollierung beeinflusst (siehe Hinweis unten).

Operation	Operanden	F-Modus, L-Modus
@EDIT	[**P**RINT] [**S**EQUENTIAL]	

PRINT Die Angabe von PRINT bewirkt, dass im Dialogbetrieb vor der Ausgabe der Eingabeaufforderung, im Stapelbetrieb vor dem Einlesen der nächsten Anweisung oder Datenzeile die Zeilennummer und der Zeileninhalt der aktuellen Zeile am Bildschirm ausgegeben werden.

Wird PRINT nicht angegeben, schaltet die Anweisung diese Funktion wieder ab, ohne den L-Modus zu verlassen.

SEQUENTIAL Der Operand beeinflusst das Hochzählen der aktuellen Zeilennummer. Im Regelfall wird im L-Modus bei Eingabe einer Datenzeile bzw. bei den Anweisungen @+ oder @– die aktuelle Zeilennummer um die Schrittweite erhöht bzw. erniedrigt. 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 dazwischen liegende Zeile gibt. Im anderen Fall wird die erste dazwischen liegende Zeile zur aktuellen Zeile.

Wird SEQUENTIAL nicht angegeben, schaltet die Anweisung diese Funktion wieder ab, ohne den L-Modus zu verlassen.

Hinweis

Im Stapelbetrieb wird generell von SYSDTA gelesen. Jedoch wird die Art der Protokollierung (siehe @LOG-Anweisung) dadurch beeinflusst, ob @EDIT Format 2 oder @EDIT Format 3 gegeben wurde. Bei @EDIT Format 2 wird jede protokollierte Eingabe mit der aktuellen Zeilennummer eingeleitet, bei @EDIT Format 3 nicht. Letzteres entspricht der Einstellung nach dem Starten des EDT im Stapelbetrieb.

9.42 @EDIT (Format 3) – Einstellen der Eingabe von SYSDTA

Das Format 3 der Anweisung @EDIT schaltet im L-Modus des Dialogbetriebs den Eingabestrom auf Eingabe von SYSDTA um. Es wird mit RDATA gelesen. Die Art und Weise der Eingabeaufforderung wird durch Einstellungen des Betriebssystems festgelegt (standardmäßig wird ein * ausgegeben).

Bei Eingabe im F-Modus wird zunächst in den L-Modus gewechselt. Im Stapelbetrieb wird nur die Protokollierung beeinflusst (siehe Hinweis unten).

Operation	**Operanden**	**F-Modus, L-Modus**
@EDIT	**O**NLY [**P**RINT] [**S**EQUENTIAL]	

PRINT Die Angabe von PRINT bewirkt, dass im Dialogbetrieb vor der Ausgabe der Eingabeaufforderung, im Stapelbetrieb vor dem Einlesen der nächsten Anweisung oder Datenzeile die Zeilennummer und der Zeileninhalt der aktuellen Zeile am Bildschirm ausgegeben werden.

Wird PRINT nicht angegeben, wird nur der Eingabestrom umgeschaltet und der evtl. angegebene SEQUENTIAL-Operand ausgewertet.

Die mit @EDIT ONLY PRINT eingeschaltete Funktion kann nur mit @EDIT (Format 2) wieder abgeschaltet werden, ohne den L-Modus zu verlassen.

SEQUENTIAL Der Operand beeinflusst das Hochzählen der aktuellen Zeilennummer. Im Regelfall wird im L-Modus bei Eingabe einer Datenzeile bzw. bei den Anweisungen @+ oder @– die aktuelle Zeilennummer um die Schrittweite erhöht bzw. erniedrigt. 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 dazwischen liegende Zeile gibt. Im anderen Fall wird die erste dazwischen liegende Zeilennummer zur aktuellen Zeilennummer.

Wird SEQUENTIAL nicht angegeben, wird nur der Eingabestrom umgeschaltet und der evtl. angegebene PRINT-Operand ausgewertet.

Die mit @EDIT ONLY SEQUENTIAL eingeschaltete Funktion kann nur mit @EDIT (Format 2) wieder abgeschaltet werden, ohne den L-Modus zu verlassen.

Wenn mit @EDIT Format 3 die Eingabe auf SYSDTA umgelenkt wird, werden Anweisungen und Daten im aktuell für SYSDTA eingestellten Zeichensatz interpretiert. Dieser unterscheidet sich im Dialogbetrieb normalerweise nicht von dem mit /MODIFY-TERMINAL-OPTIONS für die Datensichtstation vereinbarten Zeichensatz, es sei denn, SYSDTA wurde vorher einer Datei zugewiesen. Sofern sich die Anweisungen und Daten auf eine der Arbeitsdateien des EDT beziehen, muss die Eingabe evtl. in den Zeichensatz der betroffenen Arbeitsdatei konvertiert werden. Einzelheiten dazu entnehme man dem Abschnitt „Zeichensätze“ auf Seite 48.

Hinweis

Im Stapelbetrieb wird generell von SYSDTA gelesen. Jedoch wird die Art der Protokollierung (siehe @LOG-Anweisung) dadurch beeinflusst, ob @EDIT Format 2 oder @EDIT Format 3 gegeben wurde. Bei @EDIT Format 2 wird jede protokollierte Eingabe mit der aktuellen Zeilennummer eingeleitet, bei @EDIT Format 3 nicht. Letzteres entspricht der Einstellung nach dem Starten des EDT im Stapelbetrieb.
Wird im Dialogbetrieb EOF von SYSDTA erkannt, schaltet der EDT automatisch auf Terminal-Eingabe um.

9.43 @EDIT (Format 4) – Vollständige Darstellung der Sätze steuern

Das Format 4 der Anweisung @EDIT schaltet im F-Modus für die aktuelle Arbeitsdatei zwischen der vollständigen Darstellung der Sätze und der Darstellung eines Satzausschnitts im Datenfenster der aktuellen Arbeitsdatei um.

Operation	Operanden	F-Modus
@EDIT	**L**ONG [{ **ON** / **O**FF }]	

ON Die Darstellung im F-Modus wird so eingestellt, dass die Datensätze (soweit möglich) vollständig im Datenfenster dargestellt werden. Die Zeilennummernanzeige wird abgeschaltet. Weitere Einzelheiten zur Darstellung im EDIT–LONG-Modus findet man im Abschnitt „Das Arbeitsfenster“ auf Seite 107.

Das Einschalten des EDIT–LONG-Modus bewirkt implizit das Ausschalten der Zeilennummernanzeige (@PAR INDEX=OFF) und des Hexadezimal-Modus (@PAR HEX=OFF).

OFF Die Darstellung im F-Modus wird so eingestellt, dass von längeren Datensätzen jeweils nur ein Ausschnitt (je nach Datensichtstation, Einstellung mit @VDT und @PAR INDEX sind dies 72, 80, 124 oder 132 Zeichen) im Datenfenster sichtbar ist. Einzelheiten zur Darstellung eines Arbeitsfensters findet man im Abschnitt „Das Arbeitsfenster“ auf Seite 107.

Wird der EDIT–LONG-Modus verlassen, bleibt die Zeilennummernanzeige ausgeschaltet. Der EDIT–LONG-Modus wird auch durch @PAR INDEX=ON, und @PAR HEX=ON ausgeschaltet.

Bei Beginn des EDT-Laufs ist für alle Arbeitsdateien @EDIT LONG OFF voreingestellt.

Im EDIT–LONG-Modus wird weder ein mit @PAR SCALE=ON eingeschalteter Spaltenzähler noch eine mit @PAR INFORMATION=ON angeforderte Informationszeile dargestellt. Spaltenzähler und Informationszeile werden erst wieder eingeblendet, wenn der EDIT–LONG-Modus verlassen wird.

Der EDIT–LONG-Modus wird für eine Arbeitsdatei ein- oder ausgeschaltet. Wenn die Arbeitsdatei gleichzeitig in mehreren Datenfenstern am Bildschirm angezeigt wird, gilt in beiden Datenfenstern der gleiche Modus.

Anstelle von @EDIT Format 4 kann mit der gleichen Funktionalität auch die Anweisung @PAR EDIT-LONG benutzt werden. Zusätzlich lässt sich @PAR EDIT-LONG gezielt für eine Arbeitsdatei oder global für alle Arbeitsdateien benutzen und ist auch im L-Modus und damit in EDT-Prozeduren erlaubt.

9.44 @ELIM – Sätze in ISAM-Datei löschen

Diese Anweisung @ELIM löscht den Inhalt einer ISAM-Datei teilweise oder ganz. Wird der gesamte Inhalt gelöscht, bleibt - im Gegensatz zu @UNSAVE - der Dateiname im Katalog bestehen. Außerdem ist es möglich, gleichzeitig in der Arbeitsdatei und auf der Platte zu löschen.

Operation	**Operanden**	**F-Modus, L-Modus**
@ELIM	[file] [(ver)] lines[,...] [**B**OTH]	

file — Name der ISAM-Datei, in der Bereiche gelöscht werden sollen. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen oder die spezielle Angabe `'/'` sein.

Fehlt der Operand `file`, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE-Anweisung). Ist weder ein expliziter lokaler noch globaler @FILE-Eintrag vorhanden, wird die Anweisung @ELIM mit der Fehlermeldung `EDT5484` abgewiesen.

Existiert die angegebene Datei nicht, hat sie den falschen Typ oder ist sie nicht zugreifbar, wird die @ELIM-Anweisung mit einer entsprechenden Meldung abgewiesen.

Wenn der Dateikettungsname `EDTISAM` einer Datei zugeordnet ist, genügt die Angabe `'/'`, um in der Datei zu löschen (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

ver — Versionsnummer der zu löschenden Datei. Stimmt die angegebene Versionsnummer nicht mit der Versionsnummer der Datei überein, wird die Anweisung mit der Meldung `EDT4985` abgewiesen.

lines — Einer oder mehrere Zeilenbereiche, die aus der ISAM-Datei gelöscht werden sollen. Werden symbolische Zeilennummern angegeben, werden deren Werte aus der aktuellen Arbeitsdatei bestimmt, haben also normalerweise nichts mit der Satzstruktur der bei `file` angegebenen Datei zu tun.

Wird für `lines` das Bereichssymbol (Standardwert `&`) angegeben und ist noch dessen Voreinstellung `0.0001-9999.9999` eingestellt (siehe @RANGE-Anweisung), wird der gesamte Inhalt der Datei gelöscht, der Katalogeintrag bleibt aber erhalten.

BOTH — Der bezeichnete Zeilenbereich ist nicht nur in der ISAM-Datei, sondern auch in der Arbeitsdatei zu löschen.

Die Datei wird nur während des Löschvorgangs temporär geöffnet. In Dateien, die nicht die vom EDT angenommenen Standardattribute haben (z.B. keine variable Satzlänge), kann nur dann mit @ELIM gelöscht werden, wenn ein entsprechendes `/SET-FILE-LINK`-Kommando mit dem Dateikettungsnamen `EDTISAM` gegeben worden ist (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Beispiel

```
  23.00 ·····························································
@get 'bsp.elim' noreseq ·····································0001.00:00001(00)
```

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 ····································0001.00:00001(00)
```

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 überein zu stimmen.

```
   1.00 EINS<·····························································
   2.00 ZWEI<·····························································
   3.00 DREI<·····························································
   4.00 VIER<·····························································
   8.00 ACHT<·····························································
   9.00 ··································································
@delete ; @get 'bsp.elim' noreseq·····························0001.00:00001(00)
```

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.

9.45 @END – Verlassen der aktuellen Arbeitsdatei bzw. Beenden des EDT-Laufs

Mit der Anweisung @END wird im L-Modus die aktuelle Arbeitsdatei verlassen. Es wird wieder in die Arbeitsdatei zurückgeschaltet, in der die @PROC-Anweisung gegeben wurde, mit der die jetzt aktuelle Arbeitsdatei eingestellt wurde. Im F-Modus bewirkt @END das Beenden des EDT-Laufs bzw. des Bildschirmdialogs.

Operation	Operanden	F-Modus, L-Modus
@END	[comment]	

comment Der Operand `comment` kann beliebigen Text als Kommentar enthalten. Die Angabe ist nur im L-Modus erlaubt.

Zwischen dem Anweisungsnamen und einem evtl. angegebenen Operanden muss bei dieser Anweisung zwingend mindestens ein Leerzeichen angegeben werden.

Wird im L-Modus die Anweisung @END in der Arbeitsdatei 0 eingegeben, wird die Meldung `EDT4939` ausgegeben. Im Stapelbetrieb und in EDT-Prozeduren wird anschließend die nächste Anweisung gelesen. Im Dialogbetrieb wird nach der Warnungsmeldung `EDT4939` wie bei der @HALT-Anweisung ohne Operanden verfahren mit dem Unterschied, dass selbst bei gesetztem Auftragsschalter 4 die Abfragen `EDT0900` und `EDT0904` nicht unterdrückt werden.

Im F-Modus wird die Anweisung @END unabhängig von der aktuellen Arbeitsdatei stets wie @HALT ohne Operanden behandelt, d.h. der EDT-Lauf bzw. der Bildschirmdialog werden beendet, bei Aufruf des EDT als Unterprogramm wird in das rufende Programm zurückgekehrt (siehe @HALT-Anweisung).

Wurde die aktuelle Arbeitsdatei (ungleich 0) im L-Modus nicht über eine @PROC-Anweisung eingestellt, sondern mit @SETF bzw. implizit nach Wechsel aus dem F-Modus in den L-Modus, wird in die Arbeitsdatei 0 zurückgeschaltet.

Mit @END kann keine Arbeitsdatei zur aktuellen gemacht werden, in der eine @DO-Prozedur läuft (aktive Arbeitsdatei), dies wird mit der Meldung `EDT4959` abgewiesen.

Beispiel 1

```
   1.     @PROC 1 ------------------------------------------------------ (1)
   1.     @ @SET #S1 = DATE
   2.     @ @SET #S2 = TIME      ---------------------------------------- (2)
   3.     @ @PRINT #S1.-#S2 N
   4.     @END ---------------------------------------------------------- (3)
```

(1) Es wird in die Arbeitsdatei 1 umgeschaltet.

(2) In die Arbeitsdatei 1 wird eine EDT-Prozedur eingegeben.

(3) Es wird in die Arbeitsdatei 0 zurückgekehrt. Die in Arbeitsdatei 1 stehende Prozedur kann mit @DO 1 aufgerufen werden.

Beispiel 2

```
   1.     @PROC 7 ------------------------------------------------------ (1)
   1.     @PROC -------------------------------------------------------- (2)
<07>
   1.     @ @SET #S7 = 'HIER SPRICHT PROC 7'
   2.     @ @PRINT #S7
   3.     @PROC 8 ------------------------------------------------------ (3)
   1.     @ @SET #S8 = 'HIER SPRICHT PROC 8'
   2.     @PROC USED
<07>     1.0000 TO    2.0000   ------------------------------------------ (4)
<08>     1.0000 TO    1.0000
   2.     @END --------------------------------------------------------- (5)
   3.     @PROC -------------------------------------------------------- (6)
<07>
   3.     @END --------------------------------------------------------- (7)
   1.     @PROC -------------------------------------------------------- (8)
<00>
```

(1) Es wird in Arbeitsdatei 7 umgeschaltet.

(2) Abfrage der aktuellen Arbeitsdatei.

(3) Es wird in Arbeitsdatei 8 umgeschaltet.

(4) Arbeitsdateien 7 und 8 sind belegt.

(5) Der EDT kehrt wieder in die Arbeitsdatei 7 zurück (von hier aus wurde mit @PROC in die Arbeitsdatei 8 verzweigt).

(6) Die Abfrage der aktuellen Arbeitsdatei bestätigt die Rückkehr in die Arbeitsdatei 7.

(7) Es wird wieder in Arbeitsdatei 0 zurückgekehrt.

(8) Erneute Abfrage nach der aktiven Arbeitsdatei zeigt die Rückkehr in die Arbeitsdatei 0.

9.46 @ERAJV – Jobvariablen löschen

Die Anweisung @ERAJV löscht Einträge von Jobvariablen aus dem Katalog.

Operation	**Operanden**	**F-Modus, L-Modus**
@ERAJV	string [**A**LL]	

string — Zeichenfolge, die die Namen der Jobvariablen bezeichnet, die gelöscht werden sollen. Es sind alle Angaben erlaubt, die auch im BS2000-Makro `ERAJV` gegeben werden dürfen. Die Angabe kann also auch teilqualifiziert oder mit Wildcards erfolgen. Der EDT nimmt keine vollständige Überprüfung der Syntax vor.

Ist keine Jobvariable vorhanden, die die Angabe bezeichnet, wird die Meldung `EDT4982` ausgegeben.

ALL — Ist `ALL` angegeben, werden alle Jobvariablen, die string bezeichnet, ohne Nachfrage aus dem Katalog entfernt.

Ist `ALL` nicht angegeben, und ist mehr als eine Jobvariable vorhanden, die string bezeichnet, wird im Stapelbetrieb die Anweisung nicht ausgeführt. Im Dialogbetrieb gibt der EDT die Anfrage

```
%  EDT0298 ERASE ALL JOBVARIABLES (&00)? REPLY (Y=YES; N=NO)?
```

aus. Ist die Antwort `N`, wird die Meldung `EDT0299` ausgegeben und die Anweisung abgebrochen.

Wird das BS2000-Makro `ERAJV` vom System abgewiesen (z.B. wenn die Jobvariable durch ein Passwort geschützt ist), meldet der EDT den Fehler `EDT4208`.

Ist das Subsystem Jobvariablen-Support nicht installiert, wird die Anweisung mit der Fehlermeldung `EDT5254` abgewiesen. Details zu Jobvariablen können dem Handbuch JV [9] entnommen werden.

9.47 @EXEC – Programm starten

Die Anweisung @EXEC bewirkt, dass die EDT-Sitzung beendet und das angegebene Programm geladen und gestartet wird.

Operation	**Operanden**	**F-Modus, L-Modus**
@EXEC	string	

string — Zeichenfolge, die den Namen des Programms angibt, das geladen und gestartet werden soll. Es wird der Name einer BS2000-Datei erwartet, die das zu ladende Programm enthält. Die Angabe eines Bibliothekselements ist nicht möglich.

Die Anweisung @EXEC gehört zu den sicherheitsrelevanten Anweisungen des EDT (siehe hierzu auch Abschnitt „Zugriffsschutz“ auf Seite 103). In nichtunterbrechbaren Systemprozeduren im Dialog und bei Eingabe von einer Datei (Lesen mit `RDATA` von `SYSDTA` ungleich `SYSCMD`, Abarbeiten einer Start-Prozedur) wird die Anweisung abgewiesen.

Die Anweisung @EXEC führt auf jeden Fall zur Beendigung des EDT, unabhängig davon, ob die angegebene Programmdatei existiert oder ein gültiges Programm enthält.

Bezüglich der Behandlung von ungesicherten Dateien und der damit verbundenen Sicherheitsabfragen wirkt die Anweisung @EXEC wie @HALT (siehe Abschnitt „Beenden des EDT-Laufs“ auf Seite 96). Da der EDT in jedem Fall beendet wird, erfolgen eventuelle Sicherungsabfragen anders als bei @HALT auch dann, wenn die Anweisung im Bildschirmdialog (mit @DIALOG gestartet) eingegeben wurde.

Für den Fall, dass der EDT als Unterprogramm geladen und der Bildschirmdialog des EDT mit @DIALOG eingeschaltet wurde, führt die Anweisung @EXEC nicht zur Fortsetzung des Unterprogramms. Stattdessen wird auch das Benutzerprogramm entladen.

Daher kann bei Aufruf des EDT als Unterprogramm die Anweisung @EXEC für den Benutzer verboten werden. Sie wird dann mit der Meldung `EDT4976` abgewiesen.

Hinweis

Wurde @DIALOG in einer Systemprozedur eingegeben, werden nach @EXEC die restlichen Kommandos der Prozedur evtl. als Eingaben für das neu gestartete Programm interpretiert, was zu unerwünschten Effekten führen kann.

Beispiel

Es wird vorausgesetzt, dass die in der Arbeitsdatei befindlichen Sätze noch nicht gesichert wurden.

```
   1.00 Der EDT soll beendet<·············································
   2.00 und der LMS geladen werden<·······································
   3.00 Dazu wird die Anweisung @EXEC eingegeben<·························
   4.00 ··································································



@exec '$lms'·············································0001.00:00001(00)
```

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.

9.48 @FILE – Dateiname voreinstellen

Mit der Anweisung @FILE kann man für @GET, @READ, @WRITE, @SAVE, @OPEN (Format 2) und @ELIM einen Dateinamen voreinstellen. Man kann sowohl einen Dateinamen voreinstellen, der nur für die aktuelle Arbeitsdatei gilt (expliziter lokaler @FILE-Eintrag), als auch einen Dateinamen, der für alle Arbeitsdateien gilt (globaler @FILE-Eintrag).

Operation	**Operanden**	**F-Modus, L-Modus**
@FILE	[string [(ver)]] [**LOC**AL]	

string — Zeichenfolge, die einen Dateinamen angibt. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen oder die spezielle Angabe `'/'` sein.

ver — Versionsnummer der Datei. Ist `LOCAL` angegeben, so ist die Angabe der Versionsnummer wirkungslos. Die Versionsnummer kann durch eine explizite Angabe in den Dateizugriffsanweisungen überschrieben werden.

LOCAL — Der angegebene Dateiname wird als arbeitsdateispezifischer Dateiname der aktuellen Arbeitsdatei vermerkt (expliziter lokaler @FILE-Eintrag). Ist string nicht angegeben, so wird der lokale @FILE-Eintrag gelöscht.

Wird `LOCAL` nicht angegeben, wird der angegebene Dateiname als globaler @FILE-Eintrag vermerkt. Ist auch `string` nicht angegeben, so wird der globale @FILE-Eintrag gelöscht.

Existiert bei Ausführung der Anweisungen @READ 'file' bzw. @GET 'file' noch kein expliziter lokaler @FILE-Eintrag, wird der angegebene Dateiname zum lokalen Dateinamen (impliziter lokaler @FILE-Eintrag).

In den Anweisungen @WRITE und @SAVE wird der Dateiname zuerst in der Anweisung, dann im expliziten lokalen @FILE-Eintrag, dann im globalen @FILE-Eintrag und zuletzt im impliziten lokalen @FILE-Eintrag gesucht.

In den Anweisungen @GET, @READ und @ELIM wird der Dateiname zuerst in der Anweisung, dann im expliziten lokalen @FILE-Eintrag und zuletzt im globalen @FILE-Eintrag gesucht. Ein impliziter lokaler @FILE-Eintrag wird von den Anweisungen ignoriert.

In der Anweisung @OPEN (Format 2) wird der Dateiname zuerst in der Anweisung und dann im globalen @FILE-Eintrag gesucht. Lokale @FILE-Einträge werden von den Anweisung @OPEN ignoriert.

Hinweis

Der lokale @FILE-Eintrag wird auch gelöscht durch das vollständige Löschen der Arbeitsdatei mit @DELETE (Format 2) oder @DROP sowie durch das Schließen einer real geöffneten Datei mit @CLOSE.

Beispiel

```
  23.00 ·····························································
file 'bsp.file' (*) ; get ·····································0000.00:00001(00)
```

Für die folgenden Anweisungen @GET und @SAVE wird der Dateiname BSP.FILE mit der 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 ·····························································0001.00:00001(00)
```

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) ·····················0003.00:00001(00)
```

```
   3.00 DREI<·····························································
   4.00 VIER<·····························································
   5.00 FUENF<····························································
   6.00 ··································································
%  EDT0902 FILE 'BSP.FILE' VERSION 003
·····························································0003.00:00001(00)
```

9.49 @FSTAT – Ausgeben von BS2000-Kataloginformationen

Mit der Anweisung @FSTAT kann man sich eine Liste von Dateien aus dem BS2000-Katalog ausgeben lassen. Dabei kann man den Ort der Ausgabe festlegen. Wahlweise kann man sich zusätzliche Informationen über die Dateien ausgeben lassen. Die Liste ist nach den Dateinamen alphabetisch sortiert.

Operation	**Operanden**	**F-Modus, L-Modus**
@FSTAT	[{ file / svarex }] [[**T**O] line [(inc)]] [{ **S**HORT / **LONG** [ISO**4**] }]	

file — Bezeichnet die Dateien, die aufgelistet werden sollen. Der Operand `file` muss dem SDF-Datentyp `<partial-filename 1..54 with-wild(80)>` entsprechen.

Die symbolische Bezeichnung `'/'` für eine Datei, für die der `LINK`-Name `EDTSAM` bzw. `EDTISAM` durch das `SET-FILE-LINK`-Kommando vergeben wurde, ist hier nicht zulässig.

Wird keine Datei mit dem angegebenen Namen gefunden, wird die Meldung `EDT5281` ausgegeben.

svarex — Die Auswahl der Dateien, die ausgegeben werden sollen, kann auch durch eine Zeichenfolgevariable (`#S00..#S20`) angegeben werden.

line — Zeilennummer, ab der die Informationen in die aktuelle Arbeitsdatei geschrieben werden. Existierende Zeilen werden dabei überschrieben.

Wenn eine Zeile angelegt wurde, deren Nummer größer ist als die bisherige höchste Zeilennummer, wird die aktuelle Zeilennummer verändert.

Ist `line` nicht angegeben, wird das Ergebnis im L-Modus des Dialogbetriebs nach `SYSOUT` ausgegeben, im Stapelbetrieb nach `SYSLST` ausgegeben und im F-Modus in die Arbeitsdatei `9` geschrieben. Die Arbeitsdatei `9` wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei `9` eine Datei geöffnet ist, wird die Meldung `EDT5189` ausgegeben und die Anweisung nicht ausgeführt.

inc — Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36.
Die aktuelle Schrittweite der Arbeitsdatei wird durch diese Angabe nicht verändert.

SHORT Es wird eine Datei pro Zeile ausgegeben. Es werden nur Dateinamen, ergänzt um die Katalogkennung und die Benutzerkennung, ausgegeben.

LONG Zusätzlich zu den Dateinamen werden weitere Kataloginformationen ausgegeben.

Spalte	Überschrift	Bedeutung
1–7	SIZE	Anzahl der PAM-Seiten
8	P	Datei auf privatem oder gemeinschaftlichem Datenträger (*/␣)
9–62	FILENAME	Dateiname mit CATID und USERID
63–69	LAST PP	Letzte benutzte PAM-Seite
71–78	CR-DATE	Erstellungsdatum (Format YY-MM-DD)
80	S	SHARE-Attribut (Y/N/S)
81	A	ACCESS-Attribut (W/R)
83–86	FCB	FCB-Typ (SAM/ISAM/PAM/BTAM/NONE)
88	R	READ-PASS-Attribut (Y/N)
89	W	WRITE-PASS-Attribut (Y/N)
91–98	CODESET	Zeichensatz

Die Liste ist nach den Dateinamen alphabetisch sortiert.

Bei Ausgabe nach SYSOUT oder SYSLST wird die Ausgabe mit einer Überschrift (siehe Tabelle) eingeleitet. Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne line Operand) wird die Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Bei Ausgabe in eine Arbeitsdatei mit dem Operanden line wird keine Überschrift ausgegeben.

ISO4 Das Erstellungsdatum wird in der Form YYYY-MM-DD ausgegeben. Die nachfolgenden Felder (siehe Tabelle) verschieben sich entsprechend.

Ist weder file noch svarex angegeben, wird eine Liste aller Dateien der eigenen Benutzerkennung ausgegeben.

Wird weder SHORT noch LONG angegeben, werden nur Dateinamen in jeweils einer Zeile ausgegeben. Enthält die Dateispezifikation in file oder svarex eine Katalogkennung, werden die Dateinamen ergänzt um die Katalogkennung und die Benutzerkennung ausgegeben.

Andernfalls werden die Dateinamen in der Form ausgegeben, wie sie in der Anweisung spezifiziert sind.
Eine Ausnahme gilt für teilqualifizierte Dateinamen mit Benutzerkennung. Hier gibt @FSTAT aus Kompatibilitätsgründen eine Liste der Dateinamen ohne Katalogkennung und Benutzerkennung aus.

Eine Ausgabe nach SYSOUT oder SYSLST erfolgt im Zeichensatz, der für diese Systemdateien eingestellt ist. Bei Ausgabe in eine Arbeitsdatei erfolgt die Ausgabe im Zeichensatz der Arbeitsdatei, ist diese leer und hat sie den Zeichensatz *NONE, dann im Zeichensatz EDF041. Im Zielzeichensatz nicht darstellbare Zeichen werden grundsätzlich durch Leerzeichen ersetzt.

Achtung

Für große Dateien reichen die Felder für die reservierte und genutzte Größe bei der Ausgabe im LONG-Modus evtl. nicht aus. In diesem Fall wird dort nichts ausgegeben. Für eine vollständige Ausgabe auch in diesen Fällen sollte nur noch das Kommando @SHOW (Format 1) benutzt werden.

9.50 @GET – Einlesen einer ISAM-Datei

Mit der Anweisung @GET wird eine ISAM-Datei von Platte ganz oder teilweise in die aktuelle Arbeitsdatei eingelesen.

Operation	Operanden	F-Modus, L-Modus
@GET	[file] [(ver)] [lines[,...]] [:cols[,...]:] [**N**ORESEQ]	

file — Name der ISAM-Datei, die eingelesen werden soll. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen oder die spezielle Angabe `'/'` sein.

Besteht noch kein lokaler @FILE-Eintrag für die Arbeitsdatei, so wird der angegebene Dateiname bei Erfolg als impliziter lokaler @FILE-Eintrag eingetragen. Fehlt der Operand `file`, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE-Anweisung). Ist weder ein expliziter lokaler noch ein globaler @FILE-Eintrag definiert (z.B. nur ein impliziter lokaler @FILE-Eintrag), wird die Anweisung @GET mit der Fehlermeldung `EDT5484` abgewiesen.

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Wenn der Dateikettungsname `EDTISAM` einer Datei zugeordnet ist, genügt die Angabe `'/'`, um diese Datei einzulesen (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

ver — Versionsnummer der einzulesenden Datei. Stimmt die angegebene Versionsnummer nicht mit der Versionsnummer der Datei überein, wird die Meldung `EDT0902` ausgegeben, die Datei aber trotzdem eingelesen.

lines — Einer oder mehrere Zeilenbereiche, die aus der ISAM-Datei eingelesen werden sollen. Werden symbolische Zeilennummern angegeben, werden deren Werte aus der aktuellen Arbeitsdatei bestimmt, haben also normalerweise nichts mit der Satzstruktur der bei `file` angegebenen Datei zu tun.

Wird `lines` nicht angegeben, wird die gesamte Datei eingelesen.

Die mit `lines` angegebenen Zeilennummern beziehen sich immer auf die Satzschlüssel, auch wenn diese nicht als Zeilennummern übernommen werden. Es erfolgt also bei Angabe von `lines` immer auch eine Überprüfung auf gültige Satzschlüssel (siehe `NORESEQ`).

cols
: Einer oder mehrere Spaltenbereiche, durch die der einzulesende Bereich jedes Satzes festgelegt wird. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Die Spaltenangaben beziehen sich auf die Zeichen in der einzulesenden Datei.
Bei Dateien, die in einem Unicode-Zeichensatz vorliegen, stimmt das zumeist nicht mit der Byteposition innerhalb eines Satzes überein. Werden Spaltenwerte angegeben, die die Länge eines Satzes überschreiten, so werden dafür Leerzeichen in die Arbeitsdatei eingelesen.
Die Spaltenzählung beginnt erst hinter dem Satzschlüssel.

  Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge eingelesen.

NORESEQ
: Die Zeilennummern werden aus den ISAM-Schlüsseln der eingelesenen ISAM-Datei gebildet. Dabei können Zeilen mit bereits vorhandener Zeilennummer überschrieben werden.

  Der EDT prüft dabei, ob im Satzschlüssel eine gültige Zeilennummer steht. Gültig heißt, dass der Schlüssel nur aus den Ziffern `0` bis `9` bestehen darf, sonst wird die Anweisung @GET mit der Fehlermeldung `EDT4984` abgebrochen. Die bis zu diesem Zeitpunkt gelesenen Sätze werden in die Arbeitsdatei übernommen. Hat ein Satz den Schlüssel 0, dann wird er wie ein Satz mit dem Schlüssel 1 (Zeilennummer `0.0001`) behandelt und die Warnung `EDT2900` ausgegeben.

  Wird `NORESEQ` nicht angegeben, werden die Zeilennummern abhängig von der aktuellen Zeilennummer und der aktuellen Schrittweite vergeben (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Die Datei ist nur während des Lesevorgangs geöffnet. Datensätze, die nur aus dem Schlüssel bestehen, werden als leere Zeilen eingelesen. Ist die einzulesende Datei leer, wird die Warnung `EDT2903` ausgegeben.

Ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz `*NONE`, erhält die Arbeitsdatei den Zeichensatz der einzulesenden Datei. Ist dieser `*NONE`, erhält die Arbeitsdatei den Zeichensatz `EDF03IRV`.

Hat die Arbeitsdatei bereits einen Zeichensatz, dann werden die einzulesenden Sätze vom Zeichensatz der Datei in den Zeichensatz der Arbeitsdatei konvertiert. Enthält die einzulesende Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird das Lesen der Datei abgebrochen und die Fehlermeldung `EDT5453` ausgegeben. Dies gilt auch, falls sich das ungültige Zeichen außerhalb des einzulesenden Spaltenbereichs befindet. Ungültige Zeichen außerhalb des einzulesenden Zeilenbereichs werden hingegen ignoriert.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines `SUBSTITUTION-CHARACTERS` nicht eingelesen werden. In diesem Fall wird das Lesen mit der Meldung `EDT5454` abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Wird versucht, mit @GET eine SAM-Datei einzulesen, gibt der EDT die Fehlermeldung `EDT1902` aus und setzt den Schalter für EDT-Fehler. Er liest die angegebene Datei aber trotzdem ein, indem intern ein @READ für diese Datei ausgeführt wird. Die Operanden `lines` bzw. `NORESEQ` werden in diesem Fall ignoriert.

9.51 @GETJV – Wert einer Jobvariablen lesen

Die Anweisung @GETJV gibt den Wert einer Jobvariablen aus, schreibt ihn in eine Arbeitsdatei oder weist ihn einer Zeichenfolgevariablen zu.

Operation	Operanden	F-Modus, L-Modus
@GETJV	[string] [={ line / svarex }] [,**C**ODE=name]	

string — Zeichenfolge, die einen voll qualifizierten Jobvariablen-Namen angibt. Der Name muss den syntaktischen Regeln für Jobvariablen-Namen genügen, der EDT prüft diese aber nicht vollständig. Ist `string` nicht angegeben, wird die Jobvariable mit dem Kettungsnamen `*EDTLINK` angesprochen. Ist dieser nicht definiert, wird die Meldung `EDT5289` ausgegeben.

line — Nummer der Zeile, in die der Wert der Jobvariablen geschrieben werden soll.

svarex — Zeichenfolgevariablen, in die der Wert der Jobvariablen geschrieben werden soll.

name — Name des Zeichensatzes, in dem der Wert der Jobvariablen interpretiert werden soll. Der Zeichensatz muss zulässig sein, sonst wird die Anweisung mit der Meldung `EDT4980` abgewiesen. Ist der Operand nicht angegeben, wird der Wert der Jobvariablen im Zeichensatz `EDF041` interpretiert (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Ist weder `line` noch `svarex` angegeben, wird der Wert der Jobvariablen im Dialogbetrieb nach `SYSOUT` und im Stapelbetrieb nach `SYSLST` ausgegeben.

Wenn eine Zeile angelegt wird, deren Nummer größer als die bisherige höchste Zeilennummer ist, wird die aktuelle Zeilennummer verändert.

Wenn die Jobvariable nicht vorhanden ist, wird die Meldung `EDT4982` ausgegeben. Wenn sie nicht zugreifbar ist, wird die Meldung `EDT4208` ausgegeben.

Ist die Jobvariable leer, wird als Wert die leere Zeichenfolge genommen.

Enthält die Jobvariable eine ungültige Bytefolge (in Unicode-Zeichensätzen möglich), wird sie nicht eingelesen und die Meldung `EDT5454` ausgegeben.

Erfolgt die Zuweisung an eine Zeichenfolgevariable, so erhält diese den durch `name` explizit oder implizit angegebenen Zeichensatz. Beim Einfügen in eine Arbeitsdatei wird der Wert in den Zeichensatz der Arbeitsdatei konvertiert. Ist die Arbeitsdatei leer und hat noch den

Zeichensatz *NONE, erhält die Arbeitsdatei den durch name explizit oder implizit angegebenen Zeichensatz. Enthält die zuzuweisende Zeichenfolge Zeichen, die nicht in den Zeichensatz der Arbeitsdatei konvertierbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls unterbleibt die Zuweisung und die Fehlermeldung EDT5453 wird ausgegeben.

Ist das Subsystem Jobvariablen-Support nicht installiert, wird die Anweisung mit der Fehlermeldung EDT5254 abgewiesen. Details zu Jobvariablen können dem Handbuch JV [9] entnommen werden.

9.52 @GETLIST – Einlesen von Elementen einer Listenvariablen

Die Anweisung @GETLIST schreibt Elemente einer Listenvariablen in die aktuelle Arbeitsdatei.

Operation	Operanden	F-Modus, L-Modus
@GETLIST	string [lines[,...]] [:cols[,...]:] [,**C**ODE=name]	

string — Zeichenfolge, die den Namen einer S-Listenvariablen angibt.
Der Name muss den syntaktischen Regeln für einen S-Variablen-Namen genügen, diese werden aber vom EDT nicht vollständig überprüft. Ist der Name länger als 246 Zeichen, wird die Anweisung mit der Meldung `EDT3174` abgebrochen.

lines — Einer oder mehrere Zeilenbereiche, die die Listenelemente angeben, die übernommen werden sollen.

Es werden nur die Elemente der Listenvariablen eingelesen, die durch den Zeilenbereich gekennzeichnet sind. Hierbei erfolgt eine Zuordnung von Elementnummern und Zeilennummern, wobei `0.0001` für das 1. Element in der Liste steht, `0.0002` für das 2. Element, usw.

Ist zu einer angegebenen Zeilennummer kein Element vorhanden, wird die Angabe ignoriert.

Ist `lines` nicht angegeben, werden alle Elemente eingelesen.

cols — Einer oder mehrere Spaltenbereiche, der S-Variablen, die eingelesen werden sollen. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Dabei wird die Spalte `n` dem `n`-ten Zeichen zugeordnet. Alle spezifizierten Zeichen werden in der Reihenfolge der Spaltenangabe ggf. auch mehrfach aneinandergefügt und das Ergebnis in die Arbeitsdatei eingefügt. Ist das Ergebnis größer als 32768 Zeichen, wir die Anweisung mit der Meldung `EDT5474` abgebrochen.

Enthält ein Listenelement weniger Zeichen als die angegebene Spalte, wird dafür ein Leerzeichen eingefügt.

Wird kein Spaltenbereich angegeben, wird jedes Element in voller Länge eingelesen.

name — Name des Zeichensatzes, in dem der Wert der S-Variablen interpretiert werden soll. Der Zeichensatz muss zulässig sein, sonst wird die Anweisung mit der Meldung `EDT4980` abgewiesen. Ist der Operand nicht angegeben, wird der Wert der S-Variablen im Zeichensatz `EDF041` interpretiert (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Ist die angegebene S-Variable keine Liste, wird die Meldung EDT4910 ausgegeben. Hat der Wert eines Listenelements nicht den Typ STRING, wird die Anweisung mit der Meldung EDT5343 abgebrochen. Hat die Liste kein Element, wird die Meldung EDT5340 ausgegeben.

Die Vergabe der Zeilennummern erfolgt nach dem Verfahren Einfügen bei der aktuellen Zeilennummer (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Wird die maximale Zeilennummer erreicht, wird die Anweisung abgebrochen und die Meldung EDT5252 ausgegeben.

Enthält ein Listenelement eine ungültige Bytefolge (in Unicode-Zeichensätzen möglich), wird die Anweisung abgebrochen und die Meldung EDT5454 ausgegeben.

Beim Einfügen werden die Werte in den Zeichensatz der Arbeitsdatei konvertiert. Ist die Arbeitsdatei leer und hat noch den Zeichensatz *NONE, erhält die Arbeitsdatei den durch name explizit oder implizit angegebenen Zeichensatz. Enthält die zuzuweisende Zeichenfolge Zeichen, die nicht in den Zeichensatz der Arbeitsdatei konvertierbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls unterbleibt die Zuweisung und die Fehlermeldung EDT5453 wird ausgegeben.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Details zu S-Variablen können dem Handbuch SDF-P [7] entnommen werden.

9.53 @GETVAR – Lesen einer S-Variablen

Die Anweisung @GETVAR gibt den Wert einer S-Variablen aus, schreibt ihn in eine Arbeitsdatei oder weist ihn einer Variablen zu.

Operation	Operanden	F-Modus, L-Modus
@GETVAR	{ string [= { line / svarex / ivar }] / **SYSEDT** } [,**C**ODE=name]	

string — Zeichenfolge, die einen gültigen S-Variablen-Namen angibt. Der Name muss den syntaktischen Regeln für einen S-Variablen-Namen genügen, diese werden aber vom EDT nicht vollständig überprüft.

line — Nummer der Zeile, in die der Wert der S-Variablen geschrieben werden soll.

Hat der Wert der S-Variablen nicht den Typ `STRING`, wird die Anweisung mit der Meldung `EDT5342` abgewiesen.

svarex — Zeichenfolgevariable, in die der Wert der S-Variablen geschrieben werden soll.

Hat der Wert der S-Variablen nicht den Typ `STRING`, wird die Anweisung mit der Meldung `EDT5342` abgewiesen.

ivar — Ganzzahlvariable (`#I0..#I20`), in die der Inhalt der S-Variablen übernommen werden soll.

Hat der Wert der S-Variablen nicht den Typ `INTEGER`, wird die Anweisung mit der Meldung `EDT5342` abgewiesen.

SYSEDT — Den Zeichenfolgevariablen `#S00..#S20` werden die Inhalte der S-Variablen `SYSEDT-S00..SYSEDT-S20` zugewiesen, falls sie existieren und ihr jeweiliger Wert den Typ `STRING` hat. Für nicht existierende S-Variable, S-Variable ohne Wert oder mit anderem Typ wird kein Fehler gemeldet, sondern die
zugehörige Zeichenfolgevariable wird nicht verändert.

name — Name des Zeichensatzes, in dem der Wert der S-Variablen interpretiert werden soll. Ist der Wert der S-Variablen nicht vom Typ `STRING` wird die Angabe ignoriert. Der Zeichensatz muss zulässig sein, sonst wird die Anweisung mit der Meldung `EDT4980` abgewiesen. Ist der Operand nicht angegeben, wird der Wert der S-Variablen im Zeichensatz `EDF041` interpretiert (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Ist weder line noch svarex noch ivar angegeben, wird der Wert der S-Variablen im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST ausgegeben.

Ist die S-Variable nicht vorhanden, wird die Meldung EDT5274 ausgegeben, hat sie keinen Wert, wird die Meldung EDT5340 ausgegeben (außer bei SYSEDT).

Wenn eine Zeile angelegt wird, deren Nummer größer als die bisherige höchste Zeilennummer ist, wird die aktuelle Zeilennummer verändert.

Enthält die Variable eine ungültige Bytefolge (in Unicode-Zeichensätzen möglich), wird sie nicht eingelesen und die Meldung EDT5454 ausgegeben.

Erfolgt die Zuweisung an eine Zeichenfolgevariable, so erhält diese den durch name explizit oder implizit angegebenen Zeichensatz.
Beim Einfügen in eine Arbeitsdatei wird der Wert in den Zeichensatz der Arbeitsdatei konvertiert. Ist die Arbeitsdatei leer und hat noch den Zeichensatz *NONE, erhält die Arbeitsdatei den durch name explizit oder implizit angegebenen Zeichensatz.
Enthält die zuzuweisende Zeichenfolge Zeichen, die nicht in den Zeichensatz der Arbeitsdatei konvertierbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls unterbleibt die Zuweisung und die Fehlermeldung EDT5453 wird ausgegeben.

Details zu S-Variablen können dem Handbuch SDF [6] entnommen werden.

9.54 @GOTO – Sprunganweisung in Prozeduren

Mit der Anweisung @GOTO wird in einer @DO-Prozedur ein unbedingter Sprung zu der angegeben Zeile ausgeführt.

Operation	**Operanden**	**@PROC**
@GOTO	line	

line Der Operand `line` bezeichnet die Zeilennummer zu der gesprungen werden soll.

Die Anweisung @GOTO ist nur innerhalb von @DO-Prozeduren erlaubt. Wird @GOTO außerhalb einer @DO-Prozedur gegeben, wird dies mit der Meldung `EDT4942` zurückgewiesen. In einer @INPUT-Prozedur wird nach Ausgabe der Meldung mit der Anweisung fortgesetzt, die auf die fehlerhafte @GOTO-Anweisung folgt.

Wenn `line` eine Zeilennummervariable ist, die zum Zeitpunkt der Ausführung des Sprungs den Wert `0.0000` hat (mit dem die Zeilennummervariablen nach Starten des EDT vorbelegt sind), wird die @GOTO-Anweisung mit der Meldung `EDT4932` abgewiesen und die Prozedur mit der Anweisung fortgesetzt, die auf die fehlerhafte @GOTO-Anweisung folgt.

Die mit @GOTO angesprungene Zeile muss in der zugehörigen Prozedur existieren. Wird versucht eine Zeile anzuspringen, die nicht existiert, wird die Fehlermeldung `EDT4974` ausgegeben und die Prozedur mit der Anweisung fortgesetzt, die auf die fehlerhafte @GOTO-Anweisung folgt.

Sollen in EDT-Prozeduren Zeilen über @GOTO angesprungen werden, ist es bei diesen Zeilen sinnvoll, mit der Anweisung @SET (Format 6) immer die Zeilennummer explizit festzulegen, damit sich beim Einfügen oder Löschen von Anweisungen die Nummern der anzuspringenden Zeilen nicht implizit ändern.

Hinweis

Die Verwendung symbolischer Zeilennummern ist nicht zu empfehlen, weil diese sich immer auf die aktuelle Arbeitsdatei und damit nicht auf die Prozedur-Arbeitsdatei beziehen.

Ist die erste Anweisung einer Prozedur die @PARAMS-Anweisung, so kann diese nicht mit @GOTO angesprungen werden.

Beispiel

```
   1.     @SET #I3 = 1 ------------------------------------------------- (1)
   1.     @PROC 3
   1.     @1 ----------------------------------------------------------- (2)
   1.     @ @IF #I3 > 5 : @RETURN -------------------------------------- (3)
   2.     @ @STATUS = #I3
   3.     @ @SET #I3 = #I3+1
   4.     @ @GOTO 1
   5.     @END
   1.     @DO 3 -------------------------------------------------------- (4)
#I03= 0000000001
#I03= 0000000002
#I03= 0000000003
#I03= 0000000004
#I03= 0000000005
   1.
```

(1) Der Ganzzahlvariablen #I3 wird der Wert 1 zugewiesen.

(2) Angabe der Zeilenummer für die Zeile, die über @GOTO angesprungen wird.

(3) Bei Ausführung der Prozedur in der Arbeitsdatei 3 soll der Wert, den die Ganzzahlvariable #I3 hat, solange um 1 erhöht und ausgegeben werden, bis er größer als 5 ist. Realisiert wird dies durch eine Schleife. Am Ende der Schleife wird mit @GOTO zum Schleifenanfang gesprungen.

(4) Die Prozedur in Arbeitsdatei 3 wird ausgeführt.

9.55 @HALT – Beenden des EDT

Die Anweisung @HALT beendet den EDT-Lauf, den Bildschirmdialog nach @DIALOG bzw. den EDT als Unterprogramm mit oder ohne Übergabe eines Textes an das aufrufende Programm (siehe Abschnitt „Beenden des EDT-Laufs“ auf Seite 96 zu den allgemeinen Wirkungen der EDT-Beendigung).

Operation	Operanden	F-Modus, L-Modus
@HALT	[**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 der EDT als Unterprogramm aufgerufen, wird die gesamte Zeichenfolge, beginnend ab dem ersten Zeichen ungleich Leerzeichen nach @HALT als Meldungstext an das rufende Programm übergeben (siehe auch Operand message). Beginnt diese Zeichenfolge mit 'ABNORMAL' wird ein spezieller Returncode (0008002C statt 00080000) gesetzt.

message Zeichenfolge, die bei Aufruf des EDT als Unterprogramm an das aufrufende Programm übergeben wird. Dieser Operand darf nur angegeben werden, wenn der EDT als Unterprogramm aufgerufen wird.

Beginnt diese Zeichenfolge mit 'ABNORMAL' wird ein spezieller Returncode (0008002C statt 00080000) gesetzt.

Zwischen dem Anweisungsnamen und einem evtl. angegebenen Operanden muss bei dieser Anweisung zwingend mindestens ein Leerzeichen angegeben werden.

Die Anweisung @HALT bewirkt das Beenden des EDT-Laufs im Dialog- oder Stapelbetrieb, wenn der EDT mit dem Kommando /START-EDT bzw. /START-EDTU oder den äquivalenten /START-PROGRAM-Kommandos als Hauptprogramm gestartet wurde (siehe Abschnitt „Starten des EDT“ auf Seite 91) und sich nicht im Bildschirmdialog nach @DIALOG befindet. Der EDT-Lauf wird ebenfalls beendet, wenn @HALT über die CMD-Funktion der Unterprogramm-Schnittstelle (siehe Handbuch Unterprogramm-Schnittstellen [1]) eingegeben wurde.

Wurde im Dialogbetrieb der Bildschirmdialog mit der Anweisung @DIALOG aus einer Systemprozedur bzw. über die Unterprogramm-Schnittstelle eingeleitet, wird durch @HALT nur der Bildschirmdialog beendet und die Systemprozedur bzw. das aufrufende Programm

fortgesetzt. Dabei bleibt der Inhalt der Arbeitsdateien erhalten, weshalb auch keine Sicherungsabfrage erfolgt (siehe unten). In diesem Fall hat die Angabe von ABNORMAL keine besondere Wirkung, außer dass das aufrufende Programm darüber informiert wird.

Existieren noch ungesicherte Arbeitsdateien, so werden im Dialogbetrieb vor dem Beenden des EDT nach der Meldung EDT0900 die Nummern der Arbeitsdateien mit ungesicherten Daten ausgegeben.

Zusätzlich wird (falls vorhanden) für jede dieser Arbeitsdateien der lokale @FILE-Eintrag (siehe Anweisungen @FILE, @READ, @GET und @OPEN, Format 2) bzw. der Name einer geöffneten Datei oder der Bibliotheks- und Elementname eines geöffneten Bibliothekselements (siehe Anweisungen @OPEN und @XOPEN) ausgegeben.

Danach folgt im Dialogbetrieb die Anfrage an den Benutzer:

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

Bei Antwort N erscheint im F-Modus das Arbeitsfenster, im L-Modus die Eingabeaufforderung erneut. Der Benutzer kann ungesicherte Arbeitsdateien schließen bzw. zurück schreiben. Bei Antwort Y akzeptiert der Anwender, dass die ungesicherten Arbeitsdateien verloren gehen. 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 (siehe auch den Abschnitt „Beenden des EDT-Laufs“ auf Seite 96).

Beispiel

```
1.     @HALT
%  EDT0900 EDITED FILE(S) NOT SAVED!
  LOCAL FILE ( 0) :
  LOCAL FILE ( 1) :
  LOCAL FILE ( 4) : L= EDT164
                    E= HALT(001),X
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)
```

Der EDT wird im L-Modus mit @HALT beendet. Da noch ungesicherte Arbeitsdateien vorhanden sind, wird die Meldung EDT0900 und eine Liste der Arbeitsdateien ausgegeben.

9.56 @HEX – Hexadezimal-Modus einstellen

Die Anweisung @HEX schaltet den Hexadezimal-Modus für die aktuelle Arbeitsdatei ein oder aus. Im Hexadezimal-Modus werden alle Sätze in abdruckbarer und in hexadezimaler Form am Bildschirm dargestellt.

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

ON — Schaltet den Hexadezimal-Modus ein (Standardwert).
Der EDIT-LONG-Modus wird dabei implizit abgeschaltet.

OFF — Schaltet den Hexadezimal-Modus aus.

Das Layout des Hexadezimal-Modus ist ausführlich im Abschnitt „F-Modus“ auf Seite 105 beschrieben.

Bei Beginn des EDT-Laufs ist den Hexadezimal-Modus für alle Arbeitsdateien ausgeschaltet.

Der Hexadezimal-Modus wird für eine Arbeitsdatei ein- oder ausgeschaltet. Wenn die Arbeitsdatei gleichzeitig in mehreren Datenfenstern am Bildschirm angezeigt wird, gilt in beiden Datenfenstern der gleiche Modus.

Ist das Datenfenster bei geteiltem Bildschirm so klein, dass nicht wenigstens eine Datenzeile mit ihren Hexzeilen angezeigt werden kann, wird die Meldung EDT2404 ausgegeben, der Hexadezimal-Modus aber eingeschaltet. Der Anwender kann dann sein Datenfenster so vergrößern, dass die Hexzeilen auch angezeigt werden können.

Anstelle von @HEX kann mit der gleichen Funktionalität auch die Anweisung @PAR HEX benutzt werden. Zusätzlich lässt sich @PAR HEX gezielt für eine Arbeitsdatei oder global für alle Arbeitsdateien benutzen und ist auch im L-Modus und damit in EDT-Prozeduren erlaubt.

9.57 @IF (Format 1) – Abfrage von Fehlerschaltern

Mit diesem Format der @IF-Anweisung kann in EDT-Prozeduren und im L-Modus geprüft werden, 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
@IF	{ **E**RRORS | **N**O [**E**RRORS] | **D**MS [**E**RRORS] | **N**O **D**MS [**E**RRORS] } :[text]	

ERRORS Die Bedingung ist erfüllt, wenn der EDT-Fehlerschalter gesetzt ist.

NO ERRORS Die Bedingung ist erfüllt, wenn der EDT-Fehlerschalter *nicht* gesetzt ist.

DMS ERRORS
Die Bedingung ist erfüllt, wenn der DVS-Fehlerschalter gesetzt ist.

NO DMS ERRORS
Die Bedingung ist erfüllt, wenn der DVS-Fehlerschalter *nicht* gesetzt ist.

text EDT-Anweisung oder Datenzeile. Falls die Bedingung erfüllt ist, wird die Zeichenfolge so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

Wird `text` nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Hinweis

Soll eine bestimmte Anweisung geprüft werden, müssen vor der zu prüfenden Anweisung die Fehlerschalter zurückgesetzt werden (siehe @RESET). Die Anweisung @IF kann sonst ein unerwünschtes Ergebnis liefern, da auch bereits vorhergegangene Anweisungen den EDT- oder DVS-Schalter gesetzt haben könnten.

Die Anweisung @IF ERRORS sollte nicht zur Trefferabfrage nach @ON eingesetzt werden. Es ist dazu @IF Format 3 zu verwenden.

Die Verwendung von @IF mit @RETURN als Anweisung außerhalb von Prozeduren führt möglicherweise zur Beendigung des EDT (siehe @RETURN-Anweisung).

9.58 @IF (Format 2) – Vergleich von Zeichenfolgen, Zeilennummern und Zahlen

Dieses Format der @IF-Anweisung kann in EDT-Prozeduren benutzt werden, um Zeichenfolgen, Zeilennummern oder Ganzzahlen miteinander zu vergleichen. Die Operanden dürfen dabei sowohl explizit (als Literale) als auch über eine Referenz (Zeilennummer bzw. EDT-Variable) angegeben werden.

Trifft die Vergleichsbedingung zu, wird die angegebene Zeichenfolge als Eingabe abgearbeitet. Ist die Bedingung nicht erfüllt, bleibt die Anweisung wirkungslos.

Operation	Operanden	L-Modus
@IF	{ [**S**] string1 rel string2 line1 rel line2 [**I**] int1 rel int2 } :[text]	

S — Das Schlüsselwort `S` wird nur dann zwingend benötigt, wenn in `string1` und `string2` Zeilennummern bzw. Zeilennummervariablen ohne Spaltenangaben stehen und legt dann fest, dass der *Inhalt* der der durch die Zeilennummern bzw. Zeilennummervariablen bestimmten Zeilen verglichen werden soll und nicht die Zeilennummern selbst.

Beispielsweise werden mit @IF #L1=#L2 die Zeilennummern in `#L1` und `#L2` miteinander verglichen. Will man jedoch die Inhalte der durch `#L1` und `#L2` bestimmten Zeilen miteinander vergleichen, muss @IF S #L1=#L2 angegeben werden.

string1, string2 — Die miteinander zu vergleichenden Zeichenfolgen.

line1, line2 — Die miteinander zu vergleichenden Zeilennummern. Es werden nicht die Zeileninhalte verglichen.

I — Das Schlüsselwort `I` muss nur dann angegeben werden, wenn für `int1` explizit eine Zahl (ein Literal) angegeben wird, sonst erkennt der EDT nicht, ob es sich um eine Zeilennummer oder um eine Ganzzahl handelt.

int1, int2 — Die miteinander zu vergleichende Ganzzahlen.

Es kann jeweils eine ganze (positive oder negative) Zahl oder eine Ganzzahlvariable (`#I0..#I20`) angegeben werden.

rel — Legt die Vergleichsrelation fest:

Symbol			Erläuterung
>	bzw.	GT	größer als
<	bzw.	LT	kleiner als
>=	bzw.	GE	größer oder gleich
<=	bzw.	LE	kleiner oder gleich
=	bzw.	EQ	gleich
<>	bzw.	NE	ungleich

text — EDT-Anweisung oder Datenzeile. Falls die Bedingung erfüllt ist, wird die Zeichenfolge so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

Wird `text` nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Aus Kompatibilitätsgründen wird die früher mögliche Angabe von `GOTO` oder `RETURN` ohne Doppelpunkt in Prozeduren weiter unterstützt.

Der Vergleich zweier Zeichenfolgen hängt zum einen vom Zeichensatz ab, in dem die Zeichenfolgen codiert sind, zum anderen von der Länge der Zeichenfolgen (Zeichenfolgen der Länge Null sind zugelassen).

Den beiden Operanden `string1` und `string2` kann in jedem Fall ein Zeichensatz entsprechend ihrer Herkunft zugeordnet werden. Wenn die beiden so bestimmten Zeichensätze gleich sind, findet der Vergleich in diesem gemeinsamen Zeichensatz binär statt.
Sind die beiden Zeichensätze verschieden, aber beides EBCDIC 7/8-Bit-Zeichsätze, erfolgt der Vergleich wie bisher binär.
Andernfalls werden beide Zeichenketten intern nach `UTF16` konvertiert und der Vergleich findet in `UTF16` statt.

Nach der ggf. notwendigen Umcodierung werden von links nach rechts die sich entsprechenden Zeichen der beiden Zeichenfolgen miteinander verglichen. Auf diese Weise erreicht man entweder ein ungleiches Zeichenpaar, oder aber das Ende einer oder beider Zeichenfolgen. Für den Fall der Ungleichheit an einer Stelle steht die Ungleichheit der beiden Zeichenfolgen fest. Der EDT interpretiert die beiden ungleichen Zeichen gemäß ihrem

Zeichensatz als Dualzahlen. Größer ist die Zeichenfolge, deren Zeichen die größere Dualzahl darstellt. Wurde kein ungleiches Zeichenpaar gefunden, wird die längere der beiden Zeichenfolgen als die größere betrachtet. Sind beide Zeichenfolgen gleich lang und wurde kein ungleiches Zeichenpaar gefunden, sind die Zeichenfolgen gleich. Sind die beiden Zeichenfolgen unterschiedlich lang, kann also niemals Gleichheit der beiden Zeichenfolgen vorliegen.

Hinweis

Beim Vergleich von Zeilennummern bezeichnen sowohl `%+3L` als auch `%+3` gültige Zeilennummern. Bei einer nachfolgenden Vergleichsrelation `LE` kann das zu Interpretationsschwierigkeiten führen. Deshalb ist es generell empfehlenswert, die mathematischen Zeichen für die Vergleichsrelationen zu benutzen.
Auch wenn man Spaltenangaben beim Operanden `string2` verwendet, kann es zu Interpretationsschwierigkeiten mit dem Einleiter des Operanden `text` kommen. Es ist bei der Notation in diesen Fällen besondere Sorgfalt auf eine eindeutige Interpretierbarkeit zu legen. Insbesondere ist es Pflicht, die Spaltenangabe mit Doppelpunkt abzuschließen (sonst optional).

Aus Kompatibilitätsgründen wird das von XHCS festgelegte *Sortiergewicht* beim Vergleich der Zeichen nicht berücksichtigt. Dies gilt auch für Unicode-Zeichensätze. Eine Sortierung, beispielsweise mit dem Programm `SORT`, kann also eine andere Reihenfolge liefern als es das Ergebnis einer @IF-Abfrage im EDT nahe legen würde.

Die Verwendung von @IF mit @RETURN als Anweisung außerhalb von Prozeduren führt möglicherweise zur Beendigung des EDT (siehe @RETURN-Anweisung).

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' --------------- (1)
   4.     @SET #I9 = 2
   4.     @PROC 3
   1.     @ @IF S %+#I9 = $-2L : @GOTO 4 ------------------------------- (2)
   2.     @ @PRINT #S1 N
   3.     @ @RETURN
   4.     @ @PRINT #S0 N
   5.     @END
   4.     @DO 3 -------------------------------------------------------- (3)
ERSTE ZEILE = LETZTE ZEILE
   4.     @ON 1 DELETE 'NICHT'
   4.     @DO 3 -------------------------------------------------------- (4)
ERSTE ZEILE UNGLEICH LETZTE ZEILE
   4.
```

(1) Die Zeichenfolgevariablen #S0 und #S1 und die Ganzzahlvariable #I9 werden mit Inhalt versehen.

(2) Bei Ausführung der Arbeitsdatei 3 werden an dieser Stelle nicht Zeilennummern, sondern Zeileninhalte miteinander verglichen.

(3) 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.

(4) 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' ------------------------------------------------ (1)
   1.     @PROC 4
   1.     @ @PRINT #S4 N ------------------------------------------------ (2)
   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.
```

(1) Die Zeichenfolgevariable #S4 erhält das Zeichen M als Inhalt.

(2) 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 ------------------------------------ (1)
2.     @ @CREATE !: '*' * 20
3.     @ @CONTINUE
4.     @END
9.     @DO 6,!=%,$ ---------------------------------------------------- (2)
9.     @PRINT
1.0000 ********************
2.0000 WER
3.0000 ********************
4.0000 WILL
5.0000 ********************
6.0000 HIER
7.0000 ********************
8.0000 MITMACHEN?
9.
```

(1) In der Arbeitsdatei 6 werden die Zeilennummern über den Schleifenzähler ! angesprochen. Sollte der Inhalt der gerade über den Zähler ! angesprochenen Zeile von ABC verschieden sein, soll sich an diesem Zeileninhalt nichts ändern. Im anderen Fall soll der Zeileninhalt durch ******************** ersetzt werden.

(2) Arbeitsdatei 6 wird ausgeführt. Hierbei sollen alle Zeilen der aktuellen Arbeitsdatei nacheinander durch den Schleifenzähler ! 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' ---------------------------------- (1)
   4.     @SET #I9 = 1
   4.     @PROC 1 ---------------------------------------------------------- (2)
   1.     @ @IF %+#I9 = $-1L : @GOTO 4 ------------------------------------ (3)
   2.     @ @PRINT #S1 N
   3.     @ @RETURN
   4.     @ @PRINT #S0 N
   5.     @END
   4.     @DO 1 ------------------------------------------------------------ (4)
VERGLEICH POSITIV
   4.     @SET #I9 = 2
   4.     @DO 1 ------------------------------------------------------------ (5)
VERGLEICH NEGATIV
   4.
```

(1) Die Zeichenfolgevariablen `#S0` und `#S1` werden mit Inhalt versehen. Der Ganzzahlvariablen `#I9` wird der Wert 1 zugewiesen.

(2) Arbeitsdatei 1 wird geöffnet.

(3) 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.

(4) 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.

(5) 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 ---------------------------------- (1)
2.     @ @CREATE $+1: 'ODER AUCH NICHT'
3.     @ @PRINT
4.     @END
4.     @$-1
2.     @DO 2 ------------------------------------------------------------ (2)
2.     @1
1.     @DO 2 ------------------------------------------------------------ (3)
1.0000 BITTE NICHT LACHEN
2.0000 WEGEN DIESES BEISPIELS
3.0000 ES IST ZU EINFACH
4.0000 ODER AUCH NICHT
1.
```

(1) Ist bei Ausführung der Arbeitsdatei 2 die hier genannte Relation nicht erfüllt (<> steht für ungleich), wird der Prozedurablauf an dieser Stelle abgebrochen.

(2) 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.

(3) 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 ------------------------------------------------- (1)
   1.     @PROC 7
   1.     @ @IF #I3 > 5 : @RETURN
   2.     @ @STATUS = #I3 ---------------------------------------------- (2)
   3.     @ @SET #I3 = #I3+1
   4.     @ @GOTO 1
   5.     @END
   1.     @DO 7 -------------------------------------------------------- (3)
#I03= 0000000001
#I03= 0000000002
#I03= 0000000003
#I03= 0000000004
#I03= 0000000005
   1.
```

(1) Der Ganzzahlvariablen #I3 wird der Wert 1 zugewiesen.

(2) 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.

(3) Die Arbeitsdatei 7 wird ausgeführt.

9.59 @IF (Format 3) – Abfrage von @ON-Treffern bzw. des Arbeitsdatei-Status

Mit diesem Format der @IF-Anweisung kann in EDT-Prozeduren geprüft werden, ob der EDT bei der letzten Ausführung von @ON einen Treffer festgestellt hat bzw. ob die aktuelle Arbeitsdatei leer ist. Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.

Operation	Operanden	L-Modus
@IF	{ **.T**RUE**.** [rel col] / **.F**ALSE**.** / **.E**MPTY**.** } :[text]	

.TRUE. Es wird verzweigt, wenn bei der letzten Ausführung von @ON in der aktuellen Arbeitsdatei ein Treffer festgestellt wurde.

Wurden `rel` und `col` angegeben, wird die Bedingung in der Weise verschärft, dass die Spaltennummer, in der der erste festgestellte Treffer begann, mit der durch `col` angegebenen Spaltennummer gemäß `rel` verglichen wird. Die Bedingung gilt nur dann als erfüllt, wenn dieser Vergleich positiv ausfällt.

rel Bestimmt die Vergleichsoperation für die Spaltennummern (siehe oben):

Symbol			Erläuterung
>	bzw.	GT	größer als
<	bzw.	LT	kleiner als
>=	bzw.	GE	größer oder gleich
<=	bzw.	LE	kleiner oder gleich
=	bzw.	EQ	gleich
<>	bzw.	NE	ungleich

col Spaltennummer, mit der die Nummer der Spalte verglichen wird, in der bei der letzten @ON-Anweisung in der aktuellen Arbeitsdatei der erste Treffer begann.

.FALSE. Es wird verzweigt, wenn bei der letzten Ausführung von @ON in der aktuellen Arbeitsdatei *kein* Treffer festgestellt wurde.

.EMPTY. Es wird verzweigt, wenn die aktuelle Arbeitsdatei leer ist. Eine Arbeitsdatei ist leer, wenn sie keine Datensätze enthält.

text EDT-Anweisung oder Datenzeile. Falls die Bedingung erfüllt ist, wird die Zeichenfolge so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

Wird `text` nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Aus Kompatibilitätsgründen wird die früher mögliche Angabe von `GOTO` oder `RETURN` ohne Doppelpunkt weiter unterstützt.

Hinweis

Die Verwendung von @IF mit @RETURN als Anweisung außerhalb von Prozeduren führt möglicherweise zur Beendigung des EDT (siehe @RETURN-Anweisung).

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 ---------------------------------------- (1)
3.     @ @CREATE !: '*'*20
4.     @ @CONTINUE
5.     @END
5.     @DO 8,!=%,$ ---------------------------------------------------- (2)
5.     @PRINT
1.0000 WER
2.0000 ********************
3.0000 ********************
4.0000 ********************
5.
```

(1) In der Arbeitsdatei 8 werden die Zeilennummern über den Schleifenzähler `!` 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 20 Sternchen ersetzt werden.

(2) Die Arbeitsdatei 8 wird ausgeführt. Dabei sollen alle Zeilen der Hauptdatei nacheinander durch den Schleifenzähler ! 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 ------------------------------------------------------ (1)
4.     @ @SUFFIX ! WITH ' DENN UNBEDINGT'
5.     @ @CONTINUE
6.     @END
5.     @DO 9,!=%,$ ---------------------------------------------------- (2)
5.     @PRINT
1.0000 WER
2.0000 WILL
3.0000 HIER DENN UNBEDINGT
4.0000 MITMACHEN?
5.
```

(1) In der Prozedur in Arbeitsdatei 9 werden die Zeilennummern über den Schleifenzähler ! 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.

(2) Die Prozedur von Arbeitsdatei 9 wird ausgeführt. Dabei werden alle Zeilen der Hauptdatei nacheinander durch den Schleifenzähler ! angesprochen.

9.60 @IF (Format 4) – Abfrage von Auftrags- und Benutzerschaltern

Mit diesem Format der @IF-Anweisung kann in EDT-Prozeduren geprüft werden, welche Auftrags- bzw. Benutzerschalter ein- oder ausgeschaltet sind (siehe auch @SETSW und Abschnitt „Auftragsschalter“ auf Seite 102). Abhängig davon wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.

Operation	Operanden	L-Modus
@IF	{ **ON** / **OF**F } = [**U**] int :[text]	

ON — Es wird verzweigt, wenn der angegebene Schalter gesetzt ist.

OFF — Es wird verzweigt, wenn der angegebene Schalter *nicht* gesetzt ist.

U — Legt fest, dass ein Benutzerschalter geprüft werden soll. Wird `U` nicht angegeben, wird ein Auftragsschalter geprüft.

int — Nummer des Schalters (`0..31`), dessen Stellung geprüft werden soll.

Falls vor der Schalternummer das Schlüsselwort `U` angegeben ist, wird statt des Auftragsschalters `int` der Benutzerschalter `int` der eigenen Kennung geprüft.

text — EDT-Anweisung oder Datenzeile. Falls die Bedingung erfüllt ist, wird die Zeichenfolge so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

Wird `text` nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Aus Kompatibilitätsgründen wird die früher mögliche Angabe von `GOTO` oder `RETURN` ohne Doppelpunkt in Prozeduren weiter unterstützt.

Hinweis

Die Verwendung von @IF mit @RETURN als Anweisung außerhalb von Prozeduren führt möglicherweise zur Beendigung des EDT (siehe @RETURN-Anweisung).

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             ---------------------------------------- (1)
   4.     @ @PRINT #S3 N
   5.     @END
   1.     @SETSW OFF = 15 ------------------------------------------------ (2)
   1.     @DO 8 ---------------------------------------------------------- (3)
SCHALTER 15 IST AUS
   1.     @SETSW ON = 15 ------------------------------------------------- (4)
   1.     @DO 8 ---------------------------------------------------------- (5)
SCHALTER 15 IST AN
   1.
```

(1) Mit der Prozedur in Arbeitsdatei 8 soll bei gesetztem Auftragsschalter 15 die Zeichenfolgevariable #S3 ausgegeben werden, anderenfalls aber die Zeichenfolgevariable #S2.

(2) Schalter 15 wird zurückgesetzt.

(3) Die Prozedur von Arbeitsdatei 8 wird ausgeführt.

(4) Schalter 15 wird gesetzt.

(5) Die Arbeitsdatei 8 wird ausgeführt.

9.61 @IF (Format 5) – Abfrage von Einstellungen des EDT

Mit diesem Format der @IF-Anweisung kann man in EDT-Prozeduren oder im L-Modus den aktuell eingestellten Betriebsmodus (siehe Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21) abfragen. Abhängig vom Ergebnis wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.

Operation	Operanden	L-Modus
@IF	**OP**ERATING-MODE = { **U**NICODE / **C**OMPATIB- }	:[text]

OPERATING-MODE=
: Es wird der Betriebsmodus des EDT überprüft.

  UNICODE
  : Die Bedingung ist erfüllt, wenn sich der EDT im Unicode-Modus befindet.

  COMPATIBLE
  : Die Bedingung ist erfüllt, wenn sich der EDT im Kompatibilitäts-Modus befindet.

text
: EDT-Anweisung oder Datenzeile. Falls die Bedingung erfüllt ist, wird die Zeichenfolge so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

  Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

  Wird `text` nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt.

Hinweis

Die Verwendung von @IF mit @RETURN als Anweisung außerhalb von Prozeduren führt möglicherweise zur Beendigung des EDT (siehe @RETURN-Anweisung).

Beispiel

```
1.     @IF OP = C : @GOTO 5
2.     @SET #S2 = 'PROZEDUR LAEUFT NUR IM KOMPATIBILITAETS-MODUS'
3.     @PRINT #S2 N
4.     @HALT
5.     @CONTINUE
6.     ...weitere Anweisungen
```

Die Prozedur beendet den EDT abnormal, wenn er sich nicht im Kompatibilitäts-Modus befindet.

9.62 @INDEX – Steuern der Zeilennummernanzeige

Die Anweisung @INDEX schaltet die Zeilennummernanzeige im F-Modus für die aktuelle Arbeitsdatei im jeweiligen Datenfenster ein bzw. aus (siehe auch den Abschnitt „Das Arbeitsfenster“ auf Seite 107).

Operation	Operanden	F-Modus
@INDEX	[**ON** | OFF]	

ON Schaltet die Zeilennummernanzeige ein (Standardwert).

OFF Schaltet die Zeilennummernanzeige aus.

Bei Beginn des EDT-Laufs ist die Zeilennummernanzeige für beide Datenfenster aller Arbeitsdateien eingeschaltet.

Bei eingeschalteter Zeilennummernanzeige werden im F-Modus, je nach der verwendeten Datensichtstation und deren Einstellung mittels @VDT-Anweisung 72 oder 124 Zeichen je Zeile, die 6-stellige Zeilennummernanzeige mit Dezimalpunkt und ein geschütztes Leerzeichen zur optischen Trennung vom Zeileninhalt angezeigt.

Bei ausgeschalteter Zeilennummernanzeige werden pro Zeile 80 bzw. 132 Zeichen dargestellt.

In beiden Formaten bildet die erste Spalte jeder Zeile die Kurzanweisungsspalte.

Die Einstellung der Zeilennummernanzeige wird getrennt für das obere und untere (mögliche) Datenfenster jeder Arbeitsdatei gespeichert. Wird nur ein Datenfenster am Bildschirm angezeigt, so wird die Einstellung für beide (möglichen) Datenfenster vorgenommen. Bei geteiltem Bildschirm (siehe @PAR SPLIT) dagegen wirkt sie nur auf das Arbeitsfenster, in dem sie eingegeben wurde, auch wenn in beiden Arbeitsfenstern die gleiche Arbeitsdatei angezeigt wird.

Durch @INDEX ON wird der `EDIT-LONG`-Modus (siehe Anweisung @EDIT) ausgeschaltet.

Anstelle von @INDEX kann auch die Anweisung @PAR INDEX benutzt werden. Zusätzlich lässt sich @PAR INDEX gezielt für eine Arbeitsdatei oder global für alle Arbeitsdateien benutzen und ist auch im L-Modus und damit in EDT-Prozeduren erlaubt. Für welches Datenfenster der jeweils angegebenen Arbeitsdatei @PAR INDEX wirksam wird, entnehme man der Beschreibung der @PAR-Anweisung.

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
   3.00 Und leider auch Theologie<·····································
   4.00 Durchaus studiert, mit heißem Bemühn.<·····························
@index off·····················································0001.00:00001(00)
```

Die Zeilennummernanzeige wird ausgeschaltet.

```
Habe nun, ach! Philosophie,<·····································
Juristerei und Medizin,<·····································
Und leider auch Theologie<·····································
Durchaus studiert, mit heißem Bemühn.<·····························
```

Das Arbeitsfenster wird ohne Zeilennummern angezeigt.

9.63 @INPUT (Format 1) – Starten einer @INPUT-Prozedur

Mit diesem Format der @INPUT-Anweisung wird eine @INPUT-Prozedur aus einer Datei gestartet. Die Anweisungen bzw. Datensätze aus der Datei werden sequentiell abgearbeitet.

Einzelheiten zum Aufbau und zur Bearbeitung von EDT-Prozeduren findet man im Abschnitt „EDT-Prozeduren“ auf Seite 66.

Operation	Operanden	F-Modus, L-Modus
@INPUT	**L**IBRARY=path1 ([**E**LEMENT=] elname [(vers)] [,eltype]) **E**LEMENT=elname [(vers)] [,eltype] **F**ILE = { path2 / *linkname } **P**OSIX-FILE=xpath [,**C**ODE=name]	[**P**RINT]

LIBRARY=... Die @INPUT-Prozedur wird durch explizite Angabe des Bibliotheks- und des Elementnamens bestimmt.

path1 Name der Bibliothek.

elname Name des Elements.

vers Version des gewünschten Elements (siehe Handbuch LMS [14]). Wird `vers` nicht angegeben oder wird `*STD` angegeben, wird die höchste vorhandene Version des Elementes gewählt.

eltype Typ des Elements. Zulässige Typangaben sind `S`, `M`, `P`, `J`, `D`, `X`, `*STD` und freie Typnamen mit entsprechendem Basistyp. Wird `eltype` nicht angegeben, wird der mit @PAR ELEMENT-TYPE voreingestellte Wert verwendet. Die zulässigen Elementtypen und deren Bedeutung sind im Kapitel „Dateibearbeitung“ auf Seite 137 beschrieben.

ELEMENT=... Die @INPUT-Prozedur wird durch den Namen des Elements ohne Angabe des Bibliotheknamens bestimmt. Es wird implizit die mit @PAR LIBRARY voreingestellte Bibliothek verwendet (sofern @PAR LIBRARY spezifiziert wurde – andernfalls wird die Fehlermeldung `EDT5181` ausgegeben).

Die Operanden `elname`, `vers` und `eltype` haben die gleiche Bedeutung wie bei expliziter Angabe der Bibliothek (siehe oben).

FILE= Die @INPUT-Prozedur wird durch den Namen einer BS2000-Datei bestimmt.

path2 Name der Datei, die als @INPUT-Prozedur eingelesen werden soll.

*linkname Dateikettungsname der BS2000-Datei, die als @INPUT-Prozedur eingelesen werden soll. Der Dateiname und die Dateiattribute sind in der `Task File Table` abgelegt. Der Dateikettungsname darf nicht mit den Spezial-Dateinamen `*BY-PROGRAM` vereinbart worden sein. Dies führt zum Fehler `EDT4923`. Ist der Dateikettungsname nicht definiert, wird die Anweisung mit dem Fehler `EDT5480` abgewiesen.

Ist der Dateikettungsname mit dem Spezial-Dateinamen `*DUMMY` vereinbart worden, wird dies wie eine existierende leere Datei behandelt.

POSIX-FILE= Die @INPUT-Datei wird durch den Pfadnamen einer POSIX-Datei bestimmt.

xpath Pfadname der POSIX-Datei, die als @INPUT-Prozedur eingelesen werden soll.

Der Operand `xpath` kann auch als Zeichenfolgevariable angegeben werden. Er muss als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

CODE= Es wird festgelegt, welcher Zeichensatz für die POSIX-Datei angenommen wird. Da POSIX-Dateien innerhalb des POSIX-Dateisystems kein Zeichensatz zugeordnet werden kann, ist hier eine Angabe des Anwenders erforderlich.

Wird `CODE` nicht angegeben, wird für die Datei der mit @PAR CODE eingestellte Zeichensatz angenommen.

name Zeichensatz der einzulesenden POSIX-Datei. Als `name` muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).

EBCDIC Das Schlüsselwort `EBCDIC` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `EDF041` unterstützt.

ISO Das Schlüsselwort `ISO` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `ISO88591` unterstützt.

PRINT Jede Zeile der Prozedur soll vor ihrer Verarbeitung protokolliert werden. Die Ausgabe erfolgt im Dialogbetrieb nach `SYSOUT` und im Stapelbetrieb nach `SYSLST`.
Durch die Angabe von `PRINT` wird auch erreicht, dass alle Fehlermeldungen ausgegeben werden und der EDT-Fehlerschalter gesetzt wird. Normalerweise werden die beiden Meldungen `EDT0901` und `EDT4932` in Prozeduren nicht ausgegeben und der EDT-Fehlerschalter nicht gesetzt (siehe Abschnitt „Meldungstexte“ auf Seite 664).

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar bzw. kann die Datei nicht erfolgreich eingelesen werden, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Die Anweisung @INPUT (Format 1) darf weder in @INPUT- noch in @DO-Prozeduren abgesetzt werden.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Die Abarbeitung einer @INPUT-Prozedur wird beendet, wenn die Anweisung @RETURN oder die letzte Anweisung der Prozedur abgearbeitet wurde.

Enthält die @INPUT-Prozedur Datensätze, werden diese wie bei der Dateneingabe im L-Modus in die aktuelle Arbeitsdatei eingefügt (leere Sätze werden ignoriert). Ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz *NONE, erhält die Arbeitsdatei den Zeichensatz der Datei, falls Datensätze eingefügt werden. Ist der Zeichensatz der Datei *NONE, erhält die Arbeitsdatei beim Einfügen von Sätzen den Zeichensatz EDF03IRV.

Die aus der @INPUT-Prozedur gelesenen Anweisungen oder Datensätze werden in dem Zeichensatz der angegebenen Datei interpretiert. Ist dieser *NONE, wird EDF03IRV benutzt.

Dieser Zeichensatz kann sich von dem Zeichensatz der aktuellen Arbeitsdatei unterscheiden. Da die Anweisungen stets auf die aktuelle Arbeitsdatei wirken bzw. die Datensätze in die aktuelle Arbeitsdatei eingefügt werden, kann es daher notwendig sein, Literale in Anweisungen bzw. Datensätze umzucodieren. Enthält die Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Abarbeitung der @INPUT-Prozedur mit der Fehlermeldung EDT5453 abgebrochen.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird die Abarbeitung der @INPUT-Prozedur mit der Meldung EDT5454 abgebrochen.

9.64 @INPUT (Format 2) – Starten einer @INPUT-Prozedur aus einer DVS-Datei

Mit diesem Format der @INPUT-Anweisung wird eine @INPUT-Prozedur aus einer SAM- oder ISAM-Datei gestartet. Format 2 wird nur noch aus Kompatibilitätsgründen unterstützt und sollte nicht mehr verwendet werden. Die Anweisungen bzw. Datensätze aus der Datei werden sequentiell abgearbeitet. Einzelheiten zum Aufbau und zur Bearbeitung von EDT-Prozeduren findet man im Abschnitt „EDT-Prozeduren“ auf Seite 66.

Operation	**Operanden**	**F-Modus, L-Modus**
@INPUT	file [(ver)] [lines[,...]] [:cols[,...]:] [{ **R**ECORD**S** | **K**EY }] [**P**RINT]	

file
: Name der SAM- oder ISAM-Datei, die eingelesen und abgearbeitet werden soll. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen.

  Die symbolische Bezeichnung `'/'` für eine Datei, für die der `LINK`-Name `EDTSAM` bzw. `EDTISAM` durch das `/SET-FILE-LINK`-Kommando vergeben wurde, ist hier nicht zulässig.

  Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar bzw. kann die Datei nicht erfolgreich eingelesen werden, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

ver
: Dieser Operand darf zwar aus Symmetriegründen angegeben werden, wird aber vollständig ignoriert.

lines
: Einer oder mehrere Zeilenbereiche, die in der ISAM- oder SAM-Datei abgearbeitet werden sollen. Fehlt `lines`, werden alle Zeilen der Datei abgearbeitet.

  Werden symbolische Zeilennummern angegeben, werden deren Werte aus der aktuellen Arbeitsdatei entnommen, haben also normalerweise nichts mit der Satzstruktur der angegebenen Datei zu tun.
  Eine Angabe von `lines` für SAM-Dateien wird ignoriert, wenn nicht gleichzeitig eines der Schlüsselwörter `KEY` oder `RECORDS` angegeben wurde.

cols
: Einer oder mehrere Spaltenbereiche, die die abzuarbeitenden Anweisungen enthalten. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Werden Spaltenwerte angegeben, die die Länge eines Satzes überschreiten, so werden dafür Leerzeichen eingelesen.
Bei Angabe von KEY für SAM-Dateien bzw. bei ISAM-Dateien beginnt die Spaltenzählung erst nach dem Schlüssel im Datensatz.
Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge eingelesen.

KEY
: Legt fest, dass die ersten 8 Zeichen einer jeden Zeile einer SAM-Datei als Zeilennummer zu interpretieren sind. Derartige Sätze lassen sich für SAM-Dateien mit @WRITE bei Angabe des Operanden KEY erstellen.
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 Datei als Textzeile betrachten.

RECORDS
: Legt fest, dass bei SAM-Dateien ein Zeilenbereich (siehe Operanden lines) über die logische Zeilennummer ausgewählt werden soll. Die logische Zeilennummer der 1. Zeile der Datei ist 0.0001, die logische Zeilennummer der 2. Zeile der Datei ist 0.0002 usw.

PRINT
: Jede Zeile der Prozedur soll vor ihrer Verarbeitung protokolliert werden. Die Ausgabe erfolgt im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST.

  Durch die Angabe von PRINT wird auch erreicht, dass alle Fehlermeldungen ausgegeben werden und der EDT-Fehlerschalter gesetzt wird. Normalerweise werden die beiden Meldungen EDT0901 und EDT4932 in Prozeduren nicht ausgegeben und der EDT-Fehlerschalter nicht gesetzt (siehe Abschnitt „Meldungstexte“ auf Seite 664).

Die Anweisung @INPUT (Format 2) darf weder in @INPUT- noch in @DO-Prozeduren abgesetzt werden.

Die Schlüsselwörter KEY und RECORDS werden für ISAM-Dateien ignoriert. Wenn weder RECORDS noch KEY angegeben wird, wird eine Angabe des Operanden lines für SAM-Dateien ignoriert, d.h. es werden alle Zeilen der Datei abgearbeitet.

Die Abarbeitung einer @INPUT-Prozedur wird abgebrochen, wenn die Anweisung @RETURN oder die letzte Anweisung der Prozedur abgearbeitet wurde.

Für ISAM-Dateien und SAM-Dateien (mit Operanden KEY) sollte der Satzschlüssel immer eine gültige Zeilennummer enthalten und bei SAM-Dateien erwartet der EDT eine aufsteigende Ordnung der Sätze.

Wird die Datei ohne Angabe eines Zeilenbereiches gelesen, werden die Satzschlüssel einfach ignoriert, also auch nicht überprüft. Werden dagegen Zeilenbereiche angegeben so ist das Verhalten für SAM-Dateien und ISAM-Dateien unterschiedlich.
Bei ISAM-Dateien werden die Schlüssel der tatsächlich gelesenen Sätze überprüft. Enthält ein Satz einen nichtnumerischen Schlüssel, wird die Abarbeitung der Prozedur mit dem Fehler EDT4984 abgebrochen.
Bei SAM-Dateien werden Sätze mit nichtnumerischen Schlüsseln oder mit Schlüsselwerten die kleiner als die aktuelle untere Bereichsgrenze sind ignoriert. Ist der Schlüsselwert größer als die aktuelle obere Bereichsgrenze, werden dieser und alle nachfolgenden Sätze ignoriert. Außerdem werden bei SAM Sätze die kürzer als 8 Zeichen sind ebenfalls ignoriert.

Enthält die @INPUT-Prozedur Datensätze, werden diese wie bei der Dateneingabe im L-Modus in die aktuelle Arbeitsdatei eingefügt (leere Sätze werden ignoriert). Ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz *NONE, erhält die Arbeitsdatei den Zeichensatz der Datei, falls Datensätze eingefügt werden. Ist der Zeichensatz der Datei *NONE, erhält die Arbeitsdatei beim Einfügen von Sätzen den Zeichensatz EDF03IRV.

Die aus der @INPUT-Prozedur gelesenen Anweisungen oder Datensätze werden in dem Zeichensatz der angegebenen Datei interpretiert. Ist dieser *NONE, wird EDF03IRV benutzt.

Dieser Zeichensatz kann sich von dem Zeichensatz der aktuellen Arbeitsdatei unterscheiden. Da die Anweisungen stets auf die aktuelle Arbeitsdatei wirken bzw. die Datensätze in die aktuelle Arbeitsdatei eingefügt werden, kann es daher notwendig sein, Literale in Anweisungen bzw. Datensätze umzucodieren.
Enthält die Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Abarbeitung der @INPUT-Prozedur mit der Fehlermeldung EDT5453 abgebrochen. Dies gilt auch, falls sich das ungültige Zeichen außerhalb des einzulesenden Spaltenbereichs befindet. Ungültige Zeichen außerhalb des einzulesenden Zeilenbereichs werden hingegen ignoriert.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird die Abarbeitung der @INPUT-Prozedur mit der Meldung EDT5454 abgebrochen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

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 ---------------------------------------------- (1)
6.     @SAVE 'ISAM-INP' -------------------------------------------------- (2)
6.     @INPUT 'SAM-INP' KEY ---------------------------------------------- (3)
1.0000 ICH BIN ZEILE 1
2.0000 ICH BIN DIE ZWEITE ZEILE
3.     @INPUT 'ISAM-INP' 1-3,5,4 ----------------------------------------- (4)
2.0000 ICH BIN DIE ZWEITE ZEILE
1.0000 ICH BIN ZEILE 1
3.
```

(1) 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.

(2) Der Inhalt der Arbeitsdatei wird noch einmal geschrieben, diesmal aber als ISAM-Datei.

(3) Die komplette Datei `SAM-INP` wird eingelesen und ausgeführt. Da diese Datei mit @WRITE unter Benutzung von `KEY` erstellt wurde, ist `KEY` anzugeben. Anderenfalls erfolgt keine Umrechnung der gespeicherten Schlüssel in die Zeilennummern.

(4) Die Zeilen `1-3,5,4` der Datei `ISAM-INP` sollen in dieser Reihenfolge eingelesen und ausgeführt werden.

9.65 @INPUT (Format 3) – Festlegen des Eingabemodus des EDT

Mit diesem Format der @INPUT-Anweisung bestimmt der Benutzer, wie der EDT Texteingaben im L-Modus interpretieren soll.

Operation	Operanden	L-Modus
@INPUT	[{ [**C**HAR] / **H**EX | **X** [**I**SO] / **B**INARY }]	

CHAR
: Der EDT wird veranlasst, die Eingabe von Datensätzen im L-Modus als Folge von Zeichen im Zeichensatz der jeweiligen Eingabequelle (Datensichtstation, SYSDTA, Datei, Bibliothekselement, Arbeitsdatei) zu interpretieren (siehe Abschnitt „L-Modus“ auf Seite 131, auch zur Behandlung von Zeichensätzen bei der L-Modus-Eingabe).

HEX, X
: Der EDT wird veranlasst, die Eingabe von Datensätzen im L-Modus als Folge von Hexadezimalzeichen zu interpretieren (siehe Abschnitt „L-Modus“ auf Seite 131, auch zur Behandlung von Zeichensätzen bei der Hexadezimaleingabe).

ISO
: Dieser Operand wird im Unicode-Modus des EDT V17.0 nicht mehr unterstützt. Aus Kompatibilitätsgründen wird seine Angabe ohne Fehlermeldung ignoriert. ISO-Zeichensätze unterliegen im Unicode-Modus des EDT V17.0 keiner Sonderbehandlung. Wenn für die aktuelle Arbeitsdatei ein ISO-Zeichensatz eingestellt ist, werden hexadezimale Eingaben für diese Arbeitsdatei automatisch im richtigen Code interpretiert, eine implizite Umwandlung nach EBCDIC erfolgt nicht.

BINARY
: Der EDT wird veranlasst, die Eingabe von Datensätzen im L-Modus als Folge von Binärzeichen zu interpretieren (siehe Abschnitt „L-Modus“ auf Seite 131).

Nach dem Aufruf des EDT ist @INPUT CHAR voreingestellt.

Die maximal zulässige Abkürzung der Anweisung kann nur benutzt werden, wenn Operanden angegeben sind. Bei Aufruf ohne Operanden kann maximal auf @INP abgekürzt werden, ansonsten wird die Anweisung @INDEX erkannt.

Hinweis

Anweisungen dürfen, auch wenn @INPUT HEX oder @INPUT BINARY gegeben wurde, *nicht* in hexadezimaler oder binärer Codierung eingegeben werden.

9.66 @LIMITS – Zeilennummern und Anzahl der Zeilen ausgeben

Mit der Anweisung @LIMITS gibt der EDT für die aktuelle Arbeitsdatei die niedrigste und höchste vergebene Zeilennummer sowie die Anzahl der Zeilen aus.

Die Ausgabe erfolgt in einer Zeile im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST.

Operation	Operanden	F-Modus, L-Modus
@LIMITS		

Für real geöffnete Dateien wird immer die Zeilenanzahl 0 ausgegeben.

Beispiel

```
   4.     @PRINT
   1.0000 A
   2.0000 B
   3.0000 C
   4.     @LIMITS
   1.0000 TO    3.0000          3 LINES ---------------------------------- (1)

   4.     @COPY 1-3 TO 99.01 ------------------------------------------- (2)

 100.03   @LIMITS
   1.0000 TO   99.0300          6 LINES ---------------------------------- (3)

 100.03
```

(1) Die niedrigste und höchste vergebene Zeilennummer und die Zeilenanzahl werden ausgegeben.

(2) Die Zeilen 1-3 werden in die Zeilen 99.01, 99.02 und 99.03 kopiert.

(3) Nun sind die niedrigste bzw. höchste vergebene Zeilennummer 1.0000 bzw. 99.0300. Die Zeilenanzahl beträgt 6.

9.67 @LIST – Ausdrucken von Arbeitsdateibereichen oder Zeichenfolgevariablen

Mit der Anweisung @LIST können Bereiche einer Arbeitsdatei oder von Zeichenfolgevariablen auf SYSLST oder über den Drucker ausgegeben werden. Wenn nicht anders spezifiziert, wird jeder ausgegebenen Zeile die Zeilennummer, jeder ausgegebenen Zeichenfolgevariablen die Nummer der Zeichenfolgevariablen vorangestellt.

Operation	Operanden	F-Modus, L-Modus
@LIST	[{lines / svars} [:cols[:]] [**X**] [**N**] [{**C** [int] / **P** int}] [**I**] [**S**]] [,...]	

lines — Der Zeilenbereich, der ausgegeben werden soll.

svars — Der Bereich von Zeichenfolgevariablen, deren Inhalt ausgegeben werden soll.

cols — Spaltenbereich in der aktuellen Arbeitsdatei bzw. in den angegebenen Zeichenfolgevariablen, der ausgegeben werden soll.

Wird nur eine Spaltennummer angegeben, wird ab dieser Spalte der Rest der Zeile bzw. Zeichenfolgevariablen ausgegeben. Ist die erste Spaltenangabe größer als die Länge der Zeile bzw. Zeichenfolgevariablen, wird diese Zeile bzw. Zeichenfolgevariable nicht behandelt.

Wird kein Spaltenbereich angegeben, wird die Zeile bzw. Zeichenfolgevariable in voller Länge ausgegeben.

X — Die Zeilen werden hexadezimal ausgedruckt. Das Ausgabeformat entspricht dem bei @PRINT.. X.

N — Die Zeilennummern bzw. die Nummern der Zeichenfolgevariablen werden beim Ausdrucken unterdrückt.

C int — Der EDT erwartet jeweils als erstes Zeichen des ausgegebenen Spaltenbereichs ein EBCDIC-Vorschubsteuerzeichen. Dieses bewirkt beim Drucken einen Vorschub, wird aber selbst nicht ausgedruckt. Das Vorschubsteuerzeichen muss im Zeichensatz der aktuellen Arbeitsdatei bzw. der jeweiligen Zeichenfolgevariablen vorliegen (siehe unten). Wenn der EDT dieses Zeichen interpretiert, legt er die Bedeutung des äquivalenten EDF041-Zeichens zugrunde.
Alle Zeichen in der ersten Spalte, die der EDT nicht interpretieren kann, führen zu einem einfachen Zeilenvorschub.

Für int sind Werte zwischen 0 und 256 erlaubt. Wird int ungleich 0 angegeben, generiert der EDT zusätzlich zu den im Satz selbst enthaltenen Vorschubsteuerzeichen nach jeweils int ausgegebenen Zeilen (unter Berücksichtigung der im Satz selbst enthaltenen Seiten- und Zeilenvorschübe) einen Seitenvorschub. Wird int gleich 0 angegeben, wird kein zusätzlicher Seitenvorschub generiert.
Der hier eingestellte Wert für die Seitengröße bleibt eingestellt und beeinflusst damit alle Ausgaben nach SYSLST.

Wird int nicht angegeben, verwendet der EDT den zuletzt mit der Anweisung @LIST oder @PAGE eingestellten Wert für die Seitengröße, unabhängig davon, ob int in Zusammenhang mit C oder P angegeben wurde. Wurde vorher noch keine @LIST-Anweisung gegeben, die den Wert von int verändert hat, wird der Standardwert 65 (siehe @PAGE-Anweisung) angenommen.

P int

Der EDT stellt jedem ausgegebenen Satz selbst ein EBCDIC Vorschubsteuerzeichen voran. Der EDT verwendet dabei nur die EBCDIC Vorschubsteuerzeichen Zeilenvorschub nach dem Drucken jeder Zeile sowie das Vorschubsteuerzeichen Seitenvorschub in der jeweils dem Zeichensatz der Ausgabedatei (SYSLST oder temporäre Datei) entsprechenden Codierung (siehe unten).

Für int sind Werte zwischen 0 und 256 erlaubt. Wenn int nicht 0 ist, wird nach jeweils genau int Zeilen ein Seitenvorschub eingefügt. Wenn int gleich 0 ist, wird kein Seitenvorschub eingefügt. Der hier eingestellte Wert für die Seitengröße bleibt eingestellt und beeinflusst damit alle Ausgaben nach SYSLST.

I

Mit dem Ausdrucken wird sofort begonnen. Das bedeutet, dass der EDT die auszugebenden Sätze in eine temporäre Datei (unter Benutzung der Systemdatei SYSLST97) schreibt und diese nach dem Schließen sofort mit einem /PRINT-DOCUMENT-Kommando auf den eingestellten Standard-Drucker ausgibt.

Die temporäre Datei wird bei Ausgabe von Bereichen aus einer Arbeitsdatei in dem Zeichensatz der Arbeitsdatei angelegt, wenn dieser ein EBCDIC-Zeichensatz ist. Falls nicht, wird der Referenz-Zeichensatz verwendet, wenn dieser ein EBCDIC-Zeichensatz ist. In allen anderen Fällen wird die temporäre Datei im Zeichensatz UTFE angelegt. Wenn Bereiche von Zeichenfolgevariablen ausgegeben werden sollen, erfolgt die Bestimmung des Zeichensatzes analog für jede einzelne Zeichenfolgevariable. Sind die dabei ermittelten Zeichensätze alle gleich, wird dieser Zeichensatz verwendet, anderenfalls der Zeichensatz UTFE.

Wenn der Anwender nicht die Erlaubnis hat, temporäre Dateien anzulegen oder wenn vom Systemverwalter die Benutzung temporärer Dateien generell verboten ist oder wenn es aus anderen Gründen nicht gelingt die Datei anzulegen bzw. den Inhalt zu schreiben, wird die @LIST-Anweisung mit Operand `I` mit einer entsprechenden Fehlermeldung abgewiesen.

Der Operand `I` ist nur im Dialogbetrieb erlaubt. Er ist nur dann sinnvoll, wenn der eingestellte Standarddrucker in der Lage ist, den Zeichensatz, in dem die Ausgabe erfolgt, korrekt darzustellen.

Wird `I` nicht angegeben, so wird nach `SYSLST` ausgegeben. Falls `SYSLST` keiner Datei zugeordnet ist, wird dann mit dem Ausdrucken erst nach `/LOGOFF` begonnen. Falls `SYSLST` einer Datei zugeordnet ist, muss der Anwender den Ausdruck selbst veranlassen.

S — Unterdrückt den zusätzlichen Zeilenvorschub, der üblicherweise vor der ersten ausgegebenen Zeile erfolgt.

Wird weder `lines` noch `svars` angegeben, wird die gesamte aktuelle Arbeitsdatei ausgegeben.

Ist weder `P` noch `C` angegeben, erzeugt der EDT Vorschubsteuerzeichen wie bei `P`, wobei der zuletzt mit der Anweisung @LIST oder @PAGE eingestellte Wert für die Seitengröße verwendet wird.

Erzeugt der EDT die Vorschubsteuerzeichen, so berücksichtigt er die eingestellte oder angegebene Seitengröße und erzeugt normalerweise vor jeder ersten Zeile eines Bereiches einen zusätzlichen Zeilenvorschub. Dies unterbleibt nur, bei Angabe des Operanden `S` oder wenn die Ausgabe dieser Zeile an einem Seitenanfang geschieht. Die Ausgaben werden nach Ausgabe von jeweils 132 Zeichen (bzw. 160 Zeichen, wenn Auftragsschalter 6 gesetzt ist) umgebrochen.

Bei der Ausgabe von Bereichen einer Arbeitsdatei werden diese vom Zeichensatz der Arbeitsdatei in den Zeichensatz von `SYSLST` bzw. den Zeichensatz der temporären Datei (siehe Beschreibung des Operanden `I`) umcodiert. Bei der Ausgabe von Bereichen von Zeichenfolgevariablen wird jede Zeichenfolgevariable von dem ihr zugeordneten Zeichensatz in den Zeichensatz von `SYSLST` bzw. den Zeichensatz der temporären Datei (siehe Beschreibung des Operanden `I`) umcodiert. Werden dabei Zeichen gefunden, denen im Ziel-Zeichensatz kein gültiges Zeichen entspricht, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls werden dafür Leerzeichen eingesetzt.

Die EBCDIC- und ASA-Vorschubsteuerzeichen im BS2000 stellen in jedem verwendeten 8-Bit- oder Unicode-Zeichensatz gültige Zeichen dar, sind also im EDT editierbar, wenn auch nicht immer darstellbar. Das gilt insbesondere für die vom EDT interpretierten bzw.

eingefügten Vorschubsteuerzeichen (siehe Operanden `C` bzw. `P`). Folglich werden einige Vorschubsteuerzeichen als Unicode-Zeichen mit mehr als einem Byte codiert (siehe Tabelle unten).

Das BS2000-Subsystem SPOOL wandelt für Druckdateien (z.B. SYSLST), die in einem Unicode-Zeichensatz vorliegen, das Vorschubsteuerzeichen aus dem Unicode-Zeichensatz der Druckdatei nach `EDF041` um und interpretiert das erhaltene Zeichen wie es im Handbuch Kommandos Band 3 [10] bei der Beschreibung des Kommandos `/PRINT-DOCUMENT` erläutert ist.
Die Umcodierung findet nur statt, wenn die Datei mit `LINE-SPACING=*BY-EBCDIC-CONTROL` oder `LINE-SPACING=*BY-ASA-CONTROL` gedruckt wird. Für Dateien, die mit `LINE-SPACING=*BY-IBM-CONTROL` gedruckt werden sollen, findet keine Umcodierung statt. Bei diesen Dateien sind die Vorschubsteuerzeichen meist keine gültigen Unicode-Zeichen, so dass solche Dateien mit dem EDT nicht bearbeitet werden können. Liegt die Druckdatei in einem 7-Bit- oder 8-Bit-Zeichensatz vor, interpretiert SPOOL das Vorschubsteuerzeichen ohne vorherige Umcodierung wie bisher.

Für die vom EDT interpretierten bzw. generierten EBCDIC-Vorschubsteuerzeichen sei hier die Codierung in verschiedenen Zeichensätzen angegeben:

UTF16	**UTF8**	**UTFE**	**EDF041**	**Zeichen**	
0020	20	40	40	␣	Kein Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00a0	c2a0	6741	41		Eine Zeile Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e2	c3a2	68b0	42	â	Zwei Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e4	c3a4	689f	43	ä	Drei Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e0	c3a0	6841	44	à	Vier Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e1	c3a1	68aa	45	á	Fünf Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e3	c3a3	68b1	46	ã	Sechs Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e5	c3a5	68b2	47	å	Sieben Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00e7	c3a7	68b5	48	ç	Acht Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
00f1	c3b1	688f	49	ñ	Neun Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken

UTF16	UTF8	UTFE	EDF041	Zeichen	
0060	60	4a	4a	`	Zehn Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
002e	2e	4b	4b	.	Elf Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
003c	3c	4c	4c	<	Zwölf Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
0028	28	4d	4d	(	Dreizehn Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
002b	2b	4e	4e	+	Vierzehn Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
007c	7c	4f	4f	|	Fünfzehn Zeilen Vorschub vor dem Drucken, neue Zeile nach dem Drucken
0041	41	c1	c1	A	Seitenvorschub vor dem Drucken

Beim Interpretieren von Vorschubsteuerzeichen, akzeptiert der EDT zusätzlich die EBCDIC-Steuerzeichen `x'00'..x'0F'` (bzw. ihre Äquivalente in anderen Zeichensätzen) und interpretiert sie wie `x'40'..x'4F'`. Bei Ausgabe im Zeichensatz `UTF8` oder `UTFE` generiert der EDT allerdings niemals Steuerzeichen, die mit mehr als einem Byte codiert sind, stattdessen generiert er die entsprechende Anzahl einfacher Zeilenvorschübe.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Die Systemdateien `SYSLST` und `SYSOUT` sollten nur dann Dateien oder Bibliothekselementen mit Unicode-Zeichensatz zugewiesen werden, wenn man sicher sein kann, dass nur der EDT Ausgaben in diese Dateien macht. Da andere Systemkomponenten den `SYSLST` bzw. `SYSOUT` zugeordneten Zeichensatz in der Regel nicht berücksichtigen, könnten andernfalls Dateien entstehen, die ungültige Zeichen enthalten.

Beispiel

```
6.     @PRINT
1.0000 MIT DER @LIST-ANWEISUNG
2.0000 KANN DER INHALT
3.0000 EINER ARBEITSDATEI
4.0000 IN VIELEN FORMATEN
5.0000 ZU PAPIER GEBRACHT WERDEN.
6.     @LIST ------------------------------------------------------------ (1)
6.     @LIST 4-5 N ------------------------------------------------------ (2)
6.     @LIST 4 :12-13 X ------------------------------------------------- (3)
6.     @LIST & I -------------------------------------------------------- (4)
```

(1) Der komplette Inhalt der Arbeitsdatei soll nach SYSLST ausgegeben werden. Der (eventuell vom System veranlasste) Ausdruck wird erst nach LOGOFF angestartet.

Druckausgabe

```
1.0000 MIT DER @LIST-ANWEISUNG
2.0000 KANN DER INHALT
3.0000 EINER ARBEITSDATEI
4.0000 IN VIELEN FORMATEN
5.0000 ZU PAPIER GEBRACHT WERDEN.
```

(2) Die Zeilen 4 bis 5 sollen ohne Zeilennummern nach SYSLST ausgegeben werden.

Druckausgabe

```
IN VIELEN FORMATEN
ZU PAPIER GEBRACHT WERDEN.
```

(3) Die beiden Spalten 12 und 13 der Zeile 4 sollen in hexadezimaler Form nach SYSLST ausgegeben werden.

Druckausgabe

```
4.0000 D6D9
```

(4) Alle Zeilen sollen sofort ausgedruckt werden.

Druckausgabe

ist wie bei (1), jedoch erscheint zusätzlich die Systemmeldung:

```
%  SCP0810 SPOOLOUT OF FILE 'XXX' ACCEPTED, TSN: 'XXX', PNAME: 'XXX'.
```

als Bestätigung, dass der Druckauftrag erteilt wurde.

9.68 @LOAD – Programm laden

Die Anweisung @LOAD bewirkt, dass die EDT-Sitzung beendet und das angegebene Programm geladen wird.

Operation	**Operanden**	**F-Modus, L-Modus**
@LOAD	string	

string — Zeichenfolge, die den Namen des Programms angibt, das geladen werden soll. Es wird der Name einer BS2000-Datei erwartet, die das zu ladende Programm enthält. Die Angabe eines Bibliothekselements ist nicht möglich.

Die Anweisung @LOAD gehört zu den sicherheitsrelevanten Anweisungen des EDT (siehe hierzu auch Abschnitt „Zugriffsschutz“ auf Seite 103). In nichtunterbrechbaren Systemprozeduren im Dialog und bei Eingabe von einer Datei (Lesen mit `RDATA` von `SYSDTA` ungleich `SYSCMD`, Abarbeiten einer Start-Prozedur) wird die Anweisung abgewiesen.

Die Anweisung @LOAD führt auf jeden Fall zur Beendigung des EDT, unabhängig davon, ob die angegebene Programmdatei existiert oder ein gültiges Programm enthält.

Bezüglich der Behandlung von ungesicherten Dateien und der damit verbundenen Sicherheitsabfragen wirkt die Anweisung @LOAD wie @HALT (siehe Abschnitt „Beenden des EDT-Laufs“ auf Seite 96). Da der EDT in jedem Fall beendet wird, erfolgen eventuelle Sicherungsabfragen anders als bei @HALT auch dann, wenn die Anweisung im Bildschirmdialog (mit @DIALOG gestartet) eingegeben wurde.

Für den Fall, dass der EDT als Unterprogramm geladen und der Bildschirmdialog des EDT mit @DIALOG eingeschaltet wurde, führt die Anweisung @LOAD nicht zur Fortsetzung des Benutzerprogramms. Stattdessen wird auch das Benutzerprogramm entladen.

Daher kann bei Aufruf des EDT als Unterprogramm die Anweisung @LOAD für den Benutzer verboten werden. Sie wird dann mit der Meldung `EDT4976` abgewiesen.

Hinweis

Wurde @DIALOG in einer Systemprozedur gegeben, werden nach @LOAD die restlichen Kommandos der Prozedur ausgeführt, während das angegebene Programm anstelle des EDT geladen ist, was zu unerwünschten Effekte führen kann.

Beispiel

Es wird vorausgesetzt, dass die in der Arbeitsdatei befindlichen Sätze noch nicht gesichert wurden.

```
   1.00 Der EDT soll beendet<·····················································
   2.00 und der LMS geladen werden<···············································
   3.00 Dazu wird die Anweisung @LOAD eingegeben<·····························
   4.00 ··························································································
@LOAD '$lms'·····················································0001.00:00001(00)
```

Der EDT soll beendet und der LMS geladen werden.

```
%  EDT0900 EDITED FILE(S) NOT SAVED!
 LOCAL FILE ( O) :
%  EDT0904 TERMINATE EDT? REPLY (Y=YES; N=NO)?y
%  BLS0500 PROGRAM 'LMS', VERSION 'V3.0A1 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.

9.69 @LOG – Protokollsteuerung

Die Anweisung @LOG steuert die Protokollierung der Eingaben im Stapelbetrieb und im Dialog.

Operation	Operanden	F-Modus, L-Modus
@LOG	[{**A**LL | **C**OMMANDS | **N**ONE}] [{**SYSLST** | **SYSLST** n}]	

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 werden die Eingaben in den Anweisungszeilen (bei Anweisungskettung in einzelne Anweisungen getrennt) protokolliert.

COMMANDS Nur Anweisungen sollen protokolliert werden.

NONE Nichts soll protokolliert werden.

SYSLST Die Protokollierung erfolgt nach SYSLST. Dies ist die Voreinstellung nach dem Starten des EDT.

SYSLST n Die Protokollierung erfolgt in die Datei, die SYSLSTnn zugewiesen ist (für n sind Werte zwischen 1 und 99 erlaubt).

Die Voreinstellung nach dem Starten des EDT im Stapelbetrieb ist @LOG NONE, falls Auftragsschalter 4 gesetzt ist bzw. @LOG COMMANDS, falls Auftragsschalter 4 nicht gesetzt ist. Wird der EDT im Dialogbetrieb aufgerufen, ist @LOG NONE voreingestellt.

Die Festlegung des Ausgabemediums (SYSLST, SYSLSTnn) bleibt für nachfolgende @LOG-Anweisungen gültig, wenn dort nicht explizit ein anderer Wert angegeben wird.

Hinweis

Anweisungen und Dateneingaben, die aus EDT-Prozeduren gelesen und abgearbeitet werden, werden durch @LOG weder im Stapelbetrieb noch im Dialogbetrieb protokolliert. Deren Protokollierung muss explizit mit @DO ...PRINT bzw. @INPUT ...PRINT verlangt werden.
Die Anweisung @LOG wird auch im Testmodus nicht nur syntaktisch geprüft, sondern ausgeführt (siehe @SYNTAX-Anweisung).

Die in der Vorversion mögliche Protokollierung in eine Listenvariable wird nicht mehr unterstützt. Man kann aber mit dem Kommando /ASSIGN-SYSLST der Systemdatei SYSLST eine Listenvariable zuweisen.

9.70 @LOWER – Groß-/Kleinschreibung bei der Eingabe

Mit der Anweisung @LOWER kann man festlegen, ob der EDT bei der Eingabe von Daten und Anweisungen von der Datensichtstation Kleinbuchstaben in Großbuchstaben umsetzt oder nicht.

Operation	Operanden	F-Modus, L-Modus
@LOWER	[**ON** | **OF**F]	

ON Der EDT unterscheidet zwischen Groß- und Kleinbuchstaben. Zeichenfolgen werden verarbeitet, wie sie eingegeben werden.

OFF Der EDT codiert eingegebene Kleinbuchstaben in Großbuchstaben um.

Im F-Modus werden in der Arbeitsdatei enthaltene Kleinbuchstaben bei der Ausgabe im Arbeitsfenster als Schmierzeichen dargestellt. Bei Ausgabe nach SYSOUT oder SYSLST (z.B. im L-Modus oder durch die Anweisung @ON, Format 1) werden sie abdruckbar dargestellt.

Nach dem Starten des EDT ist der Wert ON für alle Arbeitsdateien voreingestellt.

Die Anweisung @LOWER wirkt global für alle Arbeitsdateien. Mit der Anweisung @PAR LOWER kann die Art der Behandlung von Kleinbuchstaben für jede Arbeitsdatei separat eingestellt werden.

Der EDT verwendet zur Umwandlung von Klein- in Großbuchstaben die Systemkomponente XHCS. Welche Zeichen umgewandelt werden, hängt daher von der Definition der jeweiligen Zeichensatz-Attribute in XHCS ab.

Wird die Anweisung @LOWER im L-Modus innerhalb eines Eingabe-Blocks (siehe @BLOCK) bzw. im F-Modus innerhalb einer Anweisungsfolge (durch ; getrennte Anweisungen) angegeben, wird der Umcodierungsmodus ab der auf @LOWER folgenden Anweisung oder Datenzeile wirksam.

Wenn @LOWER OFF eingestellt ist, werden in Eingaben von der Datensichtstation grundsätzlich alle Zeichen von Kleinbuchstaben in Großbuchstaben umgesetzt, unabhängig davon ob die Eingabe im F-Modus oder im L-Modus erfolgte oder ob es sich um Eingabe von Datenzeilen oder um Eingabe von Anweisungen handelte. Diese Umsetzung findet im F-Modus aber erst nach der Ablage einer Anweisung im Anweisungspuffer statt, d.h. dort werden Anweisungen so abgelegt, wie sie eingegeben wurden (siehe auch @SHIH-Anweisung).

Werden Eingaben aus Dateien oder Arbeitsdateien gelesen, z.B. bei der Abarbeitung von EDT-Prozeduren oder beim Lesen von SYSDTA, das einer Datei zugewiesen wurde, findet auch bei @LOWER OFF für Texteingaben und Literale in Anweisungen keine Umsetzung in Großbuchstaben statt. Dies entspricht dem Verhalten beim Einlesen von Dateien, bei dem ebenfalls keine Umsetzung erfolgt.

Die Einstellung @LOWER ON wirkt sich bei der Eingabe von Anweisungen nur auf die Literale aus. Die Anweisungen und Schlüsselwörter werden während der Analyse der Anweisung grundsätzlich in Großbuchstaben umgesetzt.

9.71 @MODE – Betriebsmodus umschalten

Mit der Anweisung @MODE kann zwischen den Betriebsmodi (Kompatibilitäts-Modus und Unicode-Modus, siehe Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21) umgeschaltet werden.

Operation	Operanden	F-Modus, L-Modus
@MODE	**OP**ERATING-MODE = { **U**NICODE / **C**OMPATIB- }	

OPERATING-MODE=
Es wird der Betriebsmodus des EDT umgeschaltet.

UNICODE Es wird vom Kompatibilitäts-Modus in den Unicode-Modus umgeschaltet. Wenn sich der EDT bereits im Unicode-Modus befindet, wird die Anweisung ignoriert.

COMPATIBLE
Es wird vom Unicode-Modus in den Kompatibilitäts-Modus umgeschaltet. Wenn sich der EDT bereits im Kompatibilitäts-Modus befindet, wird die Anweisung ignoriert.

Der Betriebsmodus kann nur dann umgeschaltet werden, wenn alle Arbeitsdateien des EDT leer und keine Dateien geöffnet sind. Andernfalls wird die Anweisung mit der Meldung `EDT4983` abgewiesen.

Der Wechsel zwischen den Betriebsmodi entspricht einem Beenden des EDT im einen Modus mit anschließendem Neustart des EDT im anderen Modus. Dabei gehen alle Einstellungen verloren und alle Variablen werden neu initialisiert. Einzelheiten dazu findet man im Abschnitt „Aktivieren von Kompatibilitäts- und Unicode-Modus“ auf Seite 637.

9.72 @MOVE – Übertragen von Zeilen oderZeichenfolgevariablen

Mit der Anweisung @MOVE werden Datensätze der aktuellen oder einer anderen Arbeitsdatei in die aktuelle Arbeitsdatei übertragen und anschließend an ihren ursprünglichen Positionen gelöscht oder Inhalte von Zeichenfolgevariablen in die aktuelle Arbeitsdatei übertragen und anschließend werden die Zeichenfolgevariablen reinitialisiert.

Der Zeilenbereich der Quell-Arbeitsdatei, der die zu übertragenden Datensätze enthält, bzw. der Bereich von Zeichenfolgevariablen wird der Einfachheit halber im Folgenden als Sendebereich bezeichnet. Der Zeilenbereich der aktuellen Arbeitsdatei, in den die Datensätze der Quell-Arbeitsdatei übertragen werden, wird als Empfangsbereich bezeichnet.

Operation	**Operanden**	**F-Modus, L-Modus**
@MOVE	{ { lines [(procnr)] / svars } [**T**O {line1 [(inc)] [:] [line2]} [,...]] } [,...]	

lines — Zusammenhängender Zeilenbereich, der in die aktuelle Arbeitsdatei übertragen werden soll. Symbolische Zeilennummern in lines beziehen sich auf die Zeilennummern der aktuellen Arbeitsdatei, auch wenn die Zeilen aus einer anderen Arbeitsdatei übertragen werden.

procnr — Nummer der Quell-Arbeitsdatei, aus der die Datensätze übertragen werden (`0..22`). Ist `procnr` nicht angegeben, werden die Datensätze aus der aktuellen Arbeitsdatei übertragen. Eine aktive Arbeitsdatei darf nicht angegeben werden. Wird kein `TO` Operand angegeben, darf `procnr` nicht die aktuelle Arbeitsdatei sein.

svars — Bereich von Zeichenfolgevariablen, deren Inhalte in die aktuelle Arbeitsdatei übertragen werden soll. Nach der Übertragung werden die Zeichenfolgevariablen gelöscht, d.h. mit einem Leerzeichen im Zeichensatz `EDF041` reinitialisiert.

TO... — Mit den auf `TO` folgenden Operanden werden der oder die Empfangsbereiche definiert. Ist kein Empfangsbereich angegeben, dann werden die Zeilennummern der Quell-Arbeitsdatei in die aktuelle Arbeitsdatei übernommen.

Ist die Quell-Arbeitsdatei die aktuelle Arbeitsdatei oder werden Zeichenfolgevariable übertragen, dann muss `TO...` angegeben werden. Ist in diesen Fällen kein Empfangsbereich angegeben, dann wird die @MOVE-Anweisung mit der Fehlermeldung `EDT3218` abgewiesen.

line1 — Nummer der ersten Zeile des Empfangsbereiches.

inc — Schrittweite, aus der die auf `line1` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line1` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

: — Die Operanden `line1` und `line2` sollten durch `:` voneinander getrennt werden, falls `inc` nicht angegeben wird.

line2 — Legt die größtmögliche Zeilennummer des Empfangsbereiches fest, bis zu der eine Übertragung von Datensätzen erlaubt wird.

Es findet somit keine Übertragung in Zeilen der aktuellen Arbeitsdatei statt, deren Zeilennummer größer als `line2` ist. Dies gilt auch dann, wenn dadurch nicht alle Datensätze des Sendebereiches in den Empfangsbereich übertragen werden können.

Ist `line2` nicht angegeben, so wird durch die @MOVE-Anweisung kein Maximalwert für die Zeilennummern des Empfangsbereiches festgelegt.

Mit der @MOVE-Anweisung können mehrere Sendebereiche mit jeweils mehreren Empfangsbereichen durch Komma getrennt angegeben werden. Die Anzahl der Sende- und Empfangsbereiche ist nur durch die maximal erlaubte Länge einer EDT-Anweisung begrenzt. Die Angabe mehrerer Empfangsbereiche ist normalerweise nicht sinnvoll, weil die Zeilen bereits nach der ersten Übertragung im Sendebereich gelöscht werden und für weitere Übertragungen nicht mehr zur Verfügung stehen.

Wenn sich der Sende- und der Empfangsbereich überlappen, wird der Sendebereich zeilenweise übertragen und gelöscht.

Bereits in der aktuellen Arbeitsdatei existierende Zeilen mit gleichen Zeilennummern werden bei der Übertragung überschrieben.
Wenn eine Zeile angelegt wird, deren Nummer größer als die bisherige höchste Zeilennummer ist, wird die aktuelle Zeilennummer verändert.

Ist die aktuelle Arbeitsdatei leer und hat sie den Zeichensatz `*NONE`, dann erhält sie beim Übertragen den Zeichensatz der Quell-Arbeitsdatei bzw. der ersten angegebenen Zeichenfolgevariablen.

Hat die aktuelle Arbeitsdatei einen Zeichensatz, dann werden die zu übertragenden Zeilen bzw. die Inhalte der Zeichenfolgevariablen in den Zeichensatz der aktuellen Arbeitsdatei konvertiert. Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @MOVE-Anweisung abgebrochen und die Fehlermeldung `EDT5453` ausgegeben.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Da es die obige Syntax erlaubt, den `TO`-Operanden wegzulassen, ist es nicht immer möglich, eindeutig zwischen einem Empfangsbereich und einem nächsten Sendebereich zu unterscheiden. In diesen Fällen entscheidet sich der EDT für die Interpretation als Empfangsbereich. So wird z.B. in der Eingabe

```
@MOVE 2-3(1) TO 7,1(1)
```

die Angabe `1(1)` als zweiter Empfangsbereich interpretiert (die `1` in Klammern wird als Schrittweite interpretiert), während die Angabe `1(0)` an dieser Stelle als nächster Sendebereich interpretiert worden wäre (die `0` kann keine Schrittweite sein und wird als Arbeitsdateinummer interpretiert). Will der Anwender in diesem Beispiel die Interpretation als Sendebereich erzwingen, könnte er z.B.

```
@MOVE 2-3(1) TO 7,1-1(1)
```

eingeben, dann gibt es keine Mehrdeutigkeiten.

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<................................................
  90.00 DIE ZEILE WIRD NIE UEBERSCHRIEBEN<..................................
  91.00 ...................................................................




move 2-4 to 20.............................................0001.00:00001(00)
```

Die Zeilen 2 bis 4 sollen in den Zeilenbereich ab Zeile 20 übertragen werden. Als Schrittweite des Empfangsbereiches 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)......................................0001.00:00001(00)
```

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·····································0001.00:00001(00)
```

Der Zeilenbereich ab Zeile 100 bis Arbeitsdateiende (`100-.$`) soll mit der expliziten Schrittweite 5 in den Zeilenbereich ab Zeile 82 übertragen werden. Da als größtmögliche Zeilennummer für den Empfangsbereich 89 festgelegt wurde, wird die Zeile 90 nicht überschrieben.

```
   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 als größtmögliche Zeilennummer für den Empfangsbereich 89 festgelegt wurde.

9.73 @NOTE – Leere Anweisung

Die Anweisung @NOTE führt keine Aktion aus. Sie wird verwendet, um in EDT-Prozeduren Kommentare einzufügen. Zeilen, die eine @NOTE-Anweisung enthalten, können auch über @GOTO angesprungen werden. Die Anweisung @CONTINUE bietet die gleiche Funktionalität wie @NOTE.

Operation	**Operanden**	**L-Modus**
@NOTE	[comment]	

comment Der Operand `comment` kann beliebigen Text als Kommentar enthalten.

Neben der Kommentierung ist eine häufige Anwendung dieser Anweisung die Definition einer letzten Zeile innerhalb einer EDT-Prozedur, die in einer @GOTO-Anweisung oder in einer @IF-Anweisung als Sprungziel angegeben werden kann. Diese Konstruktion wird benötigt, wenn eine EDT-Prozedur in einer äußeren Schleife mit einem Schleifenzähler aufgerufen wird (z.B. @DO 5,!=%,$), und ein @IF ... RETURN zu einem nicht gewollten Abbruch der äußeren Schleife führen würde. Stattdessen verzweigt man an das Ende der Prozedur, 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   @ @NOTE AUFGABE: WENN EINE ZEILE EIN 'M' ---------------------- (1)
1.01   @ @NOTE          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   @ @NOTE ------------------------------------------------------ (2)
2.01   @END
6.     @DO 1,!=1,$ ---------------------------------------------------- (3)
1.0000 MIT DEM EDT
2.0000 KANN MAN
4.0000 PROZEDUR UM PROZEDUR
6.
```

(1) Hier wird @NOTE zur Kommentierung verwendet.

(2) Hier wird @NOTE benötigt, da es eine letzte Zeile in der Prozedur geben muss, die angesprungen werden kann.

(3) Über @DO mit Schleifenzähler wird die in Arbeitsdatei 1 befindliche Prozedur ausgeführt und dabei auf Arbeitsdatei 0 angewendet.

9.74 @ON (Format 1) – Ausgeben der Zeilen bzw. Zeichenfolgevariablen, die den Suchbegriff enthalten

Dieses Format der @ON-Anweisung bewirkt, dass der EDT den Inhalt jeder Zeile bzw. Zeichenfolgevariablen ausgibt, in der ein Treffer festgestellt wird. Im Dialog erfolgt die Ausgabe nach SYSOUT und im Stapelbetrieb nach SYSLST.

Operation	Operanden	F-Modus, L-Modus
@ON	{ lines / svars } [,...] [:cols[:]] **P**RINT [**A**LL] [**F**] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**S**] [**N**] [**E**]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Der Operand ALL ist nur wirksam bei gleichzeitiger Angabe des Operanden E. In diesem Falle werden alle Treffer einer Zeile optisch hervorgehoben. Ist ALL nicht angegeben, aber E, wird nur der erste Treffer einer Zeile optisch hervorgehoben.

F — Es wird nur die erste Trefferzeile jedes angegebenen Zeilenbereiches ausgegeben. Ist F nicht angegeben, werden alle Trefferzeilen jedes angegebenen Zeilenbereiches ausgegeben. Sind F und E gleichzeitig angegeben, wird der erste Treffer in der ersten Trefferzeile jedes angegebenen Zeilenbereiches optisch hervorgehoben. Bei gleichzeitiger Angabe von F, ALL und E werden alle Treffer in der jeweils ersten Trefferzeile jedes angegebenen Zeilenbereiches optisch hervorgehoben.

R — Die Zeilen werden von rechts nach links durchsucht. Ist `R` nicht angegeben, werden sie von links nach rechts durchsucht.

Wird `R` angegeben, aber weder `ALL` noch `F`, 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). Die Operanden `ALL` und `E` haben in diesem Falle keine Bedeutung.

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int — Erst das `int`-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

S — Die Leerzeile, die der ersten auszugebenden Zeile vorausgeht, wird unterdrückt. Der Operand wirkt nur, falls die Ausgabe nach `SYSLST` erfolgt. Ist `S` nicht angegeben, beginnt die Ausgabe mit einer Leerzeile.

N — Die Trefferzeilen werden ohne zugehörige Zeilennummer ausgegeben. Analog dazu wird die Ausgabe der Namen von Zeichenfolgevariablen (`#S00..#S20`) unterdrückt, falls in Zeichenfolgevariablen nach Treffern gesucht wird. Ist `N` nicht angegeben, werden die Zeilennummern und die Namen von Zeichenfolgevariablen mit ausgegeben.

E — In der Ausgabe der Trefferzeilen am Bildschirm werden die Treffer optisch noch hervorgehoben. Sind `E` und `ALL` gleichzeitig angegeben, werden alle Treffer in der Trefferzeile hervorgehoben, ansonsten nur der erste Treffer. Ist `E` nicht angegeben, werden die Treffer bei der Ausgabe optisch nicht hervorgehoben.

Dieser Operand wirkt nur, wenn `VTCSET ON` eingestellt ist, anderenfalls wird er ignoriert. Bei der Ausgabe langer Zeilen wird die Ausgabe möglicherweise durch ein `%PLEASE ACKNOWLEDGE` vom System unterbrochen. Geschieht das innerhalb einer hervorgehobenen Trefferzeichenfolge, ist der Rest dieser Zeichenfolge, die auf dem nächsten Bildschirm angezeigt wird, nicht mehr hervorgehoben.

Würde die auszugebende Zeile durch Einfügen der Bildschirmsteuerzeichen länger als 32768 Zeichen, wird das Einfügen abgebrochen und im Rest der Zeile werden die Trefferzeichenfolgen nicht mehr hervorgehoben. Es wird zu Information die Meldung `EDT1248` 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'·····················0001.00:00001(01)
```

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';create 100: #l0:#i0-#i1:·····0001.00:00001(01)
```

Alle Zeilen sollen ausgegeben werden, in denen die Zeichenfolge TREFFER vorkommt. Anschließend soll die erste gefundene Trefferzeichenfolge in die Zeile 100 geschrieben werden.

```
    2.0000 DENEN EIN TREFFER VORKOMMT,
    5.0000 TREFFER VOR, SO WIRD NICHTS
 %PLEASE ACKNOWLEDGE
```

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'·····································0001.00:00001(01)
```

Alle Zeilen sollen ausgegeben werden, die jenseits der 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·····································0001.00:00001(01)
```

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'·····0001.00:00001(01)
```

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
```

9.75 @ON (Format 2) – Ausgeben der Anfangsspalte einer Trefferzeichenfolge

Dieses Format der @ON-Anweisung bewirkt, dass der EDT bei der Suche in Arbeitsdateien die Zeilennummern und die Nummern der Spalten ausgibt, in denen die Trefferzeichenfolgen beginnen. Die Ausgabe erfolgt im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST. Wird in der Anweisung kein Suchbegriff spezifiziert, gibt der EDT die Zeilennummern und die Länge jeder Zeile des angegebenen Zeilenbereiches aus.

Beim Durchsuchen von Zeichenfolgevariablen werden anstelle der Zeilennummern und der Zeilenlängen die Namen und Längen der Zeichenfolgevariablen ausgegeben. Die Längenangaben spezifizieren die Anzahl der Zeichen von Zeilen bzw. Zeichenfolgevariablen.

Operation	**Operanden**	**F-Modus, L-Modus**
@ON	{ lines / svars } [,...] [:cols[:]] **CO**LUMN [[**A**LL] [**F**] [**R**] [**P**ATTERN] search [,int]]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Für jede Trefferzeichenfolge einer Zeile wird der Spaltenwert ausgegeben, mit dem die Trefferzeichenfolge beginnt. Ist ALL nicht angegeben, wird nur der Spaltenwert der ersten Trefferzeichenfolge einer Zeile angezeigt.

F — Die Spaltenwerte werden nur für die erste Trefferzeile jedes angegebenen Zeilenbereichs angezeigt. Ist F nicht angegeben, so wird für jede Trefferzeile jedes angegebenen Zeilenbereiches der Spaltenwert ausgegeben.

R	Die Zeilen werden von rechts nach links durchsucht. Ist `R` nicht angegeben, werden sie von links nach rechts durchsucht.
PATTERN	Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.
search	Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.
int	Erst das `int`-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

Die ausgegebenen Spaltenwerte sind fünfstellig, da die Länge von Arbeitsdateizeilen bzw. von Zeichenfolgevariablen einen Maximalwert von 32768 Zeichen annehmen kann. In eine Ausgabezeile können bis zu 10 Spaltenangaben geschrieben werden und danach erfolgt ein Umbruch. Die Zeilennummer wird in den Fortsetzungszeilen nicht ausgegeben.

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·····················································0001.00:00001(01)
```

Die Länge aller Zeilen soll ausgegeben werden.

```
   1.0000 00022
   2.0000 00013
   3.0000 00034
%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 '.....................................0001.00:00001(01)
```

Für die Zeile 1 soll die Spalte ausgegeben werden, in der zum ersten Mal von rechts die Zeichenfolge 'E ' auftritt.

```
    1.0000 00018
 %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 '.....................................01.00:00001(01)
```

Für Zeile 1 sollen alle Spalten ausgegeben werden, in denen die Zeichenfolge `'E '` auftritt.

```
    1.0000 00003 00018
 %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.....................................0001.00:00001(01)
```

Zu allen Zeilen, die mindestens dreimal die Zeichenfolge E enthalten, soll die Spaltennummer des Treffers (drittes Auftreten des Suchbegriffs E) ausgegeben werden.

```
    1.0000 00018
    3.0000 00013
 %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'·····································001.00:00001(01)
```

Alle Zeilen- und Spaltennummern, in denen der Suchbegriff vorkommt, sollen ausgegeben werden.

```
    1.0000 00003 00015 00018
    2.0000 00006 00009
    3.0000 00002 00006 00013 00017 00020 00027 00030
 %PLEASE ACKNOWLEDGE
```

9.76 @ON (Format 3) – Markieren der Zeilen mit Suchbegriff

Dieses Format der @ON-Anweisung bewirkt, dass alle Sätze, in denen ein Treffer festgestellt wird, mit der angegebenen Satzmarkierung gekennzeichnet werden. Im F-Modus wird das Arbeitsfenster auf den ersten Treffersatz positioniert.

Operation	Operanden	F-Modus, L-Modus
@ON	{ lines / svars } [,...] [:cols[:]] **F**IND [**A**LL] [**F**] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**M**ARK [m]]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Die Angabe von `ALL` ist zulässig, aber ohne Bedeutung, da ein Satz nur einmal markiert werden kann.

F — Nur die erste Trefferzeile jedes Zeilenbereiches wird markiert. Wird `F` nicht angegeben, so wird jede Trefferzeile jedes angegebenen Zeilenbereiches markiert.

R — Die Angabe ist ohne Bedeutung, da es nicht von der Suchrichtung abhängt, ob Treffer in einer Zeile gefunden werden.

NOT — Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int — Erst beim `int`-ten Auftreten des Suchbegriffes in einer Zeile wird diese markiert. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

MARK — Die Trefferzeilen werden mit einer Markierung versehen. Ist der Operand nicht angegeben, erhalten die Trefferzeilen die Markierung 1. Sind im Suchbereich Zeichenfolgevariablen angegeben, so wird für diese der Operand `MARK` ignoriert.

m — Nummer der Markierung (`1..9`). Falls `m` nicht angegeben wird, erhalten die Trefferzeilen die Markierung 1.

Bereits vorhandene Satzmarkierungen (z.B. durch vorhergegangene @ON-Anweisungen) bleiben erhalten.

Bezieht sich die Anweisung auf eine real durch @OPEN (Format 2) geöffnete Datei, erfolgt kein Markieren. Es wird lediglich im F-Modus das Arbeitsfenster auf den ersten Treffersatz positioniert. Die explizite Angabe von `MARK` bzw. `MARK m` wird mit der Fehlermeldung `EDT4935` abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Mit der Anweisung @ON lines FIND NOT PATTERN '/'... lassen sich gezielt Leerzeilen (Sätze der Länge 0) markieren, wenn `'/'` das Musterzeichen ist, das für genau ein beliebiges Zeichen steht.

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·····································0001.00:00001(01)
```

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)·····························································.0003.00:00001(01)
```

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 ···································································
```

9.77 @ON (Format 4) – Kopieren von markierten Zeilen

Dieses Format der @ON-Anweisung bewirkt, dass alle mit der angegebenen Satzmarkierung markierten Sätze der zu durchsuchenden Zeilenbereiche in die angegebene Arbeitsdatei kopiert werden.

Operation	**Operanden**	**F-Modus, L-Modus**
@ON	lines[,...] [:cols[:]] **F**IND [**A**LL] [**F**] [**R**] [**NOT**] **M**ARK m [**C**OPY [**T**O]] (procnr) [**K**EEP] [**O**LD]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

cols — Die Spezifikation des Spaltenbereiches ist zulässig, aber ohne Bedeutung, da nur nach Satzmarkierungen gesucht wird.

ALL — Die Angabe ist zulässig, aber ohne Bedeutung, da ein Satz nur einmal kopiert wird.

F — Es wird nur der erste Treffersatz jedes angegebenen Zeilenbereiches mit der angegebenen Satzmarkierung kopiert. Ist `F` nicht angegeben, so werden alle Sätze jedes angegebenen Zeilenbereiches mit der angegebenen Satzmarkierung kopiert.

R — Die Angabe ist ohne Bedeutung, da die Suchrichtung bei Markierungen keine Rolle spielt.

NOT — Wird `NOT` angegeben, so werden diejenigen Sätze kopiert, die eine Satzmarkierung, aber nicht die angegebene Satzmarkierung `m` haben. Wird `NOT` nicht angegeben, so werden die mit der Satzmarkierung `m` markierten Sätze kopiert.

Es ist mit diesem Format nicht möglich, Sätze zu kopieren, die gar keine Satzmarkierung haben.

m — Nummer der Satzmarkierung (`1..9`), nach der gesucht werden soll.

procnr — Nummer der Arbeitsdatei (`0..22`), in die die Trefferzeilen kopiert werden.

Eine aktive oder die aktuelle Arbeitsdatei darf nicht angegeben werden. Ist `OLD` nicht angegeben und werden gleichzeitig Treffersätze gefunden, so wird die Ziel-Arbeitsdatei vor dem Kopieren vollständig gelöscht (siehe @DELETE, Format 2). Werden keine Treffersätze gefunden, bleibt der Inhalt der Ziel-Arbeitsdatei unverändert.

KEEP — Die Zeilennummern der Treffersätze werden beim Kopieren beibehalten. Wird KEEP nicht angegeben, erstellt der EDT die Ziel-Arbeitsdatei ab der aktuellen Zeilennummer, die für jeden kopierten Treffersatz um die aktuelle Schrittweite erhöht wird.

OLD — Der Inhalt der Ziel-Arbeitsdatei wird vor dem Kopieren nicht gelöscht. Eventuell existierende Zeilen mit derselben Zeilennummer in der Ziel-Arbeitsdatei werden überschrieben. Wird OLD nicht angegeben und werden gleichzeitig Treffersätze gefunden, so wird die Ziel-Arbeitsdatei vor dem Kopieren vollständig gelöscht (siehe @DELETE, Format 2).

Ist die angegebene Arbeitsdatei leer oder wurde sie vollständig gelöscht und hat sie den Zeichensatz *NONE, dann erhält sie beim Kopieren den Zeichensatz der aktuellen Arbeitsdatei.

Hat die angegebene Arbeitsdatei einen Zeichensatz, dann werden die zu kopierenden Zeilen in den Zeichensatz dieser Arbeitsdatei konvertiert. Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @ON-Anweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Die Satzmarkierungen werden beim Kopieren in die Ziel-Arbeitsdatei nicht übernommen.

Wird die Anweisung mit [K2] unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

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 1-5 find mark 2 copy to (3) keep ; 3.......................0001.00:00001(01)
```

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




...............................................................0003.00:00001(03)
```

9.78 @ON (Format 5) – Kopieren der Zeilen mit dem Suchbegriff

Mit diesem Format der @ON-Anweisung werden die Trefferzeilen der zu durchsuchenden Zeilenbereiche in die angegebene Arbeitsdatei kopiert.

Operation	Operanden	F-Modus, L-Modus
@ON	lines[,...] [:cols[:]] **F**IND [**A**LL] [**F**] [**R**] [**NOT**] [**P**ATTERN] search [,int] [**C**OPY [**T**O]] (procnr) [**K**EEP] [**O**LD]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Die Angabe ist zulässig, aber ohne Bedeutung, da ein Satz nur einmal kopiert wird.

F — Es wird nur die erste Trefferzeile jedes angegebenen Zeilenbereiches kopiert. Ist `F` nicht angegeben, werden alle Trefferzeilen aus jedem angegebenen Zeilenbereich kopiert.

R — Die Angabe ist ohne Bedeutung, da es nicht von der Suchrichtung abhängt, ob Treffer in einer Zeile gefunden werden.

NOT — Ein Treffer wird erkannt, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht enthalten ist (negatives Suchen).

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int — Erst das `int`-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

procnr Nummer der Arbeitsdatei (0..22), in die die Trefferzeilen kopiert werden.

Eine aktive oder die aktuelle Arbeitsdatei darf nicht angegeben werden. Ist OLD nicht angegeben und werden gleichzeitig Treffersätze gefunden, so wird die Ziel-Arbeitsdatei vor dem Kopieren vollständig gelöscht (siehe @DELETE, Format 2). Werden keine Treffersätze gefunden, bleibt der Inhalt der Ziel-Arbeitsdatei unverändert.

KEEP Die Zeilennummern der Treffersätze werden beim Kopieren beibehalten. Wird KEEP nicht angegeben, erstellt der EDT die Ziel-Arbeitsdatei ab der aktuellen Zeilennummer, die für jeden kopierten Treffersatz um die aktuelle Schrittweite erhöht wird.

OLD Der Inhalt der Ziel-Arbeitsdatei wird vor dem Kopieren nicht gelöscht. Eventuell existierende Zeilen mit derselben Zeilennummer in der Ziel-Arbeitsdatei werden überschrieben. Wird OLD nicht angegeben und werden gleichzeitig Treffersätze gefunden, so wird die Ziel-Arbeitsdatei vor dem Kopieren vollständig gelöscht (siehe @DELETE, Format 2).

Hat die angegebene Arbeitsdatei einen Zeichensatz, dann werden die zu kopierenden Zeilen in den Zeichensatz dieser Arbeitsdatei konvertiert. Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die @ON-Anweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

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·····························0001.00:00001(01)
```

Sätze mit dem Suchbegriff STR. sollen 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·····························································0001.00:00001(05)
```

Es wird in die Arbeitsdatei 1 verzweigt.

```
   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 & :10-20: find pattern 'M*A' copy to (6) ; 6·················0001.00:00001(01)
```

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 ···································································
·····························································0001.00:00001(06)
```

9.79 @ON (Format 6) – Ersetzen der Trefferzeichenfolge

Im Trefferfall bewirkt dieses Format der @ON-Anweisung, dass die Trefferzeichenfolge durch eine angegebene Zeichenfolge ersetzt wird.

Operation	Operanden	F-Modus, L-Modus
@ON	{ lines / svars } [,...] [:cols[:]] **C**HANGE [**A**LL] [**F**] [**R**] [**P**ATTERN] search [,int] [**T**O] string [**V**]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Alle Trefferzeichenfolgen innerhalb einer Zeile werden durch die angegebene Zeichenfolge ersetzt. Ist `ALL` nicht angegeben, wird die Ersetzung nur für die erste Trefferzeichenfolge in einer Zeile durchgeführt.

F — In jedem angegebenen Zeilenbereich wird die Ersetzung nur für die erste Trefferzeile durchgeführt. Ist `F` nicht angegeben, findet die Ersetzung für jede Trefferzeile jedes angegebenen Zeilenbereiches statt.

R — Die Zeilen werden von rechts nach links durchsucht. Ist `R` nicht angegeben, werden sie von links nach rechts durchsucht.

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int
: Erst das `int`-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

string
: Zeichenfolge, die die Trefferzeichenfolge ersetzen soll. Auch die Angabe einer leeren Zeichenfolge ist erlaubt.

  Die Zeichenfolge wird in den Zeichensatz der Arbeitsdatei bzw. der Zeichenfolgevariablen konvertiert. Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), anderenfalls wird die @ON-Anweisung abgebrochen und die Fehlermeldung `EDT5453` ausgegeben.

V
: Die Anzahl der Trefferzeilen wird in der Ganzzahlvariablen `#I2` und die Gesamtanzahl der Treffer in der Ganzzahlvariablen `#I3` gespeichert. Die Zählung erfolgt unabhängig davon, ob die Ersetzung stattfindet oder nicht (wegen Längenüberschreitung). Der Operand `V` wirkt nur bei gleichzeitiger Angabe des Operanden `ALL`. Ist `V` nicht angegeben, bleiben die Ganzzahlvariablen `#I2` und `#I3` unverändert.

Überschreitet ein Satz durch die Ersetzung die maximale Satzlänge von 32768 Zeichen, dann wird diese nicht durchgeführt und stattdessen die Meldung `EDT1937` ausgegeben, die Abarbeitung aber fortgesetzt.

Wird die Anweisung mit [K2] unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

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 ''.........0001.00:00001(01)
```

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----+----2----+----3----+----4----+----5----+----6----+----7--
  1.00 ABCDEFGHIJKLMNOPQRSTUVWXYZ<·····································
  2.00 WER IST HIER STUR WIE OSKAR ?<···································
  3.00 ·····························································
```

9.80 @ON (Format 7) – Ersetzen oder Einfügen vor oder nach der Trefferzeichenfolge

Mit diesem Format der @ON-Anweisung kann vor oder nach einer Trefferzeichenfolge in Arbeitsdateizeilen bzw. Zeichenfolgevariablen Text eingefügt oder ersetzt werden.

Operation	Operanden	F-Modus, L-Modus
@ON	{ lines / svars } [,...] [:cols[:]] **F**IND [**A**LL] [**F**] [**R**] [**P**ATTERN] search [,int] { **C**HANGE / **I**NSERT } { **P**REFIX / **S**UFFIX } string	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Bei einer Texteinfügung wird vom EDT vor bzw. hinter allen Trefferzeichenfolgen einer Zeile die Zeichenfolge `string` eingefügt. Zur Ermittlung der weiteren Treffer wird bei der Suche von links nach rechts hinter und bei der Suche von rechts von links vor der Einfügung wieder aufgesetzt.

Bei einer Textersetzung wird vor bzw. hinter allen Trefferzeichenfolgen einer Zeile der Zeileninhalt durch die Zeichenfolge `string` ersetzt. Zur Ermittlung der weiteren Treffer wird die restliche Suche vor bzw. hinter der Ersetzung fortgeführt.

Falls bei der Suche von links nach rechts der Text hinter einer Trefferzeichenfolge ersetzt wird, ist die Angabe von `ALL` ohne Bedeutung. Dasselbe gilt, wenn bei der Suche von rechts nach links der Text vor einer Trefferzeichenfolge ersetzt wird.

Ist ALL nicht angegeben, wird nur vor bzw. hinter der ersten Trefferzeichenfolge einer Zeile Text eingefügt oder ersetzt.

F — In jedem angegebenen Zeilenbereich wird das Ersetzen bzw. das Einfügen nur für die erste Trefferzeile durchgeführt. Ist F nicht angegeben, findet das Ersetzen bzw. das Einfügen für jede Zeile jedes angegebenen Zeilenbereiches statt.

R — Die Zeilen werden von rechts nach links durchsucht. Ist R nicht angegeben, werden sie von links nach rechts durchsucht.

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int — Erst das int-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für int sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für int beträgt 1.

CHANGE — Der vor oder nach der Trefferzeichenfolge stehende Text bis zum Satzanfang bzw. zum Satzende wird durch die mit string angegebene Zeichenfolge ersetzt.

INSERT — Die mit string angegebene Zeichenfolge wird vor oder nach der Trefferzeichenfolge eingefügt.

PREFIX — Die mit string angegebene Zeichenfolge ersetzt den Zeileninhalt vor der Trefferzeichenfolge oder wird vor der Trefferzeichenfolge eingefügt.

SUFFIX — Die mit string angegebene Zeichenfolge ersetzt den Zeileninhalt nach der Trefferzeichenfolge oder wird nach der Trefferzeichenfolge eingefügt.

string — Zeichenfolge, die den Text vor oder nach der Trefferzeichenfolge ersetzen bzw. vor oder nach der Trefferzeichenfolge eingefügt werden soll. Auch die Angabe einer leeren Zeichenfolge ist erlaubt.

Die Zeichenfolge wird in den Zeichensatz der Arbeitsdatei bzw. der Zeichenfolgevariablen konvertiert. Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), anderenfalls wird die @ON-Anweisung abgebrochen und die Fehlermeldung EDT5493 ausgegeben.

Die Zeichenfolge string sollte von den Operanden PREFIX bzw. SUFFIX durch ein Leerzeichen getrennt sein.

Das Ersetzen bzw. das Einfügen findet nicht statt, wenn ein Satz dadurch die maximale Satzlänge von 32768 Zeichen überschreitet. Stattdessen wird in diesem Falle die Meldung EDT1937 ausgegeben, die Abarbeitung aber fortgesetzt.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Beispiel 1

```
  23.00 ·····················································
fstat '$user2.bsp.'·····························0001.00:00001(01)
```

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.'·············0001.00:00001(09)
```

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 ·····················································
:
:
:
on & find pattern 'BSP./' insert suffix ''''·················0001.00:00001(09)
```

Das abschließende Hochkomma wird hinter den vier Dateinamen $USER2.BSP.1 bis $USER2.BSP.4 eingefügt.

```
   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·····································0001.00:00001(09)
```

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',3 insert suffix '++++'····················0001.00:00001(01)
```

Im Zeilenbereich 1 bis 2 soll von rechts nach links nach dem dritten Auftreten der Zeichenfolge 11 gesucht und im Trefferfall dahinter ++++ eingefügt werden.

```
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
   1.00 A11B11C11D11E11F11++++G11H11<·····································
   2.00 A1111B1111C1111D1111++++E1111<·····································
   3.00 ··········································································
:
:
:
on &:4 find '111',3 change suffix '####'····················0001.00:00001(01)
```

In Zeile 1 trat der Suchbegriff zum dritten Mal von rechts in den Spalten 17-18 auf.

In Zeile 2 trat der Suchbegriff zum ersten Mal in den Spalten 24-25, zum zweiten Mal in den Spalten 22-23 und zum dritten Mal 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 A11B11C11D11E11F11++++G11H11<·····································
  2.00 A1111B1111C111D111####<···········································
  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 12-14 und zum dritten Mal als Treffer in den Spalten 17-19 auf. Nach dem Treffer wurde der Zeilenrest durch die Zeichenfolge #### ersetzt.

9.81 @ON (Format 8) – Löschen der Trefferzeichenfolge

Im Trefferfall bewirkt dieses Format der @ON-Anweisung, dass beim Durchsuchen von Zeileninhalten bzw. Zeichenfolgevariablen die Trefferzeichenfolge gelöscht wird. Der restliche Inhalt der Arbeitsdateizeilen bzw. Zeichenfolgevariablen bleibt in diesem Falle erhalten.

Operation	**Operanden**	**F-Modus, L-Modus**
@ON	{ lines / svars } [,...] [:cols[:]] **D**ELETE [**A**LL] [**F**] [**R**] [**P**ATTERN] search [,int]	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Alle Trefferzeichenfolgen in einer Zeile werden gelöscht. Ist `ALL` nicht angegeben, wird nur die erste Trefferzeichenfolge einer Zeile gelöscht.

F — In jedem angegebenen Zeilenbereich wird das Löschen von Trefferzeichenfolgen nur für die erste Trefferzeile durchgeführt. Wird `F` nicht angegeben, so findet das Löschen von Trefferzeichenfolgen für jede Zeile jedes angegebenen Zeilenbereiches statt.

R — Die Zeilen werden von rechts nach links durchsucht. Ist `R` nicht angegeben, so werden sie von links nach rechts durchsucht.

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int — Erst das `int`-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

Wird die Anweisung mit [K2] unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Beispiel

```
   1.00 XXXYYYZZZ *** XXXYYYZZZ ### XXXYYYZZZ %%%<.............................
   2.00 AAA XXXYYYZZZ BBB XXXYYYZZZ CCC XXXYYYZZZ<.............................
   3.00 ........................................................................



on & delete f r 'XXXYYYZZZ'...................................0001.00:00001(01)
```

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'...................................0001.00:00001(01)
```

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` bei jedem Vorkommen gelöscht werden.

```
   1.00  ***  ###  %%%<........................................................
   2.00 AAA  BBB  CCC<.........................................................
   3.00 ........................................................................
```

9.82 @ON (Format 9) – Löschen vor oder nach der Trefferzeichenfolge

Dieses Format der @ON-Anweisung bewirkt, dass im Trefferfalle der Inhalt einer Arbeitsdateizeile bzw. Zeichenfolgevariablen vor oder nach der Trefferzeichenfolge gelöscht wird.

Operation	**Operanden**	**F-Modus, L-Modus**
@ON	{ lines / svars } [,...] [:cols[:]] **F**IND [**A**LL] [**F**] [**R**] [**P**ATTERN] search [,int] **D**ELETE { **P**REFIX / **S**UFFIX }	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Bei jedem Auffinden einer Trefferzeichenfolge innerhalb der Zeile wird der Text vor bzw. hinter der Trefferzeichenfolge gelöscht. Die Angabe von `ALL` ist ohne Bedeutung, wenn bei der Suche von links nach rechts der Text hinter der Trefferzeichenfolge gelöscht wird. Dasselbe gilt, wenn bei der Suche von rechts nach links der Text vor der Trefferzeichenfolge gelöscht wird. Ist `ALL` nicht angegeben, wird das Löschen nur für die erste Trefferzeichenfolge einer Zeile durchgeführt.

F — In jedem angegebenen Zeilenbereich wird das Löschen des Textes vor bzw. hinter den Trefferzeichenfolgen nur für die erste Trefferzeile durchgeführt. Wird `F` nicht angegeben, so wird das Löschen des Textes vor bzw. hinter den Trefferzeichenfolgen für jede Trefferzeile jedes angegebenen Zeilenbereiches durchgeführt.

R — Die Zeilen werden von rechts nach links durchsucht. Ist R nicht angegeben, so werden sie von links nach rechts durchsucht.

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search — Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int — Erst das int-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für int sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für int beträgt 1.

PREFIX — Der Inhalt der Trefferzeile vor der Trefferzeichenfolge bis zum Satzanfang wird gelöscht.

SUFFIX — Der Inhalt der Trefferzeile nach der Trefferzeichenfolge bis zum Satzende wird gelöscht.

Wird die Anweisung mit [K2] unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Beispiel

```
   1.00 ABABAB ABABAB ABABAB ABABAB<.......................................
   2.00 ABABABABABABABABABABABAB<..........................................
   3.00 ...................................................................



on %.-.$ find 'AB'*3,4 delete prefix.........................0001.00:00001(01)
```

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 ABABAB<............................................................
   3.00 ...................................................................
```

In Zeile 1 wurde der Suchbegriff zum vierten Mal als Treffer ab Spalte 22 gefunden.

In Zeile 2 wird der Suchbegriff ab Spalte 1 zum ersten Mal, ab Spalte 7 zum zweiten Mal, ab Spalte 13 zum dritten Mal und ab Spalte 19 als Treffer zum vierten Mal gefunden. Bei der Suche nach dem zweiten und weiteren Treffern wird jeweils hinter einer Trefferzeichenfolge wieder aufgesetzt.

9.83 @ON (Format 10) – Löschen der Zeilen bzw. Zeichenfolgevariablen, die den Suchbegriff enthalten

Dieses Format der @ON-Anweisung bewirkt, dass die Zeile bzw. der Inhalt einer Zeichenfolgevariablen, die den Suchbegriff enthalten, gelöscht wird. Zeichenfolgevariablen werden in diesem Falle wieder mit einem Leerzeichen initialisiert und erhalten den Zeichensatz EDF041.

Operation	Operanden	F-Modus, L-Modus
@ON	{ lines / svars } [,...] [:cols[:]] **F**IND [**A**LL] [**F**] [**R**] [**NOT**] [**P**ATTERN] search [,int] **D**ELETE	

lines — Einer oder mehrere Zeilenbereiche, in denen gesucht werden soll.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen gesucht werden soll.

cols — Zusammenhängender Spaltenbereich, auf den die Suche eingeschränkt werden soll.

Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis zum Zeilenende spezifiziert. Ist die erste Spaltenangabe bereits größer als die Zeilenlänge, wird die Zeile bzw. Zeichenfolgevariable ignoriert.

Wird kein Spaltenbereich angegeben, wird der mit @SEARCH-OPTION eingestellte Spaltenbereich verwendet.

ALL — Die Angabe ist zulässig, aber ohne Bedeutung, da eine Zeile nach dem Auffinden der ersten Trefferzeichenfolge immer gelöscht wird.

F — Nur die erste Trefferzeile jedes angegebenen Zeilenbereiches wird gelöscht. Wird F nicht angegeben, so wird jede Zeile jedes Zeilenbereiches gelöscht, die die Trefferzeichenfolge enthält.

R — Die Angabe ist ohne Bedeutung, da es nicht von der Suchrichtung abhängt, ob Treffer in einer Arbeitsdateizeile gefunden werden.

NOT — Eine Zeile wird gelöscht, wenn im angegebenen Spaltenbereich einer Zeile der Suchbegriff nicht gefunden wird (negatives Suchen).

PATTERN — Die im Suchbegriff vorkommenden Musterzeichen werden interpretiert.

search
: Suchbegriff, der im Suchbereich aufgefunden werden soll (Details siehe Abschnitt „Suchen mit @ON“ auf Seite 81). Die Angabe einer leeren Zeichenfolge ist nicht erlaubt.

int
: Erst das `int`-te Auftreten des Suchbegriffes in einer Zeile ist als erster Treffer zu werten. Als Eingaben für `int` sind alle Werte zwischen 1 und 32768 erlaubt. Der Standardwert für `int` beträgt 1.

Wird die Anweisung mit [K2] unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Sollen in einem Bereich alle Leerzeilen (Sätze der Länge 0) gelöscht werden, kann dies mit der Anweisung @ON lines FIND NOT PATTERN '/' DELETE geschehen, wenn `'/'` das Musterzeichen ist, das für genau ein beliebiges Zeichen steht.

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...................................0001.00:00001(01)
```

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...................................0004.00:00001(01)
```

Die Zeilen 1, 2 und 3 wurden gelöscht. Anschließend sollen in der gesamten Arbeitsdatei alle Zeilen gelöscht werden, in denen hinter Spalte 7 das Zeichen `A` auftritt.

```
   5.00 1  ABC<·····························································
   6.00 1<··································································
   7.00 ····································································
```

```
on & find ' '*2 delete·····································0005.00:00001(01)
```

Zeile 4 wurde gelöscht. Anschließend sollen in der gesamten Arbeitsdatei alle Zeilen gelöscht werden, in der zwei aufeinander folgende Leerzeichen '␣␣' auftreten.

```
   6.00 1<··································································
   7.00 ····································································
```

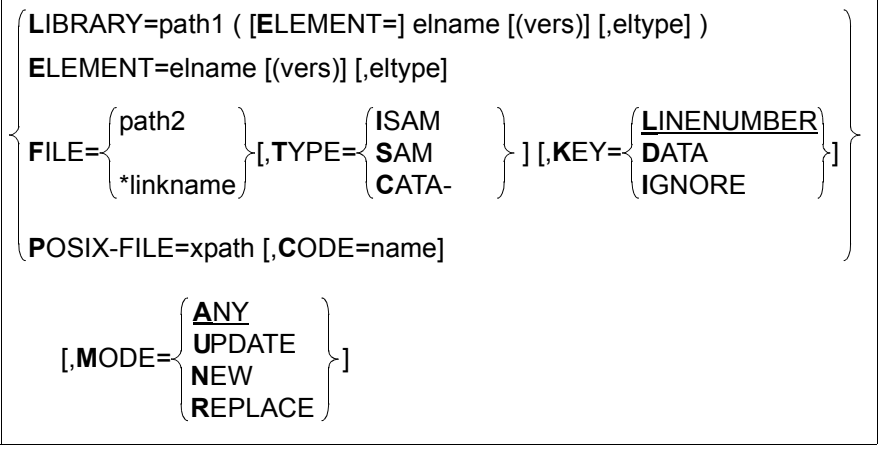
Zeile 5 wurde gelöscht.

9.84 @OPEN (Format 1) – Öffnen und Einlesen einer Datei

Mit @OPEN (Format 1) wird eine existierende Datei geöffnet und in die aktuelle Arbeitsdatei eingelesen oder eine Datei neu erzeugt und zur Bearbeitung geöffnet.

Die Arbeitsdatei muss leer sein und eine SAM-Datei, eine ISAM-Datei oder ein Bibliothekselement darf nicht bereits in einer anderen Arbeitsdatei geöffnet sein. Die Datei bleibt geöffnet, bis sie durch @CLOSE geschlossen wird.

Wenn in diesem Abschnitt von Datei die Rede ist, dann kann dies eine SAM-Datei, eine ISAM-Datei, ein Bibliothekselement oder eine POSIX-Datei sein.

Operation	Operanden	F-Modus, L-Modus
@OPEN		

LIBRARY= Ein Bibliothekselement soll geöffnet und eingelesen werden. Dieses wird durch explizite Angabe des Bibliotheksnamens und der Elementbezeichnung bestimmt.

path1 Name der Bibliothek.

elname Name des Elements.

vers Version des gewünschten Elements (siehe Handbuch LMS [14]). Wird `vers` nicht angegeben oder wird `*STD` angegeben, wird die höchste vorhandene Version des Elementes gewählt.

eltype Typ des Elements. Zulässige Typangaben sind `S`, `M`, `P`, `J`, `D`, `X`, `*STD` und freie Typnamen mit entsprechendem Basistyp. Wird `eltype` nicht angegeben, wird der mit @PAR ELEMENT-TYPE voreingestellte Wert verwendet. Die zulässigen Elementtypen und deren Bedeutung sind im Kapitel „Dateibearbeitung“ auf Seite 137 beschrieben.

ELEMENT=... Ein Bibliothekselement soll geöffnet und eingelesen werden. Dieses wird durch die Elementbezeichnung ohne Angabe des Bibliotheknamens bestimmt. Es wird implizit die mit @PAR LIBRARY voreingestellte Bibliothek verwendet (sofern @PAR LIBRARY spezifiziert wurde, andernfalls wird die Fehlermeldung EDT5181 ausgegeben).

Die Operanden elname, vers und eltype haben die gleiche Bedeutung wie bei expliziter Angabe der Bibliothek (siehe oben).

FILE= Eine BS2000-Datei soll geöffnet und eingelesen werden.

path2 Name der BS2000-Datei (voll qualifizierter Dateiname), die geöffnet werden soll.

*linkname Dateikettungsname der BS2000-Datei, die geöffnet und eingelesen werden soll. Der Dateiname und die Dateiattribute sind in der Task File Table abgelegt. Dateien mit vom Standard abweichenden Attributen können so neu angelegt werden. Der Dateikettungsname darf nicht mit den Spezial-Dateinamen *BY-PROGRAM vereinbart worden sein. Dies führt zum Fehler EDT4923. Ist der Dateikettungsname nicht definiert, wird die Anweisung mit dem Fehler EDT5480 abgewiesen.

Ist der Dateikettungsname mit dem Spezial-Dateinamen *DUMMY vereinbart worden, wird dies wie eine nicht existierende Datei behandelt, es entsteht aber keine Datei.

TYPE= Legt die Zugriffsmethode der BS2000-Datei fest.

SAM Falls die Datei noch nicht existiert, wird eine SAM-Datei angelegt, andernfalls wird die Angabe ignoriert. Dies ist der Standardwert beim Neuanlegen von Dateien.

ISAM Falls die Datei noch nicht existiert, wird eine ISAM-Datei angelegt, andernfalls wird die Angabe ignoriert.

CATALOG Falls die Datei schon existiert, werden die Attribute aus dem Katalogeintrag übernommen, andernfalls wird die Meldung EDT5281 ausgegeben. Die Zugriffsmethode wird durch das FCBTYPE-Attribut im Katalogeintrag bestimmt. Dies ist der Standardwert für existierende Dateien.

KEY= Legt für ISAM-Dateien den Ort fest, an dem der ISAM-Schlüssel in der Arbeitsdatei abgelegt ist. Für andere Dateitypen wird der Operand ignoriert.

LINENUMBER
: Der ISAM-Schlüssel wird als Zeilennummer in der Arbeitsdatei abgelegt. Dies ist der Standardwert. Kann der ISAM-Schlüssel nicht als Zeilennummer interpretiert werden, weil die Schlüsselposition vom Standard abweicht, die Schlüssellänge zu groß ist oder die Schlüssel nicht numerisch sind, wird die Meldung `EDT5459` ausgegeben und die Datei nicht geöffnet.

DATA
: Der ISAM-Schlüssel wird Bestandteil des Datenbereichs der Arbeitsdatei.

IGNORE
: Der ISAM-Schlüssel wird nicht in der Arbeitsdatei abgelegt. Weicht die Schlüsselposition vom Standard ab, wird die Meldung `EDT5466` ausgegeben und die Datei nicht geöffnet.

POSIX-FILE= Es soll eine POSIX-Datei geöffnet werden.

xpath
: Pfadname der POSIX-Datei, die geöffnet werden soll.

  Der Operand `xpath` kann auch als Zeichenfolgevariable angegeben werden. Er muss als Zeichnfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

CODE= Es wird festgelegt, welcher Zeichensatz für die POSIX-Datei angenommen wird. Da POSIX-Dateien innerhalb des POSIX-Dateisystems kein Zeichensatz zugeordnet werden kann, ist hier eine Angabe des Anwenders nötig.

  Wird `CODE` nicht angegeben, wird für die Datei der mit @PAR CODE eingestellte Zeichensatz angenommen.

name
: Zeichensatz der einzulesenden POSIX-Datei. Als `name` muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).

EBCDIC
: Das Schlüsselwort `EBCDIC` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `EDF041` unterstützt.

ISO
: Das Schlüsselwort `ISO` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `ISO88591` unterstützt.

MODE= Legt fest, ob die Datei schon vorhanden sein soll.

- ANY: Falls die Datei schon existiert, wird sie zur Bearbeitung geöffnet und eingelesen, andernfalls wird sie neu angelegt und zur Bearbeitung geöffnet. Dies ist der Standardwert.
- UPDATE: Die Datei, die zur Bearbeitung geöffnet und eingelesen werden soll, muss bereits existieren oder über den Dateikettungsnamen mit *DUMMY verknüpft sein, andernfalls wird je nach Dateityp die Meldung EDT5281, EDT5284 oder EDT5310 ausgegeben.
- NEW: Die Datei wird neu angelegt und zur Bearbeitung geöffnet. Sie darf noch nicht vorhanden sein, andernfalls wird je nach Dateityp die Meldung EDT5258, EDT5273 oder EDT5311 ausgegeben.
- REPLACE: Falls die Datei schon existiert, wird sie zur Bearbeitung geöffnet, ihr alter Inhalt wird aber gelöscht und nicht in die Arbeitsdatei eingelesen. Andernfalls wird sie neu angelegt und zur Bearbeitung geöffnet.

Ist die aktuelle Arbeitsdatei nicht leer oder ist in ihr bereits eine Datei geöffnet, wird die Anweisung mit der Meldung EDT5191 bzw. EDT5180 abgewiesen.

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar bzw. kann eine existierende Datei nicht erfolgreich eingelesen werden, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Die Zeilennummern der Arbeitsdatei werden beim Einlesen von ISAM-Dateien mit Operand KEY=LINENUMBER aus dem ISAM-Schlüssel der Datei gebildet, in allen anderen Fällen nach dem Verfahren Einfügen zwischen zwei Zeilen (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Hat beim Öffnen einer existierenden Datei die leere Arbeitsdatei den Zeichensatz *NONE, erhält die Arbeitsdatei den Zeichensatz der einzulesenden Datei. Ist dieser *NONE, erhält die Arbeitsdatei den Zeichensatz EDF003IRV.

Hat beim Öffnen einer existierenden Datei die leere Arbeitsdatei bereits einen Zeichensatz (z.B. durch vorhergehendes @CODENAME), dann werden die einzulesenden Sätze vom Zeichensatz der Datei in den Zeichensatz der Arbeitsdatei konvertiert. Enthält die einzulesende Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht eingelesen und die Fehlermeldung EDT5453 ausgegeben.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird das Lesen mit der Meldung EDT5454 abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Beispiel

```
@OPEN LIBRARY=PROGLIB(ELEMENT=TEST)
```

Das Element TEST der Bibliothek PROGLIB wird geöffnet und in die aktuelle Arbeitsdatei eingelesen. Dabei wird die höchste vorhandene Version und der mit @PAR ELEMENT-TYPE voreingestellte Typ verwendet. Die Arbeitsdatei muss vorher leer sein.

```
@PAR LIBRARY=BIB1
@SET #S01='PROC.EX'
@OPEN ELEMENT=.#S01(V01),J
```

Das Element mit dem Namen PROC.EX, der Version V01 und dem Elementtyp J (Prozedur) aus der Bibliothek BIB1 wird geöffnet und in die aktuelle Arbeitsdatei eingelesen.

```
@OPEN FILE=DATEI1,TYPE=ISAM,MODE=NEW
```

Die ISAM-Datei DATEI1 wird neu angelegt und geöffnet. Die Zeilennummern der Arbeitsdatei stellen den ISAM-Schlüssel dar.

```
@OPEN POSIX-FILE=/home/user1/test/data,CODE=UTF8
```

Die POSIX-Datei data in dem Verzeichnis /home/user1/test mit dem Zeichensatz UTF8 wird geöffnet und in die aktuelle Arbeitsdatei eingelesen.

9.85 @OPEN (Format 2) – Reale Bearbeitung einer ISAM-Datei

Mit @OPEN (Format 2) wird eine ISAM-Datei zur Bearbeitung direkt auf der Platte geöffnet. Sie kann bereits existieren, vor dem Öffnen neu erzeugt werden oder als Kopie einer existierenden SAM- oder ISAM-Datei entstehen.

In die aktuelle Arbeitsdatei wird nur der gerade benötigte Teil der ISAM-Datei eingelesen. Im F-Modus sind das die Sätze, die ganz oder teilweise am Bildschirm angezeigt werden. Im L-Modus werden erst bei der Ausführung einer EDT-Anweisung die benötigten Sätze eingelesen. Die Datei bleibt bis zum Ende ihrer Bearbeitung durch @CLOSE geöffnet.

Das Öffnen von ISAM-Dateien zur realen Bearbeitung ist nur in der Arbeitsdatei 0 möglich. Diese muss leer sein oder eine mit @OPEN (Format 2) real geöffnete Datei enthalten.

Operation	**Operanden**	**F-Modus, L-Modus**
@OPEN	[file1] [(ver)] [[**K**EY] [**A**S file2 [[,]**O**VERWRITE]]	

file1 Name der Datei, die geöffnet oder kopiert werden soll. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen.

Die symbolische Bezeichnung `'/'` für eine Datei, für die der Dateikettungsname `EDTSAM` bzw. `EDTISAM` durch das `/SET-FILE-LINK`-Kommando vergeben wurde, ist hier nicht zulässig.

Falls eine Datei mit diesem Namen nicht existiert und der Operand `AS` nicht vorhanden ist, wird eine ISAM-Datei erzeugt und geöffnet. Ist der Operand `AS` vorhanden, wird die Anweisung mit der Meldung `EDT4971` abgewiesen.

Ist die Datei eine SAM-Datei und ist der Operand `AS` nicht vorhanden, wird die Meldung `EDT4934` ausgegeben.

Fehlt der Operand `file1`, dann wird die Datei geöffnet oder kopiert, die mit @FILE global voreingestellt worden ist. Gibt es keine solche Voreinstellung, wird die @OPEN-Anweisung mit der Meldung `EDT5484` abgewiesen.

ver Versionsnummer der Datei. Wird für eine existierende Datei die falsche Versionsnummer angegeben, wird die Anweisung mit der Meldung `EDT4985` abgewiesen. Existiert die zu öffnende Datei nicht, wird die Angabe ignoriert.

KEY Nur wirksam, wenn eine SAM-Datei in eine ISAM-Datei kopiert wird, die dann real geöffnet wird. Die ersten 8 Zeichen eines jeden Satzes der SAM-Datei bilden dann den Schlüssel der ISAM-Datei und sind nach dem Einlesen als Zeilennummer und nicht als Satzinhalt zu interpretieren. Derartige Sätze lassen sich mit @WRITE bei Angabe des Operanden `KEY` erstellen.

Ist `file1` eine ISAM-Datei, wird der Operand `KEY` ignoriert.

AS ... Die Datei file1 wird in eine ISAM-Datei kopiert. Wenn das Kopieren gelungen ist, wird die Kopie zur realen Bearbeitung geöffnet.

file2 Name der ISAM-Datei, in die file1 kopiert wird und die dann anschließend geöffnet wird.
Die symbolische Bezeichnung '/' für eine Datei, für die der Dateikettungsname EDTSAM bzw. EDTISAM durch das /SET-FILE-LINK-Kommando vergeben wurde, ist hier nicht zulässig.
Falls eine Datei mit diesem Namen nicht existiert, wird vor dem Kopieren eine ISAM-Datei erzeugt.

Falls die Datei file2 bereits existiert und der Operand OVERWRITE nicht angegeben ist, wird im Dialogbetrieb folgende Abfrage ausgegeben:

```
%  EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)?
```

Die Datei file2 wird nur dann überschrieben und geöffnet, wenn der Benutzer mit Y antwortet. Im Stapelbetrieb wird die Datei überschrieben und geöffnet.

Die Dateinamen file1 und file2 dürfen nicht gleich sein, sonst wird die Anweisung mit der Meldung EDT5489 abgewiesen.

OVERWRITE
Unterdrückt die Abfrage EDT0296 für eine vorhandene Datei file2. Die Datei wird überschrieben und geöffnet. Existiert die Datei file2 noch nicht, ist OVERWRITE wirkungslos.

Nach @OPEN (Format 2) werden die Anweisungen @RENUMBER, @SORT und @COMPARE abgewiesen. Sätze von real geöffneten Dateien können nicht markiert werden (Markierungs-Kurzanweisung, @ON, Format 3) und werden nicht automatisch gesichert (siehe @AUTOSAVE).

Ist bereits eine Datei zur realen Bearbeitung geöffnet, dann wird diese durch eine weitere @OPEN (Format 2)-Anweisung implizit geschlossen und die Arbeitsdatei wird gelöscht. Erst dann wird die zweite Datei geöffnet.

Die Zeilennummern der Arbeitsdatei werden beim Einlesen aus dem ISAM-Schlüssel der Datei gebildet. Beim Kopieren einer SAM-Datei in eine ISAM-Datei werden die ISAM-Schlüssel aus den ersten 8 Zeichen der Datei gebildet, wenn der Operand KEY angegeben wird, andernfalls werden sie nach dem Verfahren „Einfügen bei der aktuellen Zeilennummer“ gebildet (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Die Arbeitsdatei erhält den Zeichensatz der einzulesenden Datei oder (wenn diese noch nicht existiert) den durch die aktuellen Systemeinstellungen festgelegten Zeichensatz. Ist dieser *NONE, erhält die Arbeitsdatei den Zeichensatz EDF03IRV. Der Zeichensatz der Arbeitsdatei kann dann bis zum Schließen der Datei nicht mehr geändert werden.

Hat die Arbeitsdatei bereits einen Zeichensatz, der von diesem verschieden ist, wird die Anweisung mit der Meldung EDT5452 abgewiesen.

Ist der zu öffnenden ISAM-Datei der Dateikettungsname EDTMAIN zugeordnet und wird die Datei neu erzeugt, so werden die in der TFT hinterlegten Attribute beim Erzeugen der Datei berücksichtigt, ansonsten wird eine Datei mit Standard-Attributen angelegt. Das gilt nicht, wenn eine vorhandene ISAM-Datei kopiert und dann geöffnet wird. Sie erhält dann die Attribute der zu kopierenden Datei.

Fehler in der Datei (z.B. nicht-numerische Schlüssel, illegale Byte-Folgen u.a.) werden möglicherweise erst viel später bemerkt, wenn die entsprechenden Sätze gelesen werden sollen. Beim Öffnen werden nicht alle Sätze der Datei eingelesen. Später festgestellte Fehler meldet dann das jeweilige Kommando, bei dessen Abarbeitung er auftrat. In diesen Fällen wird dann immer die real geöffnete Datei automatisch geschlossen.

Enthält eine real geöffnete ISAM-Datei Sätze mit gleichen Schlüsselwerten (*duplicate keys*), ist das Verhalten des EDT nicht in allen Fällen vorhersagbar. Je nach Art und Weise des Lesens kann es sein, dass der erste oder auch der letzte der Sätze mit gleichem Schlüssel angezeigt wird. Überschrieben wird dann immer der erste Satz. Bemerkt der EDT das Auftreten solcher gleichen Schlüsselwerte, so beendet er wie bei anderen Dateifehlern die Bearbeitung, gibt die Meldung EDT5445 aus und schließt die real geöffnete Datei.

In einer real geöffneten ISAM-Datei mit kürzerem Schlüssel (weniger als 8 Zeichen) können keine Sätze erzeugt werden, deren Zeilennummer zu groß ist, um als Schlüssel korrekt dargestellt werden zu können. Anweisungen die dies versuchen, werden mit der Fehlermeldung EDT5446 abgebrochen.

Ist der AS-Operand angegeben und ist die erste Datei eine ISAM-Datei, so wird sie durch ein implizites COPY-FILE-Kommando kopiert. Ist sie eine SAM-Datei, wird das Kopieren durch eine implizite @READ-Anweisung und eine implizite @SAVE-Anweisung realisiert.

In diesem Fall kann die Anweisung mit K2 unterbrochen werden. Wird sie unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Hinweis

Das Kommando ist geeignet Dateien zu bearbeiten, die wegen ihrer Größe sonst nicht vollständig in den EDT zur Bearbeitung geladen werden können.

9.86 @P-KEYS – Belegen programmierbarer Tasten

Mit der Anweisung @P-KEYS kann man im Dialogbetrieb die programmierbaren Tasten (P-Tasten) der Tastatur mit einer vom EDT vorgegebenen Standardbelegung laden oder sich die vom EDT vorgegebene Standardbelegung anzeigen lassen.

Operation	**Operanden**	**F-Modus, L-Modus**
@P-KEYS	[**S**HOW]	

SHOW Es wird die vom EDT vorgegebene Standardbelegung angezeigt, mit der die P-Tasten durch die Anweisung @P-KEYS belegt werden.

Ist SHOW nicht angegeben, so werden die P-Tasten mit der vom EDT vorgegebenen Standardbelegung geladen.

Im Stapelbetrieb wird die Anweisung @P-KEYS ignoriert.

Beim Start des EDT werden bereits bestehende Belegungen der P-Tasten nicht verändert. Ebenso bleiben bei Beendigung des EDT die aktuell bestehenden Belegungen der P-Tasten erhalten.

Nach Ausführung der Anweisung @P-KEYS sind die P-Tasten mit Anweisungen belegt, die das folgende bewirken (Ausgabe der Anweisung @P-KEYS SHOW):

```
***  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
```

Sind die P-Tasten bereits mit der vom EDT vorgegebenen Standardbelegung geladen und wird mit der Anweisung @PAR SPLIT die Bildschirmaufteilung verändert oder mit der Anweisung @VDT an einer Datensichtstation 9763 das Format gewechselt, ist die Anweisung @P-KEYS erneut einzugeben, wenn die vom EDT vorgegebene Standardbelegung der P-Tasten weiterhin verwendet werden soll (dies ist notwendig, weil sich die in den P-Tasten gespeicherten Anweisungsfolgen auf absolute Positionen auf dem Bildschirm beziehen).

Die Eingabe der Anweisung @P-KEYS an einer Datensichtstation ohne P-Tasten wird mit der Fehlermeldung `EDT5366` abgewiesen.

9.87 @PAGE – Seitenvorschub

Die Anweisung @PAGE bewirkt einen Seitenvorschub auf SYSLST.

Operation	Operanden	F-Modus, L-Modus
@PAGE		

Zusätzlich zur Ausgabe des Vorschubsteuerzeichens für Seitenvorschub wird die Maximalzahl der auszudruckenden Zeilen pro Seite, die durch die Anweisung @LIST auf einen Wert zwischen 1 und 256 gesetzt werden kann, wieder auf den Standardwert 65 zurückgesetzt.

9.88 @PAR – Festlegung von Einstellungen des EDT

Mit der Anweisung @PAR werden Einstellungen des EDT festgelegt. Es handelt sich um Einstellungen für die Darstellung am Bildschirm, für die Steuerung des Verhaltens bei der Eingabe, um Standardwerte für Anweisungen sowie um die Vereinbarung von Sonderbedeutungen für bestimmte Zeichen.

Operation	**Operanden**	**F-Modus, L-Modus**
@PAR	[{ $0..$22 | **GL**OBAL }] [[,] parameter[,...]]	

Der Operand `parameter` besteht seinerseits aus einem Schlüsselwort und der jeweiligen Wertzuweisung. Die folgende Syntaxbeschreibung für die möglichen Werte von `parameter` ist thematisch geordnet. In der anschließenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

Parameter für die Darstellung und Ausgabe der Arbeitsfenster im F-Modus	
parameter	**E**DIT [-] **F**ULL [={**ON**] | **O**FF}] **E**DIT [[-] **L**ONG] [={**ON**] | **O**FF}] **H**EX [={**ON**] | **O**FF}] **I**NDEX [={**ON**] | **O**FF}] **INF**ORMATION [={**ON**] | **O**FF}] **OPT**IMIZE [={**ON**] | **O**FF}] **PROT**ECTION [={**ON**] | **O**FF}] **SC**ALE [={**ON**] | **O**FF}] **SPLIT** = {n $0..$22 | n (0..22) | **O**FF}

Steuerung des Verhaltens bei der Eingabe und Datenübernahme	
parameter	**D**ATA-REPLACEMENT [={**ON**] | **O**FF}] **INC**REMENT = {inc | ***S**TD} **LIM**IT = {col | ***S**TD} **LOW**ER [={**ON**] | **O**FF}] **REN**UMBER [={**ON**] | **O**FF}]

Voreinstellungen für andere Anweisungen	
parameter	**C**ODE = {name | **E**BCDIC | **I**SO} [**ELEMENT** [-]] **TYP**E = {eltype | ***S**TD} **L**IBRARY = {path | ***N**ONE} **SDF-NAME**-TYPE = {**I**NTERNAL | **E**XTERNAL | ***S**TD} **SDF-PROG**RAM = {progname | ***N**ONE}

Vereinbarung von Sonderbedeutungen für Zeichen	
parameter	**ESC**APE-CHARACTER = {strspec | ***N**ONE} **SEP**ARATOR = {strchar | ***N**ONE} **STRUC**TURE = {strchar | ***S**TD} **SUBST**ITUTION-CHARACTER = {strchar | ***N**ONE}

$0..$22 Die Arbeitsdatei, auf welche sich die @PAR-Anweisung bezieht.
Das Komma nach der Arbeitsdateiangabe darf nur fehlen, wenn kein weiterer Operand angegeben ist.

Die Operanden `OPTIMIZE` und `SUBSTITUTION-CHARACTER` beziehen sich immer auf alle Arbeitsdateien, der Operand `SPLIT` bezieht sich immer auf die aktuelle Arbeitsdatei. Die Angabe einer Arbeitsdatei wird in diesen Fällen ignoriert, bei `SPLIT` wird zusätzlich die Warnungsmeldung `EDT3127` ausgegeben.

Wenn weder `$0..$22` noch `GLOBAL` angegeben wird, bezieht sich die Anweisung auf die aktuelle Arbeitsdatei (falls nicht einer der Operanden `OPTIMIZE`, oder `SUBSTITUTION-CHARACTER` angegeben wurde).

GLOBAL Es wird festgelegt, dass sich die @PAR-Anweisung auf alle Arbeitsdateien beziehen soll. Das Komma nach `GLOBAL` darf nur fehlen, wenn kein weiterer Operand angegeben ist.

Der Operand `SPLIT` bezieht sich (auch bei Angabe von `GLOBAL`) immer auf die aktuelle Arbeitsdatei. Die Angabe von `GLOBAL` wird in diesem Fall ignoriert.

Wenn weder `$0..$22` noch `GLOBAL` angegeben wird, bezieht sich die Anweisung auf die aktuelle Arbeitsdatei (falls nicht einer der Operanden `OPTIMIZE`, oder `SUBSTITUTION-CHARACTER` angegeben wurde).

CODE= Legt den Standardwert des Operanden CODE fest für Anweisungen, die POSIX-Dateien lesen oder schreiben (siehe Abschnitt „POSIX-Dateien“ auf Seite 140 und den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist der Wert EDF041 voreingestellt.

name Name eines gültigen Zeichensatzes. Dieser wird für den CODE-Operanden in Anweisungen verwendet, die POSIX-Dateien lesen oder schreiben, wenn dort nicht explizit ein Zeichensatz spezifiziert wurde.

EBCDIC Dieser Operand wird nur noch aus Kompatibilitätsgründen unterstützt. Er stellt ein Synonym für EDF041 dar.

ISO Dieser Operand wird nur noch aus Kompatibilitätsgründen unterstützt. Er stellt ein Synonym für ISO88591 dar.

DATA-REPLACEMENT=

Legt fest ob bei Dateneingabe in der angegebenen Arbeitsdatei (im Datenfenster des F-Modus bzw. im L-Modus) die Ersatzdarstellung für Unicode-Zeichen (siehe Operand ESCAPE-CHARACTER) berücksichtigt werden soll (siehe Abschnitt „Unicode-Ersatzdarstellung“ auf Seite 53).

Nach dem Starten des EDT ist DATA-REPLACEMENT=OFF eingestellt.

ON Bei der Dateneingabe in der angegebenen Arbeitsdatei werden Ersatzdarstellungen der Form %uxxxx (% sei das mit @PAR ESCAPE-CHARACTER definierte Fluchtsymbol und die x seien Hexadezimalziffern) durch das entsprechende Unicode-Zeichen ersetzt. Dies gilt für die Eingabe von Zeilen im F-Modus wie im L-Modus aber z.B. nicht für das Einlesen von Sätzen aus Dateien oder Bibliothekselementen.

OFF Ersatzdarstellungen für Unicode-Zeichen werden nur innerhalb von Literalen in EDT-Anweisungen berücksichtigt.

EDIT-FULL= Bestimmt, ob im F-Modus das Datenfenster und die Kurzanweisungsspalte gleichzeitig auf überschreibbar gestellt werden sollen. Diese Einstellung ist nur wirksam, wenn die Zeilennummernanzeige eingeschaltet ist (@PAR INDEX=ON).

Nach dem Starten des EDT ist EDIT-FULL=OFF eingeschaltet.

ON Das Datenfenster und die Kurzanweisungsspalte sind gleichzeitig auf überschreibbar gestellt. Es ist möglich, in einer Zeile eine Kurzanweisung einzugeben und gleichzeitig Daten in dieser Zeile zu ändern. Dadurch können auch Kurzanweisungen für noch nicht angelegte Datenzeilen eingeben werden, etwa die O-Kurzanweisung.

Durch den Wechsel in den EDIT LONG-Modus (@PAR EDIT-LONG=ON) oder durch Ausschalten der Zeilennummernanzeige (@PAR INDEX=OFF) wird diese Einstellung unwirksam. Die Einstellung wird jedoch nicht zurückgenommen, sondern gemerkt und wieder wirksam, sobald die konkurrierenden Modi ausgeschaltet werden.

Solange der Schreibschutz (@PAR PROTECTION=ON) eingestellt ist, wird die Eingabe von @PAR EDIT-FULL=ON ignoriert.

OFF Es kann in einer Bildschirmzeile nur entweder die Kurzanweisungsspalte oder der Datenteil beschrieben werden.

EDIT-LONG= Bestimmt, ob im F-Modus Sätze, die länger sind als eine Bildschirmzeile, vollständig oder verkürzt im Arbeitsfenster dargestellt werden.

Nach dem Starten des EDT ist EDIT-LONG=OFF eingeschaltet.

ON Die Sätze werden, soweit möglich, vollständig im Arbeitsfenster dargestellt. Im EDIT-LONG-Modus wird weder ein mit @PAR SCALE=ON eingeschalteter Spaltenzähler noch eine mit @PAR INFORMATION=ON angeforderte Informationszeile dargestellt. Spaltenzähler und Informationszeile werden erst wieder eingeblendet, wenn der EDIT-LONG-Modus verlassen wird. Weitere Einzelheiten zum Bildschirm-Layout im EDIT-LONG-Modus findet man im Abschnitt „Das Arbeitsfenster“ auf Seite 107).

Das Einschalten des EDIT-LONG-Modus bewirkt implizit das Ausschalten der Zeilennummernanzeige (@PAR INDEX=OFF) und des Hexadezimal-Modus (@PAR HEX=OFF).

OFF Die Sätze werden, wenn erforderlich, gekürzt dargestellt. Die Länge des dargestellten Ausschnitts ist abhängig von der Datensichtstation und der Einstellung mit @VDT bzw. @PAR INDEX 72, 80, 124 oder 132 Zeichen pro Bildschirmzeile.

Wird der EDIT-LONG-Modus verlassen, bleibt die Zeilennummernanzeige ausgeschaltet. Der EDIT-LONG-Modus wird auch durch @PAR INDEX=ON und @PAR HEX=ON ausgeschaltet.

ELEMENT-TYPE=

Legt die Voreinstellung des Elementtyps für Bibliothekselemente fest.

Auf Elemente dieses Typs wird zugegriffen, wenn bei @COPY, @OPEN, @DELETE, @WRITE, @INPUT kein Elementtyp angegeben wurde (siehe auch den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist der Wert S voreingestellt.

eltype Zulässige Typangaben sind S, M, P, J, D, X, R, C, H, L, U, F und freie Typnamen mit entsprechendem Basistyp. Der Operand eltype kann auch als Zeichenfolgevariable in der Form ELEMENT-TYPE=.svarex angegeben werden. Der Punkt muss angegeben werden, um den Namen der Zeichenfolgevariablen von einem mit # beginnenden freien Typnamen unterscheiden zu können.

Nicht-textartige Typen (R, C, H, L, U, F) sind in den Anweisungen zum Lesen und Schreiben nicht zulässig. Ihre Angabe in @PAR ELEMENT-TYPE ist daher nur sinnvoll, wenn die Voreinstellung ausschließlich für die @DELETE-Anweisung benutzt wird. Die zulässigen Elementtypen und deren Bedeutung sind im Abschnitt „Dateibearbeitung“ auf Seite 137 beschrieben.

*STD Stellt die Voreinstellung (d.h. den Wert S) wieder her.

ESCAPE-CHARACTER=

Legt das Zeichen fest, mit dem die Ersatzdarstellung für Unicode-Zeichen eingeleitet wird (siehe auch Abschnitt „Unicode-Ersatzdarstellung“ auf Seite 53 und den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist ESCAPE-CHARACTER=*NONE eingestellt.

strspec Sonderzeichen, das als Fluchtsymbol die Ersatzdarstellung für Unicode-Zeichen einleitet. Wenn etwa mit @PAR ESCAPE-CHARACTER='%' das Fluchtsymbol % vereinbart wurde, wird eine Zeichenfolge der Form %Uxxxx mit vier Hexadezimalziffern x als Ersatzdarstellung für das Unicode-Zeichen mit dem Code xxxx interpretiert. Über die Einstellung @PAR DATA-REPLACEMENT wird gesteuert, ob die Ersatzdarstellung nur für Literale in Anweisungen berücksichtigt werden soll oder auch für die Dateneingabe im Arbeitsfenster bzw. im L-Modus. Sowohl bei Eingabe einer Anweisung wie bei Dateneingabe wird immer das Fluchtsymbol für die aktuelle Arbeitsdatei zugrunde gelegt.

*NONE Es wird kein Fluchtsymbol definiert. Der EDT geht davon aus, dass es keine Ersatzdarstellung in den Eingaben gibt.

HEX= Dient zum Ein- und Ausschalten des Hexadezimalmodus (siehe Abschnitt „Hexadezimalmodus“ auf Seite 125). Die Zeilen einer Arbeitsdatei, für die der Hexadezimalmodus eingeschaltet ist, werden im F-Modus in abdruckbarer und in hexadezimaler Form am Bildschirm dargestellt. Einzelheiten des Layouts sind im Abschnitt Hexadezimalmodus beschrieben.

Nach dem Start des EDT ist HEX=OFF eingestellt.

ON — Der Hexadezimalmodus wird für die angegebene Arbeitsdatei eingeschaltet. Der `EDIT-LONG`-Modus wird dabei implizit abgeschaltet.
Wenn die gleiche Arbeitsdatei gleichzeitig in mehreren Arbeitsfenstern am Bildschirm angezeigt wird, gilt in beiden Datenfenstern die gleiche Einstellung.

Ist das Datenfenster bei geteiltem Bildschirm so klein, dass nicht wenigstens eine Datenzeile mit ihren Hexzeilen angezeigt werden kann, wird die Meldung `EDT2404` ausgegeben, der Hexadezimal-Modus aber eingeschaltet. Der Anwender kann dann sein Datenfenster so vergrößern, dass die Hexzeilen auch angezeigt werden können.

OFF — Der Hexadezimalmodus wird ausgeschaltet.

INCREMENT= — Legt die Schrittweite fest, die bei der Vergabe von Zeilennummern verwendet werden soll. Bei welchen Anweisungen und Aktionen diese Einstellung wirksam wird, ist im Abschnitt „Zeilennummernvergabe“ auf Seite 37 beschrieben.

Nach dem Starten des EDT ist der Wert `1.0` voreingestellt.

inc — Schrittweite, die bei der Vergabe von Zeilennummern verwendet werden soll. Bei @PAR INCREMENT mit einer Schrittweite `<0.01` ist zu beachten, dass im F-Modus Zeilennummern von eingelesenen, kopierten oder eingefügten Zeilen nicht vollständig dargestellt werden (6-stellige Zeilennummernanzeige). Werden diese unvollständig ausgegebenen Zeilennummern in Anweisungen verwendet, kann dies zu unerwünschten Ergebnissen führen.

*STD — Der beim Starten des EDT voreingestellte Wert wird wieder eingestellt.

INDEX= — Aus- und Einschalten der Zeilennummernanzeige im F-Modus. Das Bildschirm-Layout bei ein- bzw. ausgeschalteter Zeilennummernanzeige ist im Abschnitt „F-Modus“ auf Seite 105 beschrieben.

Die Einstellung der Zeilennummernanzeige wird getrennt für das obere und untere (mögliche) Datenfenster jeder Arbeitsdatei gespeichert. Wird nur ein Datenfenster am Bildschirm angezeigt, so überschreiben dessen Einstellungen die des oberen Datenfensters der entsprechenden Arbeitsdatei.

Wenn die Anweisung @PAR GLOBAL,INDEX=... eingegeben wurde, wirkt sie auf alle (möglichen) Bildschirmfenster aller Arbeitsdateien. Bei Angabe einer einzelnen Arbeitsdatei (auch bei impliziter Angabe der aktuellen Arbeitsdatei) müssen mehrere Fälle unterschieden werden:

Bei *geteiltem* Bildschirm (siehe @PAR SPLIT) wirkt die Anweisung @PAR INDEX nur auf das jeweils sichtbare Bildschirmfenster der angegebenen Arbeitsdatei. Wenn beide Bildschirmfenster der angegebenen Arbeitsdatei sichtbar sind, wirkt die Anweisung auf das Eingabefenster.

Wenn kein Bildschirmfenster der angegebenen Arbeitsdatei sichtbar ist, wirkt die Anweisung auf das obere Bildschirmfenster. Dies gilt sinngemäß auch für Eingaben im L-Modus, wobei *Sichtbarkeit* und *Eingabefenster* sich auf den Zustand nach einem möglichen Wechsel in den F-Modus (mit der Anweisung @EDIT FULL) beziehen. Die entsprechende Information kann im L-Modus auch durch @STATUS=PAR(..) angezeigt werden.

Bei *ungeteiltem* Bildschirm wirkt die Anweisung auf beide (mögliche) Bildschirmfenster der angegebenen Arbeitsdatei unabhängig davon ob diese sichtbar sind oder nicht. Dies gilt ebenfalls wieder sinngemäß im L-Modus, d.h. wenn nach Eingabe von @EDIT FULL ein ungeteilter Bildschirm angezeigt würde.

Nach dem Starten des EDT ist `INDEX=ON` voreingestellt.

ON — Die Zeilennummernanzeige wird eingeschaltet. Durch @PAR INDEX=ON wird auch der `EDIT-LONG`-Modus ausgeschaltet.

OFF — Die Zeilennummernanzeige wird ausgeschaltet. Der `EDIT-FULL`-Modus wird beim Abschalten der Zeilennummeranzeige unwirksam, aber nicht abgeschaltet.

INFORMATION=

Legt fest, ob im F-Modus für die angegebene Arbeitsdatei eine Informationszeile im Datenfenster ausgegeben wird.

Die Informationszeile enthält, neben der Nummer der Arbeitsdatei und dem Namen des für diese Arbeitsdatei eingestellten Zeichensatzes, wenn vorhanden, den Namen eines lokalen @FILE-Eintrags, bzw. den Namen der geöffneten Datei oder des geöffneten Bibliothekselements (siehe Abschnitt „Dateibearbeitung“ auf Seite 137). Falls die Arbeitsdatei 9 die Ausgabe einer der Anweisungen @FSTAT LONG, @STAJV LONG oder @SHOW (ohne Zielangabe) enthält und der Inhalt nicht geändert wurde, enthält die Informationszeile der Arbeitsdatei 9 Spaltenüberschriften für die jeweilige Ausgabe. Trifft keiner der genannten Fälle zu, enthält die Informationszeile nur die Nummer der Arbeitsdatei und den Zeichensatz.

Nach dem Starten des EDT ist `INFORMATION=OFF` voreingestellt.

ON — Die Informationszeile erscheint als erste Zeile im Arbeitsfenster noch vor einem eventuell eingeschaltetem Zeilenlineal. Die Bildschirmdarstellung wird nicht verändert, wenn das Arbeitsfenster zu klein ist, um neben der Informationszeile noch mindestens eine Datenzeile anzuzeigen. Die Einstellung wird dann erst wirksam, wenn das Arbeitsfenster entsprechend vergrößert wird.

OFF — Es wird keine Informationszeile ausgegeben.

LIBRARY= Legt die Voreinstellung eines Bibliotheksnamens für die Anweisungen @COPY, @OPEN, @WRITE, @INPUT und @SHOW fest. Diese Voreinstellung wird verwendet, wenn bei den genannten Anweisungen kein Bibliotheksname angegeben wird (siehe auch den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist `LIBRARY=*NONE` voreingestellt.

path Name der Bibliothek.

*NONE Die Voreinstellung des Bibliotheksnamens wird zurückgesetzt. Bei den oben genannten Anweisungen muss die Bibliothek dann jeweils explizit angegeben werden.

LIMIT= Definiert die maximale Satzlänge im F-Modus-Datenfenster. Wird ein Satz eingegeben, dessen Länge den angegebenen Wert überschreitet, wird dieser Satz abgeschnitten und die Meldung `EDT2267` ausgegeben.

Indirekte Satzänderungen durch Anweisungen oder Dateneingaben im L-Modus sind von dieser Prüfung nicht betroffen. Insbesondere hat dieser Parameter nichts mit der durch @CHECK oder @TABS eingestellten Satzlängenprüfung des L-Modus zu tun.

Nach dem Starten des EDT ist `LIMIT=32768` voreingestellt.

col Die maximale erlaubte Satzlänge (`1..32768`) bei Eingaben im F-Modus-Datenfenster.

*STD Der beim Starten des EDT voreingestellte Wert wird wieder eingestellt.

LOWER= Legt fest, ob der EDT an der Datensichtstation eingegebene Kleinbuchstaben in Großbuchstaben umsetzt oder nicht (siehe auch den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist der Wert `LOWER=ON` voreingestellt.

ON Der EDT unterscheidet zwischen Groß- und Kleinbuchstaben. Text und Zeichenfolgen werden verarbeitet, wie sie eingegeben wurden.

OFF Der EDT setzt an der Datensichtstation eingegebene Kleinbuchstaben in Großbuchstaben um. Im F-Modus werden in der Arbeitsdatei enthaltene Kleinbuchstaben bei der Ausgabe im Arbeitsfenster als Schmierzeichen dargestellt.

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.

Nach dem Starten des EDT ist `OPTIMIZE=ON` voreingestellt.

ON Es werden bei jedem Dialogschritt nur geänderte Zeileninhalte ausgegeben.

OFF Der gesamte Arbeitsfensterinhalt wird bei jedem Dialogschritt ausgegeben.

PROTECTION=

Schaltet den Schreibschutz auf Satzebene ein bzw. aus. Dieser Operand ist nur im Zusammenhang mit der Anwendung der EDT-Unterprogramm-Schnittstelle sinnvoll. Dann können vom Anwenderprogramm aus Sätze durch eine entsprechende Satzmarkierung schreibgeschützt oder überschreibbar dargestellt werden (siehe Handbuch Unterprogramm-Schnittstellen [1]).

Nach dem Starten des EDT ist `PROTECTION=OFF` voreingestellt.

ON Es werden entsprechend gekennzeichnete Sätze im F-Modus Dialog schreibgeschützt dargestellt bzw. automatisch auf überschreibbar gestellt.

Wurde vorher mit @PAR EDIT-FULL=ON die gleichzeitige Überschreibbarkeit von Kurzanweisungsspalte und Datenzeilen eingestellt, wird diese Einstellung zurückgesetzt.

OFF Die vom Anwenderprogramm angegebene Voreinstellung (Schreibschutz oder Überschreibbarkeit) ist nicht wirksam.

RENUMBER= Steuert die automatische Umnummerierung der Zeilennummern. Bei welchen Anweisungen und Aktionen diese Einstellung wirksam wird, ist im Abschnitt „Zeilennummernvergabe“ auf Seite 37 beschrieben.

Nach dem Starten des EDT ist `RENUMBER=ON` voreingestellt.

ON Die Zeilennummern einer Arbeitsdatei werden bei Bedarf umnummeriert. 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 vollständig auszuführen.

OFF Die Zeilennummern einer Datei werden vom EDT nicht verändert. Es erfolgt eine Meldung des EDT, wenn die Anweisung nicht ausgeführt werden kann, weil die Schrittweite nicht klein genug ist, um alle Sätze aufnehmen zu können.

SCALE= Schaltet im F-Modus die Anzeige eines Spaltenzählers (Zeilenlineal) im Datenfenster ein oder aus.

Im EDIT-LONG-Modus wird der Spaltenzähler grundsätzlich nicht ausgegeben. Bei Ausgabe im Hexadezimalmodus wird statt des einen Spaltenzählers für jede Zeile ein Spaltenzähler ausgegeben. Die mit @PAR SCALE getroffene Einstellung wird jedoch wirksam, wenn der entsprechende Modus verlassen wird.

Nach dem Starten des EDT ist SCALE=OFF voreingestellt.

ON Der Spaltenzähler erscheint als erste Zeile nach einer eventuell vorhandenen Informationszeile (siehe @PAR INFORMATION) und gibt für die gewünschte Arbeitsdatei die aktuellen Spaltennummern des Arbeitsfensters an (z.B. nach horizontalem Verschieben des Arbeitsfensters).

Falls ein Tabulator definiert ist (siehe Anweisung @TABS), wird eine weitere Bildschirmzeile ausgegeben, in der die aktuellen Positionen des Tabulators mit I angezeigt werden. In dieser Zeile wird auch das Tabulatorzeichen selbst in der Kurzanweisungsspalte abgebildet.

Falls das Arbeitsfenster zu klein ist, um neben dem Spaltenzähler noch mindestens eine Datenzeile anzuzeigen, wird der Spaltenzähler nicht angezeigt. Beim Vergrößern des Arbeitsfensters wird zunächst (wenn vorhanden) die Informationszeile, dann der Spaltenzähler und schließlich (wenn vorhanden) die Tabulatorzeile wieder eingeblendet.

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

SDF-NAME-TYPE=

Legt den Namenstyp des mit @PAR SDF-PROGRAM vordefinierten Programmnamens sowie die Voreinstellung des Namenstyps in der Anweisung @SDFTEST fest (siehe auch den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist SDF-NAME-TYPE=INTERNAL voreingestellt.

INTERNAL

Der Programmname ist der maximal 8-stellige interne Name. Der interne Name kann mit SDF-A ermittelt werden, falls er nicht dem Namen des Programms entspricht.

EXTERNAL

Der Programmname ist der maximal 30-stellige externe Name (z.B. LMS, SDF-A oder HSMS).

*STD Der beim Starten des EDT voreingestellte Wert wird wieder eingestellt.

SDF-PROGRAM=

Es wird für die Anweisung @SDFTEST und die Kurzanweisung `T` der Name eines Programms vordefiniert. Beim Abarbeiten dieser Anweisung bzw. Kurzanweisung werden Datenzeilen, die mit `//` beginnen, dann mit der Syntaxdatei dieses Programmes analysiert (siehe auch den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist `SDF-PROGRAM=*NONE` voreingestellt.

progname Name eines Programms. Es kann sowohl der maximal 8-stellige interne Name wie auch der maximal 30-stellige externe Name angegeben werden (siehe @PAR SDF-NAME-TYPE)

*NONE Mit `*NONE` wird die Definition eines Programmes zurückgenommen.

SEPARATOR= Legt das Satztrennzeichen zum Auftrennen eines Datensatzes bzw. die Voreinstellung für die Anweisung @SEPARATE fest (siehe Abschnitt „Anweisung im Datenfenster - Auftrennen eines Datensatzes“ auf Seite 117 bzw. @SEPARATE-Anweisung und den Hinweis am Ende des Abschnitts).

Nach dem Starten des EDT ist `SEPARATOR=*NONE` voreingestellt.

strchar Das Satztrennzeichen kann als alphanumerisches Zeichen oder Sonderzeichen in Hochkommas bzw. in der Form `U'unicode'` in seiner `UTF16`-Codierung angegeben werden.

Für Satztrennzeichen und Tabulatorzeichen sind unterschiedliche Zeichen zu wählen.

*NONE Aufheben einer Satztrennzeichen-Definition. Aus Kompatibilitätsgründen ist hier auch das Schlüsselwort `OFF` zugelassen.

SPLIT= Mit `SPLIT` kann im F-Modus ein zweites Arbeitsfenster am Bildschirm ausgegeben werden. Die Arbeitsdatei, in der diese Anweisung eingegeben wurde (aktuelle Arbeitsdatei), wird im oberen Arbeitsfenster gezeigt. Die mit `$0..$22` angegebene Arbeitsdatei wird im unteren Arbeitsfenster gezeigt. Nach dem Starten des EDT ist `SPLIT=OFF` voreingestellt.

n $0..$22 | n (0..22)

Die angegebene Arbeitsdatei wird in der Länge `n` im unteren Arbeitsfenster angezeigt. Für die Länge `n` ist mindestens `2` anzugeben und höchstens zwei weniger als die von der Datensichtstation je nach Bildschirmformat maximal erlaubte Zeilenzahl (siehe @VDT-Anweisung).

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 Blendet die zweite Arbeitsdatei aus und setzt für das verbleibende Bildschirmfenster die Länge auf den von der Datensichtstation zugelassenen Maximalwert. Es bleibt das Bildschirmfenster 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 abgewiesen. Die Anzeige zweier Arbeitsfenster wird implizit auch durch die Anweisung @VDT ausgeschaltet.

STRUCTURE=

Legt das Struktursymbol fest, das beim Positionieren nach der Strukturtiefe (siehe Kurzanweisungen + und -) ausgewertet wird.

Nach dem Starten des EDT ist der Wert `'@'` voreingestellt.

strchar Zeichen, das als Struktursymbol zu verwenden ist. Dieses Symbol kennzeichnet Sätze, die beim Positionieren nach der Strukturtiefe berücksichtigt werden sollen. Das Zeichen kann als alphanumerisches Zeichen oder Sonderzeichen in Hochkommas bzw. in der Form `U'unicode'` in seiner `UTF16`-Codierung angegeben werden.

Ist ein Struktursymbol ungleich Leerzeichen definiert, wird beim Blättern nach der Strukturtiefe nur zu solchen Zeilen positioniert, die dieses Struktursymbol enthalten. Wird für das Struktursymbol ein Leerzeichen angegeben, so kann zu allen Zeilen positioniert werden.

*STD Der beim Starten des EDT voreingestellte Wert wird wieder eingestellt.

SUBSTITUTION-CHARACTER=

Legt das Ersatzzeichen fest, das bei Umcodierungen anstelle eines im Ziel-Zeichensatz ungültigen Zeichens eingesetzt wird. Die Einstellung gilt für alle Arbeitsdateien und Zeichenfolgevariablen, unabhängig davon, welche Arbeitsdatei in der @PAR-Anweisung spezifiziert wurde. Beim Umcodieren in den Kommunikations-Zeichensatz (siehe Abschnitt „Zeichensätze“ auf Seite 48) gilt diese Vereinbarung nicht. In diesem Fall wird immer das für die Datensichtstation gerätespezifische Schmierzeichen als Ersatzzeichen verwendet.

Nach dem Starten des EDT ist `SUBSTITUTION-CHARACTER=*NONE` voreingestellt.

strchar
: Zeichen, das bei Umcodierungen anstelle eines im Ziel-Zeichensatz ungültigen Zeichens eingesetzt wird. Das Zeichen kann als alphanumerisches Zeichen oder Sonderzeichen in Hochkommas bzw. in der Form `U'unicode'` in seiner `UTF16`-Codierung angegeben werden.

  Da nicht für jeden möglichen Ziel-Zeichensatz ein eigenes Ersatzzeichen vereinbart werden kann, sind nur solche Zeichen sinnvoll, die in allen verwendeten Zeichensätzen vorhanden sind. Es bieten sich etwa einige Sonderzeichen des EBCDIC-DF03-Kernels (`&-/:.,<*%_+>?`) oder das Zeichen `X'00'` an (siehe Abschnitt „Zeichensätze“ auf Seite 48). Dies wird bei der Eingabe jedoch nicht geprüft. Wenn eine Umcodierung mit einem ungültigen Ersatzzeichen erfolgen soll, verhält sich der EDT als ob für das Ersatzzeichen `*NONE` definiert worden wäre.

*NONE
: Bei Umcodierungen wird für im Ziel-Zeichensatz ungültige Zeichen keine Ersatzdarstellung verwendet. Stattdessen werden Umcodierungen, die zu solchen Zeichen führen würden mit einer Fehlermeldung abgewiesen. Eine Ausnahme bilden die Ausgaben nach `SYSOUT` und `SYSLST`. Bei diesen werden in diesem Fall Leerzeichen eingesetzt.

Werden keine Operanden angegeben, werden alle Voreinstellungen für die aktuelle Arbeitsdatei (außer `SPLIT` und `OPTIMIZE`) auf die Werte zurückgesetzt, die nach dem Starten des EDT eingestellt sind. Der Wert für `SUBSTITUTION-CHARACTER` wird immer für alle Arbeitsdateien auf diesen Standardwert zurückgesetzt.

Werden nur die Operanden `GLOBAL` bzw. `$1..$22` angegeben, werden alle Voreinstellungen für alle Arbeitsdateien bzw. alle Voreinstellungen für die angegebene Arbeitsdatei (außer `SPLIT` und `OPTIMIZE`) auf diese Standardwerte zurückgesetzt. Der Wert für `SUBSTITUTION-CHARACTER` wird immer für alle Arbeitsdateien auf diesen Standardwert zurückgesetzt.

Die Einstellungsoperanden werden in der Reihenfolge ihrer Angabe gesetzt. Das ist von Bedeutung für Parameter, die sich gegenseitig beeinflussen bzw. für den Abbruch der Anweisung bei Laufzeitfehlern (können bei `SPLIT` und `CODE` auftreten).

Mit der @PAR-Anweisung können auch im L-Modus Einstellungen geändert werden, die die Darstellung im F-Modus betreffen. Diese sind zwar sofort wirksam, werden jedoch naturgemäß erst bei Wechsel in den F-Modus sichtbar.

Mit Ausnahme von `INDEX` gelten alle Einstellungen für eine Arbeitsdatei unabhängig davon, ob eines, beide oder keines der möglichen Bildschirmfenster für diese Arbeitsdatei angezeigt wird. Wenn also die gleiche Arbeitsdatei mehrfach am Bildschirm angezeigt wird, wirkt sich die Einstellung (bis auf `INDEX`) in beiden Bildschirmfenstern aus.

Die aktuellen Werte für die mit @PAR getroffenen Einstellungen können mit der Anweisung @STATUS ausgegeben werden.

Hinweis

Die mit @PAR getroffenen Einstellungen (bis auf OPTIMIZE, SPLIT und SUBSTITUTION-CHARACTER) können für jede Arbeitsdatei einen anderen Wert haben.
Dies gilt auch für die mit CODE, ELEMENT-TYPE, ESCAPE-CHARACTER, LIBRARY, LOWER, SDF-NAME-TYPE, SDF-PROGRAM und SEPARATOR vorgenommenen Einstellungen, mit denen Standardwerte oder Verhalten einer anderen Anweisung (nutzende Anweisung) beeinflusst wird. Für diese Einstellungen gilt die Regel, dass der Wert derjenigen Arbeitsdatei verwendet wird, die zum Zeitpunkt der nutzenden Anweisung als aktuelle Arbeitsdatei eingestellt ist.
Wenn für einen Wert Zeichenfolgevariable angegeben werden dürfen, findet die Ersetzung durch den Inhalt der Zeichenfolgevariablen bereits bei der Verarbeitung der @PAR-Anweisung statt. Spätere Änderungen der Zeichenfolgevariablen haben also keinen Einfluss mehr auf die Einstellung.

9.89 @PARAMS – Definieren von Prozedur-Parametern

Mit der Anweisung @PARAMS können symbolischen Parameter definiert werden, die innerhalb einer @DO-Prozedur benutzt werden. Man unterscheidet Stellungsparameter und Schlüsselwortparameter.

Die Parameter können als Zeichenvariablen betrachtet werden, die vor dem Ausführen der EDT-Prozedur durch den jeweiligen Wert ersetzt werden, der beim Aufruf der Prozedur in der @DO-Anweisung angegeben wurde oder der bei Schlüsselwortparametern als Standardwert in der @PARAMS-Anweisung selbst definiert ist.

Operation	**Operanden**	**@PROC**
@PARAMS	[formal[,...]] [[,] {formal=param} [,...]]	

formal — Formaler (symbolischer) Parameter. Ein Parameter beginnt mit dem Zeichen `&`. Ihm folgt ein Buchstabe (`A..Z, a..z`), 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. Die Angaben `&A` und `&a` bezeichnen also zwei verschiedene Parameter. Die in einer Arbeitsdatei verwendeten Parameternamen gelten nur innerhalb dieser Arbeitsdatei. Bei Angabe eines unzulässigen formalen Parameters wird die Ausführung der Prozedur mit der Meldung `EDT4924` abgelehnt.

param — Standardwert des Schlüsselwortparameters. Der angegebene Wert wird in der Prozedur eingesetzt, wenn beim Aufruf der Prozedur mit der @DO-Anweisung für diesen Schlüsselwortparameter kein anderer Wert spezifiziert wird. Schlüsselwortparameter müssen nach allen Stellungsparametern vereinbart werden.

Die Anweisung @PARAMS muss in der ersten Zeile einer @DO-Prozedur stehen. Eine @PARAMS-Anweisung in Folgezeilen führt zur Ausgabe der Meldung `EDT5479` und wird ignoriert. Fehler in der @PARAMS-Anweisung verhindern die Ausführung der Prozedur und werden von @DO gemeldet. Die @PARAMS-Anweisung ist nicht eigentlicher Bestandteil der Prozedur, kann also z.B. auch nicht angesprungen werden, wird bei der Ausführung mit @DO...PRINT nicht protokolliert und wird bei Prozeduren mit äußeren Schleifen auch nicht mehrfach ausgeführt.

Es muss mindestens ein formaler Parameter angegeben werden, sonst wird die Ausführung der Prozedur mit der Meldung `EDT4918` abgelehnt. Die gleiche Meldung erfolgt, wenn ein Parameter nicht mit `&` beginnt oder nach einem Komma kein weiterer Parameter folgt.

Werden Stellungsparameter nach Schlüsselwortparametern angegeben, erfolgt die Meldung EDT4948. Wird ein Schlüsselwortparameter mehrfach angegeben, wird die Fehlermeldung EDT3910 ausgegeben. Für sonstige Fehler wird die Meldung EDT5478 ausgegeben.

Die Anzahl der in @PARAMS möglichen formalen Parameter ist nur durch die maximale Länge einer EDT-Anweisung beschränkt.

Der EDT ignoriert in der @PARAMS-Anweisung Leerzeichen, die unmittelbar vor bzw. nach einem Parameternamen auftreten. Die im Kompatibilitäts-Modus mögliche Angabe von Leerzeichen innerhalb des Parameternamens wird nicht mehr unterstützt.

In Schlüsselwortparametern steht hinter dem Parameternamen (Schlüsselwort) ein Gleichheitszeichen. Dem Gleichheitszeichen folgt der Standardwert dieses Parameters. Schlüsselwortparameter erhalten den aktuellen Wert ebenfalls aus der Parameterliste im @DO-Aufruf. Falls er dort fehlt, wird der vordefinierte Standardwert aus @PARAMS eingesetzt. Der zugewiesene Standardwert ergibt sich aus allen angegebenen Zeichen, einschließlich der Leerzeichen. Soll einem Schlüsselwortparameter kein Standardwert zugewiesen werden, ist unmittelbar nach dem Gleichheitszeichen ein Komma anzugeben oder @PARAMS zu beenden. Alternativ kann auch die leere Zeichenfolge '' explizit als Standardwert angegeben werden.

Der Standardwert eines Schlüsselwortparameters kann in Hochkommas eingeschlossen werden, diese werden nicht übernommen, wenn sie als erstes und letztes Zeichen des Parameterwerts auftreten und dazwischen entweder keine oder nur doppelte Hochkommas auftreten. In allen anderen Fällen bleiben alle angegebenen Hochkommas Bestandteil des Parameters (siehe Beispiele). Die Zeichen Komma und schließende Klammer können nur Bestandteil eines Parameters sein, wenn sie innerhalb einer durch Hochkomma eingeschlossenen Teilzeichenfolge des Parameterwertes vorkommen. Hochkommas müssen immer paarig in einem Parameterwert vorkommen. Ein einzelnes Hochkomma kann nicht in einem Parameterwert übergeben werden. Wurde zuvor mit @QUOTE einem anderen Zeichen die Funktion des Hochkommas zugeordnet, gilt dies nicht für die den Standardwert einschließenden Hochkommas.

Wird für einen Stellungsparameter beim Aufruf der Prozedur mit @DO kein Wert angegeben, so erhält er als Wert die leere Zeichenfolge. Wird für einen Schlüsselwortparameter beim Aufruf der Prozedur mit @DO kein Wert angegeben, so erhält er als Wert den in der @PARAMS-Anweisung definierten Standardwert.

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 (siehe @DO-Anweisung).

Kommt in der Prozedurdatei eine Zeichenfolge vor, die mit & beginnt und syntaktisch einen formalen Parameter bezeichnen könnte, der aber in der @PARAMS-Anweisung nicht deklariert ist oder fehlt die @PARAMS-Anweisung ganz, findet für diese Zeichenfolgen keine Ersetzung statt.

Die Parameter können innerhalb der Prozedur an beliebiger Stelle verwendet und mit anderen Zeichenfolgen und Parametern verkettet werden.

Folgt innerhalb der Prozedur auf den formalen Parameter unmittelbar ein Buchstabe, eine Ziffer oder ein Punkt, muss der formale Parameter von diesem Zeichen durch einen Punkt abgetrennt werden (außer für Parameternamen mit maximaler Länge 7).

Ein einzelner Punkt nach einem formalen Parameter wird als Trennzeichen interpretiert und bei der Ersetzung des formalen Parameters durch seinen Wert nicht übernommen (siehe Beispiele). Dies gilt unabhängig von dem Zeichen, das auf den Punkt folgt.

Soll eine Zeichenfolge, die mit & beginnt und einem bei @PARAMS angeführten formalen Parameter entspricht, in der Prozedurdatei nicht durch den aktuellen Parameterwert ersetzt werden, ist das Zeichen & doppelt anzugeben. Beim Ablauf der Prozedur wird eines der beiden & entfernt. Auch sonstige doppelt auftretende & werden durch nur ein & ersetzt, wenn die Prozedur eine @PARAMS-Anweisung enthält.

Wird in der Prozedur ein formaler Parameter durch den aktuellen Parameterwert ersetzt, kann dies zur Folge haben, dass die Zeile die maximal erlaubte Zeilenlänge überschreitet. Das führt zur Fehlermeldung EDT1938 bei der Ausführung der Prozedur. Die betroffene Zeile wird dann nicht ausgeführt.

Zur Behandlung von Zeichensätzen bei der Parameterbearbeitung sei auf den Abschnitt „EDT-Prozeduren“ auf Seite 66 und die Anweisung @DO verwiesen.

Eine indirekte Operandenangabe ist bei dieser Anweisung nicht erlaubt.

Beispiel 1

Angabe Parameterwert	**substituierte Zeichenfolge**
&F=A,...	A
&F=,...	leere Zeichenfolge
&F='',...	leere Zeichenfolge
&F=␣ABC␣,...	␣ABC␣
&F='X',...	X
&F='X''X',...	X''X
&F='X'Y'X',...	'X'Y'X'
&F='AB'C,...	'AB'C
&F=␣'ABC',...	␣'ABC'
&F=␣,...	␣
&F=',)',...	,)
&F=A','B,...	A','B

Beispiel 2

Zeile in Prozedurdatei	Parametereingabe	Erzeugte Zeile
&PARAM(BC)	&PARAM=A	A(BC)
&PARAM.(BC)	&PARAM=A	A(BC)
&PARAM..(BC)	&PARAM=A	A.(BC)
&PARAM..BC	&PARAM=A	A.BC
&PARAM.2BC	&PARAM=A	A2BC
&PARAMBC	&PARAM=A	&PARAMBC
&PARAM,.2B	&PARAM=A	A,.2B
BC&PARAM	&PARAM=A	BCA
BC,&PARAM	&PARAM=A	BC,A
B2&PARAM	&PARAM=A	B2A
&PARAM.&PARAM	&PARAM=A	AA
&PARAM&PARAM	&PARAM=A	AA
&PARAM..&PARAM	&PARAM=A	A.A
&PARAM&&PARAM	&PARAM=A	A&PARAM
@ON &F'&SEARCH'	&SEARCH=A''B	@ON &F'A''B'
@SET #S1=&STR	&STR=␣'TEXT'	@SET #S1=␣'TEXT'
&DATA	&DATA='A'B	'A'B
@P RANGE	&RANGE='3,7'	@P 3,7
&CMD #S1	&CMD=@PRINT	@PRINT #S1

Beispiel 3

```
   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 ------------------------------------------- (1)
   2.     @ @NOTE Gib &ZEILEN aus
   3.     @ @PRINT &ZEILEN -------------------------------------------- (2)
   4.     @END
   7.     @DO 1(2-4) -------------------------------------------------- (3)
   2.0000 AUSGEGEBEN
   3.0000 WERDEN SOLL,
   4.0000 WIRD ERST
   7.     @DO 1(#S3),PRINT
   7.     @NOTE Gib #S3 aus ------------------------------------------- (4)
   7.     @PRINT #S3
     #S03 *** SO IST ES ***
   7.     @DO 1(2,4N) ------------------------------------------------- (5)
 %  EDT4963 TOO MANY OPERANDS
   7.     @DO 1('2,4N') ----------------------------------------------- (6)
   2.0000 AUSGEGEBEN
WIRD ERST
   7.
```

(1) Innerhalb der Arbeitsdatei 1 wird in der ersten Zeile der Stellungsparameter `&ZEILEN` vereinbart.

(2) Dieser Parameter taucht innerhalb von @PRINT auf. Welche Zeilen nun ausgegeben werden sollen, hängt von dem in der @DO-Anweisung genannten Parameterwert ab.

(3) Die Arbeitsdatei 1 wird ausgeführt. Vor der Ausführung der einzelnen Anweisung wird jedoch dem Parameter `&ZEILEN` der Wertbereich 2-4 zugeordnet.

(4) 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.

(5) 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.

(6) 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 4

```
   1.     @PROC 2
   1.     @ @PARAMS &STRVAR1,&STRVAR2,&INHALT1=**** ---------------------- (1)
   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) ------------------------------------------------ (2)
     #S00 ****
     #S01 ****
     #S02 #S0
     #S03 ****
     #S04 VON #S0 BIS #S1
   1.     @DO 2(#S15,#S13,INHALT1=AUWEIA) ------------------------------- (3)
     #S15 AUWEIA
     #S13 AUWEIA
     #S02 #S15
     #S03 AUWEIA
     #S04 VON #S15 BIS #S13
1.
```

(1) 2 Stellungs- und 1 Schlüsselwortparameter werden für die Arbeitsdatei 2 definiert.

(2) 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.

(3) Wird für einen Schlüsselwortparameter ein Parameterwert bei @DO angegeben, wird dadurch der Standardwert ersetzt.

Beispiel 5

```
1.     @PROC 3
1.     @ @PARAMS &A,&B,&C,&X=111,&Y=222,&Z=333 ----------------------- (1)
2.     @ @CREATE #S10: '&A','&B','&C','&X','&Y','&Z'
3.     @ @PRINT #S10
4.     @END
1.     @DO 3 (AAAAA,BB,CCCCCCC) ---------------------------------------- (2)
  #S10 AAAAABBCCCCCCC111222333
1.     @DO 3 (AA,BBBB,C,Y=****,X=########) ---------------------------- (3)
  #S10 AABBBBC########****333
1.
```

(1) Innerhalb der Arbeitsdatei 3 werden 3 Stellungs- und 3 Schlüsselwortparameter definiert.

(2) Die Arbeitsdatei 3 wird ausgeführt. Da jedoch kein Wert für einen Schlüsselwortparameter angegeben ist, werden hierfür die Standardwerte angenommen.

(3) 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.

9.90 @PREFIX – Voranstellen von Zeichenfolgen

Mit der @PREFIX-Anweisung wird jeder Zeile bzw. Zeichenfolgevariable jedes angegebenen Bereiches eine Zeichenfolge vorangestellt (siehe auch @SUFFIX).

Operation	Operanden	F-Modus, L-Modus
@PREFIX	{ lines / svars } [,...] **W**ITH string	

lines — Einer oder mehrere Zeilenbereiche, in denen am Zeilenanfang Text eingefügt werden soll. Es werden nur existierende Zeilen bearbeitet.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen am Beginn Text eingefügt werden soll.

string — Zeichenfolge, die jeder Zeile bzw. Zeichenfolgevariablen jedes angegebenen Bereiches vorangestellt wird. Die Angabe einer leeren Zeichenfolge ist ebenfalls gestattet.

Die Zeichenfolge wird in den Zeichensatz der Arbeitsdatei bzw. der Zeichenfolgevariablen konvertiert. Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), anderenfalls wird die @PREFIX-Anweisung abgebrochen und die Fehlermeldung `EDT5453` ausgegeben.

Überschreitet eine Zeile bzw. eine Zeichenfolgevariable durch das Einfügen die maximale Länge von 32768 Zeichen, dann wird das Einfügen nicht durchgeführt und stattdessen die Meldung `EDT5474` ausgegeben.

Beim Auftreten von Fehlern während der Verarbeitung (`EDT5453` oder `EDT5474`) wird die Anweisung abgebrochen. Evtl. zu diesem Zeitpunkt schon erfolgreich modifizierte Zeilen und/oder Zeichenfolgevariablen bleiben modifiziert.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Beispiel

```
   1.00 UND<·····························································
   2.00 NOCH<····························································
   3.00 EINMAL<··························································
   4.00 EINMAL<··························································
   5.00 EINMAL<··························································
   6.00 ·····························································
 
 
 
 
prefix 4-5 with ' NOCH '·····································0001.00:00001(00)
```

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·····························································0001.00:00001(00)
```

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·····································0001.00:00001(00)
```

Im Zeilenbereich von 4 bis 5 sollen 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·············································0001.00:00001(00)
```

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 ·····························································
```

9.91 @PRINT – Zeilenbereiche bzw. Inhalte von Zeichenfolgevariablen ausgeben

Die Anweisung @PRINT gibt den Inhalt der angegebenen Zeilenbereiche bzw. von Zeichenfolgevariablen aus. Im Dialogbetrieb erfolgt die Ausgabe nach SYSOUT, im Stapelbetrieb nach SYSLST.

In den Operandenbeschreibungen wird zur Vereinfachung meist nur von Zeilen bzw. Zeilenbereichen gesprochen. Die Aussagen gelten jedoch auch für Zeichenfolgevariablen bzw. Bereiche von Zeichenfolgevariablen, wenn nicht explizit auf Abweichungen hingewiesen wird.

Operation	**Operanden**	**F-Modus, L-Modus**
@PRINT	[{lines / svars} [:cols[:]] [**X**] [**N**] [**S**] [{**V** / **E**}]] [,...]	

lines — Zeilenbereich, der ausgegeben werden soll.

svars — Bereich von Zeilchenfolgevariablen, deren Inhalt ausgegeben werden soll.

cols — Spaltenbereich der Arbeitsdatei oder der Zeichenfolgevariablen, der ausgegeben werden soll.

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 ausgegeben.

Wird kein Spaltenbereich angegeben, werden die gesamten Zeilen bzw. Zeichenfolgevariablen ausgegeben, auch wenn sie leer sind

X — Die Ausgabe erfolgt in hexadezimaler Form. Pro Zeichen werden zwei bis acht Hexadezimalziffern ausgegeben, je nach dem Zeichensatz, der für die Arbeitsdatei bzw. für die Zeichenfolgevariablen eingestellt ist.

N — Unterdrückt die Zeilennummern bzw. die Bezeichnung der Zeichenfolgevariablen bei der Ausgabe.

S — Bei Ausgabe auf SYSLST wird normalerweise jede erste Zeile eines Bereiches mit einem zusätzlichen Zeilenvorschub ausgegeben, wenn die Ausgabe nicht an einem Seitenanfang erfolgt. Bei Angabe von S wird die Ausgabe dieser zusätzlichen Leerzeile unterdrückt. Der Operand ist nur im Stapelbetrieb sinnvoll und wird im Dialogbetrieb ignoriert.

V, E

Die Operanden V bzw. E bewirken im Dialogbetrieb, dass der EDT den angegebenen Zeilenbereich abschnittsweise (entsprechend der Bildschirmgröße) ausgibt und gegebenenfalls am Ende eines Abschnitts dem Benutzer die Möglichkeit zur Eingabe von Blätteranweisungen (siehe unten) gibt. Aus wie vielen physikalischen Zeilen (Bildschirmzeilen) ein Abschnitt maximal besteht, hängt von der benutzten Datensichtstation ab.

Die Operanden sind nur beim Arbeiten am Bildschirm sinnvoll und werden im Stapelbetrieb ignoriert.

Bei Angabe von V (oder wenn durch eine Blätteranweisung auf den V-Modus umgeschaltet wurde) wird am Ende eines jeden Abschnitts immer zur Eingabe einer Blätteranweisung aufgefordert.
Nur durch Eingabe von 0 oder durch Wechsel zum E-Modus (siehe unten), kann die Ausgabe beendet bzw. zum nächsten angegebenen Bereich übergegangen werden.

Im Gegensatz dazu wird bei Angabe von E (oder wenn durch eine Blätteranweisung auf den E-Modus umgeschaltet wurde) nur dann am Ende eines Abschnitts zur Eingabe einer Blätteranweisung aufgefordert, wenn dieser Abschnitt nicht bereits die letzte Zeile des aktuellen Bereiches enthält. Enthält der auszugebende Abschnitt die letzte Zeile des aktuellen Bereiches, wird der Abschnitt ohne Aufforderung zum Blättern ausgegeben, wenn kein weiterer Bereich folgt oder der nachfolgende Bereich keinen der Operanden V oder E benutzt.

Wurde ein weiterer Bereich mit Operand V oder E angegeben, so wird der letzte Abschnitt des aktuellen Bereiches mit dem Beginn des nächsten Bereiches zu einem Abschnitt zusammengefügt. Das Verhalten am Ende dieses zusammengefügten Abschnitts bestimmt sich durch den Operanden V oder E des nächsten Bereiches.

Wird weder V noch E angegeben, wird der komplette Bereich ausgegeben. Die Ausgabe wird im Dialogbetrieb 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 oder durch jede andere Eingabe fortsetzen. Weiteres Positionieren ist hierbei nicht möglich.

Wird kein Operand angegeben, wird die ganze aktuelle Arbeitsdatei abschnittsweise (wie bei Angabe von E) ausgegeben.

Der Inhalt des angegebenen Zeilenbereichs wird zeilenweise mit oder ohne vorangestellte Zeilennummer (siehe Operand N) ausgegeben. Ist SYSOUT im Dialogbetrieb einer Datensichtstation zugewiesen, werden die einzelnen Zeilen durch das Zeichen LZE (logisches Zeilenende) voneinander getrennt.
Es können mehrere auszugebende Bereiche mit eigenen Ausgabeoptionen durch Komma getrennt angegeben werden. Die Anzahl der Bereichsangaben ist nur durch die maximal erlaubte Länge einer EDT-Anweisung begrenzt.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Wird im Dialogbetrieb abschnittsweise ausgegeben, fordert der EDT nach der Ausgabe eines *Abschnitts* (Dateiabschnitt oder Folge von Zeichenfolgevariablen) eine *Blättereingabe*, mit der abgebrochen oder positioniert werden kann. Dabei kann der durch lines bzw. svars festgelegte Bereich auch verlassen werden. Die folgenden Blättereingaben sind möglich:

DUE — Die leere Eingabe beim Blättern bewirkt den Übergang zum nächsten (also den dem zuletzt ausgegebenen Abschnitt unmittelbar folgenden) Abschnitt der Ausgabe. Ist im V-Modus bereits das Ende des aktuellen Bereiches erreicht, wird die letzte Zeile dieses Bereiches erneut ausgegeben und wieder zur Eingabe einer Blätteranweisung aufgefordert.
Bei Beginn der Ausgabe eines Bereiches ist der aktuelle Bereich gleich dem angegebenen Bereich. Das kann aber durch die Blätteranweisungen + oder − geändert werden.

* — Es wird zunächst zum E-Modus gewechselt und der aktuelle Bereich (falls verändert) wieder auf den in der Anweisung angegebenen Bereich zurückgesetzt.
Dann wird zum nächsten (also den dem zuletzt ausgegebenen Abschnitt unmittelbar folgenden) Abschnitt der Ausgabe übergegangen, es sei denn, die zuletzt ausgegebene Zeile ist die letzte Zeile des angegebenen Bereiches oder liegt bereits oberhalb des angegebenen Bereiches (durch + möglich). Im diesem Fall wird zum nächsten angegebenen Bereich übergegangen bzw. die Ausgabe beendet.

\+ — Es wird zunächst zum V-Modus gewechselt und als aktueller Bereich wird unabhängig von der Angabe im Operanden lines bzw. svars der Bereich 0.0001-9999.9999 bei Zeilen bzw. #S01-#S20 bei Zeichenfolgevariablen eingestellt.

Dann wird der Abschnitt ausgegeben, der dem zuletzt ausgegebenen Abschnitt unmittelbar folgt. Nach Erreichen der letzten Zeile der Arbeitsdatei bzw. der letzten Zeichenfolgevariablen (#S20) führen weitere Eingaben von + nur zur Ausgabe der letzten Zeile verbunden mit der Aufforderung zur Blättereingabe.

+n
: Es wird zunächst zum V-Modus gewechselt und als aktueller Bereich wird unabhängig von der Angabe im Operanden lines bzw. svars der Bereich 0.0001–9999.9999 bei Zeilen bzw. #S01–#S20 bei Zeichenfolgevariablen eingestellt.
Dann wird der Abschnitt ausgegeben, der in der Arbeitsdatei n Zeilen hinter der letzten ausgegebenen Zeile beginnt (analog für Zeichenfolgevariablen). Nach Erreichen der letzten Zeile der Arbeitsdatei bzw. der letzten Zeichenfolgevariablen (#S20) führen weitere Eingaben von +n nur zur Ausgabe der letzten Zeile, verbunden mit der Aufforderung zur Blättereingabe.

–
: Es wird als aktueller Bereich unabhängig von der Angabe im Operanden lines bzw. svars der Bereich 0.0001–9999.9999 bei Zeilen bzw. #S01–#S20 bei Zeichenfolgevariablen eingestellt (Achtung, es wird nicht wie bei + in den V-Modus gewechselt).
Dann wird der Abschnitt ausgegeben, der unmittelbar vor dem zuletzt ausgegebenen steht. Nach Erreichen der ersten Zeile der Arbeitsdatei bzw. der ersten Zeichenfolgevariablen (#S00) führen weitere Eingaben von – nur zur Ausgabe des mit der ersten Zeile beginnenden Abschnitts, verbunden mit der Aufforderung zur Blättereingabe.

–n
: Es wird als aktueller Bereich unabhängig von der Angabe im Operanden lines bzw. svars der Bereich 0.0001–9999.9999 bei Zeilen bzw. #S01–#S20 bei Zeichenfolgevariablen eingestellt (Achtung, es wird nicht wie bei + in den V-Modus gewechselt).
Dann wird der Abschnitt ausgegeben, der in der Arbeitsdatei n Zeilen vor der ersten Zeile des zuletzt ausgegebenen Abschnitts beginnt (analog für Zeichenfolgevariablen). Nach Erreichen der ersten Zeile der Arbeitsdatei bzw. der ersten Zeichenfolgevariablen (#S00) führen weitere Eingaben von –n nur zur Ausgabe des mit der ersten Zeile beginnenden Abschnitts, verbunden mit der Aufforderung zur Blättereingabe.

0
: Beendet die Ausgabe des aktuellen Bereiches. Wurden in der @PRINT-Anweisung mehrere auszugebende Bereiche angegeben, wird zum nächsten Bereich übergegangen.

Passt bei abschnittsweiser Ausgabe eine lange Zeile nicht mehr auf den Bildschirm, wird der Abschnitt vor dieser Zeile beendet (falls er bereits Zeilen enthält) auch wenn der Bildschirm damit nicht ausgeschöpft wird. Ist bereits die erste Zeile eines Abschnitts länger als der Bildschirm erlaubt, dann wird diese Zeile als ein Abschnitt allein ausgegeben.
Die Ausgabe der Zeile wird im Dialogbetrieb nur unterbrochen, wenn die Überlaufkontrolle des Betriebssystems eingeschaltet ist.

Alle Ausgaben werden vom Zeichensatz der Arbeitsdatei bzw. der Zeichenfolgevariablen in den Zeichensatz von SYSOUT bzw. SYSLST konvertiert.

Werden dabei Zeichen gefunden, denen im Ziel-Zeichensatz kein gültiges Zeichen entspricht, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls werden dafür Leerzeichen eingesetzt.

Die Ausgaben nach SYSLST werden nach Ausgabe von jeweils 132 Zeichen (bzw. 160 Zeichen, wenn Auftragsschalter 6 gesetzt ist) umgebrochen. Die Ausgaben nach SYSOUT werden so umbrochen, wie die Makros WROUT bzw. WRTRD es erfordern. Es wird aber immer an einer Zeichengrenze umbrochen.

Hinweis

Anders als im Kompatibilitäts-Modus wird die Eingabe eines Kommandos an der Aufforderung zur Blättereingabe immer abgelehnt und mit erneuter Ausgabe der Aufforderung zum Blättern quittiert. Um ein Kommando einzugeben muss also @PRINT immer erst durch Eingabe von 0 beendet werden.
Die Ausgaben von @PRINT werden im Dialogbetrieb entweder mit WROUT oder mit WRTRD erzeugt, bei @PRINT...E sogar gemischt. Eine Umlenkung von SYSOUT in eine Datei macht daher im Dialog nur bedingt Sinn, insbesondere auch wegen der im Abschnitt „Systemdateien“ auf Seite 156 beschriebenen Zeichensatz-Problematik.

9.92 @PROC (Format 1) – Umschalten von Arbeitsdateien

Mit diesem Format der @PROC-Anweisung schaltet man in eine andere Arbeitsdatei um. Diese Arbeitsdatei wird damit zur aktuellen Arbeitsdatei.

Operation	Operanden	L-Modus
@PROC	procnr [comment]	

procnr Nummer der Arbeitsdatei (`1..22`), in die gewechselt werden soll.

comment Beliebiger Text als Kommentar.

Die Arbeitsdatei, in die durch die Anweisung gewechselt wurde, bleibt solange aktuell bis mit @END in die zuletzt aktuelle Arbeitsdatei zurückgekehrt wird oder mit einem weiteren @PROC oder @SETF(`procnr`) in eine andere Arbeitsdatei gewechselt wird.

Wenn mit @PROC in eine andere Arbeitsdatei gewechselt wird, merkt sich der EDT die verlassene Arbeitsdatei und ihre Vorgänger (geschachtelte Arbeitsdateien), d.h. mit der @END-Anweisung kommt man in die verlassene(n) Arbeitsdatei(en) zurück. Im Gegensatz dazu wird beim Wechsel der Arbeitsdatei mit @SETF(`procnr`) vor dem Umschalten in die neue Arbeitsdatei die Schachtelung der Arbeitsdateien aufgehoben, eine anschließende @END-Anweisung führt dann stets in die Arbeitsdatei 0 zurück (sofern man nicht schon mit @SETF in Arbeitsdatei 0 gewechselt hatte). Die Schachtelungstiefe ist auf 22 beschränkt, danach wird die Fehlermeldung `EDT4962` ausgegeben.

Gibt man in der @PROC-Anweisung die Nummer der aktuellen Arbeitsdatei an, wird die Anweisung mit der Meldung `EDT4909` abgewiesen. Eine aktive Arbeitsdatei kann nicht zur aktuellen Arbeitsdatei werden, beim Versuch wird die Meldung `EDT4959` ausgegeben.

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 ------------------------------------------------------- (1)
5.     @PROC 10 --------------------------------------------------- (2)
1.     @ @SET #S10 = CHAR #I6
2.     @ @ON #S10:2-2: DELETE '0'
3.     @ @IF .TRUE. : @GOTO 2
4.     @END
5.     @PROC 12 --------------------------------------------------- (3)
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.
```

(1) Es wird in die Arbeitsdatei 0 zurückgeschaltet. 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.

(2) Die Arbeitsdatei 10 wird eingerichtet. Sie enthält eine EDT-Prozedur, die den in `#I6` festgehaltenen Wert in `#S10` abdruckbar abbildet und führende Nullen löscht.

(3) Die Arbeitsdatei 12 wird eingerichtet. Sie enthält eine EDT-Prozedur, die den Inhalt der Zeilennummervariablen `#L6` in `#S12` abdruckbar abbildet.

Beispiel 2

```
1.     @SET #I4 = 1
1.     @PROC #I4 ------------------------------------------------------ (1)
1.     @4.00
4.00   @ @SET #I4 = #I4 + 1 ------------------------------------------- (2)
4.01   @ @IF #I4 > 4 : @GOTO 8
4.02   @ @PROC #I4
4.03   @ @PROC ------------------------------------------------------- (3)
4.04   @ @GOTO 4
4.05   @8.00
8.00   @ @SET #I4 = #I4 - 1
8.01   @ @END -------------------------------------------------------- (4)
8.02   @ @IF #I4 = 2 : @RETURN
8.03   @ @PROC ------------------------------------------------------- (5)
8.04   @ @GOTO 8
8.05   @END
1.     @DO #I4 -------------------------------------------------------- (6)
<02>
<03>
<04>
<03>
<02>
1.
```

(1) In eine Arbeitsdatei kann auch über eine Ganzzahlvariable umgeschaltet werden. Die Ganzzahl muss zwischen 1 und 22 liegen.

(2) Beim Ausführen der Arbeitsdatei 1 wird in die Arbeitsdateien 2 bis 4 umgeschaltet, wobei die jeweilige Arbeitsdateinummer immer in `#I4` übergeben wird.

(3) Mit @PROC wird abgefragt, in welcher Arbeitsdatei man sich befindet. Somit wird beim Ausführen der Arbeitsdatei `#I4` an dieser Stelle das jeweilige `#I4` protokolliert.

(4) Ein einziges @END wird verwendet, um in allen Fällen wieder in die Arbeitsdatei, die eine Ebene tiefer liegt, zurückzuschalten.

(5) Nach dem Zurückschalten wird wieder gefragt, in welcher Arbeitsdatei man sich befindet. Somit wird eine absteigende Sequenz der Arbeitsdateinummern 3 bis 2 ausgegeben.

(6) Auch der erste Aufruf einer Arbeitsdatei kann über eine Ganzzahlvariable erfolgen.

9.93 @PROC (Format 2) – Ausgeben von Informationen über Arbeitsdateien

Mit diesem Format der @PROC-Anweisung kann man sich die Nummer der aktuellen Arbeitsdatei, die Nummern aller freien Arbeitsdateien bzw. die Nummern aller belegten Arbeitsdateien ausgeben lassen. Eine Arbeitsdatei (ungleich 0) gilt als belegt, wenn sie einmal aktuelle Arbeitsdatei war oder ist und nicht zwischenzeitlich mit @DROP oder @DELETE (Format 2) freigegeben worden ist. Eine belegte Arbeitsdatei muss nicht unbedingt Datensätze enthalten.

Operation	Operanden	L-Modus
@PROC	[{ **F**REE / **U**SED }]	

FREE Es sollen die Nummern jener Arbeitsdateien (1-22) ausgegeben werden, die noch nicht belegt wurden.

Ist keine Arbeitsdatei außer der Arbeitsdatei `0` belegt, gibt der EDT die Meldung `EDT0907` aus. Sind alle Arbeitsdateien belegt, erfolgt keine Ausgabe.

USED Es sollen die Nummern jener Arbeitsdateien (1-22) ausgegeben werden, 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 die Meldung `EDT0907` aus.

Wird kein Operand angegeben, dann wird die Nummer der aktuellen Arbeitsdatei am Bildschirm angezeigt.

Hinweis

Im Kompatibilitäts-Modus gilt eine Arbeitsdatei erst dann als belegt, wenn sie einmal aktuelle Arbeitsdatei war, mit einer @END-Anweisung, einer @PROC-Anweisung, einer @SETF-Anweisung bzw. im F-Modus über eine der Anweisungen `0..22` verlassen wurde und sie nicht zwischenzeitlich mit @DROP freigegeben worden ist. Dort gilt also die z.B. mit @PROC eingestellte aktuelle Arbeitsdatei nicht als belegt, selbst wenn sie Daten enthält.

Beispiel 1

```
   1.     @PROC ------------------------------------------------------- (1)
   <00>
   1.     @PROC 15 ---------------------------------------------------- (2)
   1.     @PROC ------------------------------------------------------- (3)
   <15>
   1.     @END -------------------------------------------------------- (4)
   1.     @PROC ------------------------------------------------------- (5)
   <00>
```

(1) Die Abfrage nach der aktuellen Arbeitsdatei bringt die Nummer <00>.

(2) Es wird in die Arbeitsdatei 15 umgeschaltet.

(3) Bei Abfrage der Nummer der Arbeitsdatei wird dann <15 > ausgegeben.

(4) Es wird in die Arbeitsdatei 0 zurückgekehrt.

(5) Man befindet sich jetzt wieder in der Arbeitsdatei 0.

Beispiel 2

```
   1.     @DROP ALL --------------------------------------------------- (1)
   1.     @PROC USED -------------------------------------------------- (2)
% EDT0907 NO WORK FILES USED
   1.     @PROC 13
   1.     A
   2.     @END
   1.     @PROC USED -------------------------------------------------- (3)
   <13>     1.0000 TO    1.0000
```

(1) Alle Arbeitsdateien werden freigegeben.

(2) Es wird gefragt, welche Arbeitsdateien belegt sind.

(3) Nach Anlegen der Arbeitsdatei 13 wird wieder gefragt, welche Arbeitsdateien belegt sind.

Beispiel 3

```
   1.     @DROP ALL ------------------------------------------------- (1)
   1.     @PROC FREE ------------------------------------------------ (2)
% EDT0907 NO WORK FILES USED
   1.     @PROC 13
   1.     A
   2.     @END
   1.     @PROC FREE ------------------------------------------------ (3)
   01 02 03 04 05 06 07 08 09 10 11 12 14 15 16 17 18 19 20 21 22
```

(1) Alle Arbeitsdateien werden freigegeben.

(2) Es wird gefragt, welche Arbeitsdateien nicht belegt sind.

(3) Nach Anlegen der Arbeitsdatei 13 wird wieder gefragt, welche Arbeitsdateien nicht belegt sind.

9.94 @QUOTE – Begrenzersymbol für Zeichenfolgen umdefinieren

Mit der Anweisung @QUOTE können die Begrenzersymbole apostrophe (Hochkomma) und quotation mark (Anführungszeichen) umdefiniert werden (siehe Abschnitt „Begrenzersymbole“ auf Seite 84).

Zeichenfolgen, die in diese Begrenzersymbole eingeschlossen sind, sind Literale, die in Anweisungen eine besondere Rolle spielen (siehe Abschnitt „Anweisungssyntax“ auf Seite 165).

Operation	Operanden	F-Modus, L-Modus
@QUOTE **@QE**	[spec] [,char]	

spec Sonderzeichen, das als *apostrophe* vereinbart werden soll. Beim Start des EDT ist ' eingestellt.

char Zeichen, das als *quotation mark* vereinbart werden soll. Beim Start des EDT ist " eingestellt.

Einer der beiden Operanden muss zwingend angegeben werden. Sonderzeichen, die eine besondere Bedeutung haben wie Leerzeichen, Semikolon im F-Modus oder Komma können nicht angegeben werden.

Die Zeichen *apostrophe* und *quotation mark* müssen verschieden sein, sonst wird die Anweisung mit der Meldung `EDT4903` abgewiesen. Sie müssen auch von den Musterzeichen verschieden sein, sonst wird die Anweisung mit der Meldung `EDT5461` abgelehnt.

Ist `spec` keines der zugelassenen Sonderzeichen, wird @QUOTE mit der Fehlermeldung `EDT3952` abgewiesen.

Hinweis

Die Vereinbarung wirkt nicht für Operanden der Anweisungen @DO und @PARAMS.

9.95 @RANGE – Zeilenbereichssymbol vereinbaren

Mit der Anweisung @RANGE kann ein Symbol für einen Zeilenbereich vereinbart werden.

Operation	**Operanden**	**F-Modus, L-Modus**
@RANGE	[= spec = { lines | svars }]	

spec — Sonderzeichen, das als Zeilenbereichssymbol vereinbart werden soll. Ist `spec` keines der zugelassenen Sonderzeichen, wird @RANGE mit der Fehlermeldung `EDT3952` abgewiesen.

lines — Zeilenbereich, der dem Zeilenbereichssymbol zugeordnet werden soll.

svars — Bereich von Zeichenfolgevariablen, der dem Zeilenbereichssymbol zugeordnet werden soll.

Beim Start des EDT ist als Zeilenbereichssymbol das Zeichen `&` mit dem Zeilenbereich `0.0001-9999.9999` definiert. Wird ein neues Zeilenbereichssymbol vereinbart, wird das alte ungültig.

Wird kein Operand angegeben, wird das Zeilenbereichssymbol zurückgenommen. Es existiert dann kein Zeilenbereichssymbol, bis wieder ein neues durch @RANGE vereinbart wird.

9.96 @READ – Einlesen einer SAM-Datei

Mit der Anweisung @READ wird eine SAM-Datei ganz oder teilweise von Platte oder Band in die aktuelle Arbeitsdatei eingelesen.

Operation	Operanden	F-Modus, L-Modus
@READ	[file] [(ver)] [lines[,...]] [:cols[,...]:] [{ **R**ECORD**S** / **K**EY }] [**S**TRIP]	

file — Name der SAM-Datei, die eingelesen werden soll. Der Name muss dem SDF-Datentyp <filename 1..54> entsprechen oder die spezielle Angabe '/' sein.

Besteht noch kein lokaler @FILE-Eintrag für die Arbeitsdatei, so wird der angegebene Dateiname bei Erfolg als impliziter lokaler @FILE-Eintrag eingetragen. Fehlt der Operand file, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag und andernfalls der globale @FILE-Eintrag als Dateiname herangezogen (siehe auch @FILE-Anweisung).
Ist weder ein expliziter lokaler noch ein globaler @FILE-Eintrag definiert (z.B. nur ein impliziter lokaler @FILE-Eintrag), wird die Anweisung @READ mit der Fehlermeldung EDT5484 abgewiesen.

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Wenn der Dateikettungsname EDTSAM einer Datei zugeordnet ist, genügt die Angabe '/', um diese Datei einzulesen (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

ver — Versionsnummer der einzulesenden Datei. Stimmt die angegebene Versionsnummer nicht mit der Versionsnummer der Datei überein, wird die Meldung EDT0902 ausgegeben, die Datei aber trotzdem eingelesen.

lines — Einer oder mehrere Zeilenbereiche, die aus der SAM-Datei eingelesen werden sollen. Werden symbolische Zeilennummern angegeben, werden deren Werte aus der aktuellen Arbeitsdatei bestimmt, haben also normalerweise nichts mit der Satzstruktur der bei file angegebenen Datei zu tun.

Wird lines zusammen mit RECORDS angegeben, bezieht sich lines auf die logischen Zeilennummern der Datei (siehe RECORDS). Wird lines ohne RECORDS angegeben, wirkt dies wie die Angabe von lines mit KEY.

Wird lines nicht angegeben, wird die gesamte Datei eingelesen.

cols
: Einer oder mehrere Spaltenbereiche, durch die der einzulesende Bereich jedes Satzes festgelegt wird. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Die Spaltenangaben beziehen sich auf die Zeichen in der einzulesenden Datei. Bei Dateien, die in einem Unicode-Zeichensatz vorliegen, stimmt das zumeist nicht mit der Byteposition innerhalb eines Satzes überein. Werden Spaltenwerte angegeben, die die Länge eines Satzes überschreiten, so werden dafür Leerzeichen in die Arbeitsdatei eingelesen. Bei Angabe von KEY (bzw. lines ohne RECORDS) beginnt die Spaltenzählung erst nach dem Schlüssel im Datensatz.

  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 eingelesen werden, erhalten diesen Schlüssel als Zeilennummer und nicht als Teil des Zeileninhalts. Der Operand KEY ist Standardwert, wenn lines angegeben wurde.

  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 die Anweisung @READ mit der Fehlermeldung EDT4984 abgebrochen. Die bis zu diesem Zeitpunkt gelesenen Sätze werden in die Arbeitsdatei übernommen. Hat ein Satz den Schlüssel 0, dann wird er wie ein Satz mit dem Schlüssel 1 (Zeilennummer 0.0001) behandelt und die Warnung EDT2900 ausgegeben.

  Wird über lines eine Auswahl von Zeilen vorgenommen, geht der EDT davon aus, dass die Sätze der Datei aufsteigende Schlüssel haben. Ist das nicht der Fall werden gegebenenfalls nicht alle erwarteten Sätze eingelesen.

RECORDS
: Legt fest, dass ein Zeilenbereich (siehe Operand lines) über die logische Zeilennummer der Datei ausgewählt werden soll. Die logische Zeilennummer der ersten Zeile ist 0.0001, die der zweiten 0.0002 usw. Die eingelesenen Sätze werden an der durch die aktuelle Zeilennummer bestimmten Stelle in die Arbeitsdatei eingefügt (siehe unten).

  Wenn STRIP nicht angegeben wird, kann das Schlüsselwort RECORDS auch durch R abgekürzt werden. Der Operand RECORDS ist Standardwert, wenn lines nicht angegeben wurde.

STRIP
: Bewirkt, dass eventuell vorhandene Leerzeichen am Ende jeder erzeugten Arbeitsdateizeile gelöscht werden. Besteht eine Zeile nur aus Leerzeichen, werden alle Leerzeichen bis auf eines gelöscht.

Ist KEY (bzw. lines ohne RECORDS) angegeben, werden Zeilen ignoriert, die weniger als 8 Zeichen lang sind.

Bei der Angabe von lines bzw. cols können sich Zeilennummern bzw. Spaltennummern wiederholen, was zu einem mehrmaligen Einlesen der entsprechenden Zeilen bzw. Spalten führt.

Sofern ohne die Angabe KEY gelesen wird, werden die Zeilennummern abhängig von der aktuellen Zeilennummer und der aktuellen Schrittweite vergeben (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Die Datei wird nur während des Lesevorgangs temporär geöffnet. Ist die einzulesende Datei leer, wird die Warnung EDT2903 ausgegeben.

Ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz *NONE, erhält die Arbeitsdatei den Zeichensatz der einzulesenden Datei. Ist dieser *NONE, erhält die Arbeitsdatei den Zeichensatz EDF03IRV.

Hat die Arbeitsdatei bereits einen Zeichensatz, dann werden die einzulesenden Sätze vom Zeichensatz der Datei in den Zeichensatz der Arbeitsdatei konvertiert. Enthält die einzulesende Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird das Einlesen der Datei abgebrochen und die Fehlermeldung EDT5453 ausgegeben. Dies gilt auch, falls sich das ungültige Zeichen außerhalb des einzulesenden Spaltenbereichs befindet. Ungültige Zeichen außerhalb des einzulesenden Zeilenbereichs werden hingegen ignoriert.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird das Lesen mit der Meldung EDT5454 abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Hinweis

Wird versucht, mit @READ eine ISAM-Datei einzulesen, gibt der EDT die Meldung EDT1901 aus und setzt den Schalter für EDT-Fehler. Er liest die angegebene Datei aber trotzdem ein, indem intern ein @GET für diese Datei ausgeführt wird.
Die Operanden STRIP, RECORDS und KEY werden dabei ignoriert.

Wird file angegeben, kann @READ im F-Modus aus Kompatibilitätsgründen auch weiterhin mit R abgekürzt werden.

9.97 @RENUMBER – Neu nummerieren

Mit der Anweisung @RENUMBER werden die in der Arbeitsdatei befindlichen Zeilen neu nummeriert. Es kann sowohl die Zeilennummer angegeben werden, die die erste Zeile der Arbeitsdatei erhalten soll als auch die Schrittweite, mit der neu nummeriert werden soll. Diese Schrittweite wird auch zur neuen aktuellen Schrittweite (unabhängig davon, ob sie explizit angegeben ist oder implizit durch die angegebene Zeilennummer festgelegt wird). Die neue aktuelle Zeilennummer ergibt sich aus der nach der Neunummerierung höchsten Zeilennummer plus der aktuellen Schrittweite.

Operation	**Operanden**	**F-Modus, L-Modus**
@RENUMBER	[line [(inc)]]	

line — Der Operand gibt die Zeilennummer an, die die erste Zeile der Arbeitsdatei durch die Neunummerierung erhalten soll.

Ist der Operand `line` nicht angegeben, erhält die erste Zeile der Arbeitsdatei die Zeilennummer `1`.

inc — Der Operand gibt die neue aktuelle Schrittweite an. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

Die Anweisung @RENUMBER kann nicht für eine durch die Anweisung @OPEN (Format 2) real geöffnete Datei gegeben werden.

Für eine leere Arbeitsdatei wird @RENUMBER ignoriert, insbesondere wird auch keine neue Schrittweite eingestellt.

Wird die Anweisung im Dialogbetrieb 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)?
```

Wird die Meldung `EDT0911` mit N beantwortet, so wird die Anweisung @RENUMBER nicht ausgeführt. Wird dagegen die Meldung `EDT0911` mit `Y` beantwortet, so wird die Anweisung @RENUMBER ausgeführt, überzählige Zeilen werden gelöscht und es wird die Meldung `EDT2904` ausgegeben.

Im F-Modus bleibt der Dateiausschnitt im Datenfenster nach einer Neunummerierung mit der Anweisung @RENUMBER normalerweise erhalten. Es ändert sich nur die Nummernanzeige. Die einzige Ausnahme davon ist, wenn durch die Umnummerierung Zeilen am Ende der Arbeitsdatei verloren gehen, die vor der Umnummerierung im Datenfenster sichtbar waren. Nach der Umnummerierung werden die verloren gegangenen Zeilen nicht mehr

im Datenfenster angezeigt. Gehen alle im Datenfenster sichtbaren Zeilen durch die Umnummerierung verloren, so wird im Datenfenster nur noch die letzte nach der Umnummerierung noch vorhandene Zeile der Arbeitsdatei angezeigt.

Die Bearbeitung der Anweisung @RENUMBER wird auch dann nicht abgebrochen, wenn der EDT-Lauf nach Unterbrechung mit K2 mit `/INFORM-PROGRAM` fortgesetzt wird.

Hinweis

Die von der Anweisung @SET (Format 6) in einem internen Speicherbereich abgelegten Paare Zeilennummer/Schrittweite (siehe Anweisung @SET, Format 6) werden von der Anweisung @RENUMBER nicht verändert und auch nicht die im Kopierpuffer eventuell vorhandenen Zeilennummern (siehe Kurzanweisungen, `C`, `R`, `M`).
Nach einer erfolgreichen Neunummerierung durch die Anweisung @RENUMBER haben deshalb die Zeilen, die ursprünglich durch die in diesen beiden Bereichen abgelegten Zeilennummern identifiziert wurden, möglicherweise neue Zeilennummern. Deshalb ist es in der Regel nicht sinnvoll, nach einer Umnummerierung durch die Anweisung @RENUMBER auf die in diesen beiden Bereichen abgelegten Zeilennummern zurückzugreifen (durch die Anweisung @SET ohne Operanden bzw. durch die Kurzanweisungen `A`, `B` oder `O`).

9.98 @RESET – EDT- und DVS-Fehlerschalter rücksetzen

Mit der Anweisung @RESET können EDT- und DVS-Fehlerschalter zurückgesetzt werden.

Operation	**Operanden**	**F-Modus, L-Modus**
@RESET		

Es können nur beide Fehlerschalter simultan zurückgesetzt werden, eine Auswahl ist nicht möglich. Die Anweisung wird hauptsächlich in EDT-Prozeduren im Zusammenhang mit @IF (Format 1) benutzt.

9.99 @RETURN – Rückkehr aus EDT-Prozeduren

Die Anweisung @RETURN wird in EDT-Prozeduren benutzt, um die Abarbeitung der Prozedur zu beenden und an die Stelle ihres Aufrufs zurückzukehren. Wird die Anweisung @RETURN außerhalb einer EDT-Prozedur gegeben, wird der EDT-Lauf bzw. der Bildschirmdialog nach @DIALOG beendet.

Operation	Operanden	F-Modus, L-Modus
@RETURN	[message]	

message Der Operand `message` kann beliebigen Text enthalten, der bei Aufruf des EDT als Unterprogramm an das aufrufende Programm übergeben wird.

Der Operand `message` darf nur bei Aufruf des EDT über die Unterprogramm-Schnittstelle bzw. bei Rückkehr aus einer EDT-Prozedur angegeben werden.

Zwischen dem Anweisungsnamen und einem evtl. angegebenen Operanden muss bei dieser Anweisung zwingend mindestens ein Leerzeichen angegeben werden.

Wird die Anweisung @RETURN außerhalb einer EDT-Prozedur verwendet, wirkt sie wie @HALT (siehe @HALT-Anweisung).

Wird @RETURN in @DO- oder @INPUT-Prozeduren verwendet, wird die Abarbeitung der Prozedur abgebrochen und dort weitergearbeitet, wo der Prozeduraufruf erfolgte. Der Operand `message` wird in diesem Fall ignoriert.

Hinweis

Eine @RETURN-Anweisung in einer @DO-Prozedur bricht auch eine äußere Schleife ab, d.h. die Prozedur wird dann nicht so oft ausgeführt, wie durch Anfangswert, Endwert und Schrittweite des Schleifenzählers in der @DO-Anweisung spezifiziert ist (siehe Abschnitt „EDT-Prozeduren“ auf Seite 66 und @DO-Anweisung).

Beispiel 1

```
1.     @PROC 6 ------------------------------------------------------ (1)
1.     @ @SET #S1 = 'ICH BIN #S1'
2.     @ @SET #S2 = 'ICH BIN #S2'
3.     @ @PRINT #S1
4.     @ @RETURN ---------------------------------------------------- (2)
5.     @ @PRINT #S2
6.     @END --------------------------------------------------------- (3)
1.     @DO 6 -------------------------------------------------------- (4)
  #S01 ICH BIN #S1
```

(1) Die Arbeitsdatei 6 wird zur Bearbeitung geöffnet.

(2) Kommt es beim Ausführen der Prozedur in Arbeitsdatei 6 zu dieser Anweisung, werden die nachfolgenden Anweisungen nicht mehr ausgeführt.

(3) Die Bearbeitung der Arbeitsdatei 6 wird beendet.

(4) Die Prozedur wird ausgeführt.

Beispiel 2

```
1.     AAAA
2.     BBBB
3.     CCCC
4.     @PROC 7 ------------------------------------------------------ (1)
1.     @ @PRINT ! --------------------------------------------------- (2)
2.     @ @RETURN ---------------------------------------------------- (3)
3.     @END
4.     @DO 7,!=%,$ -------------------------------------------------- (4)
1.0000 AAAA
4.
```

(1) Die Prozedur wird in der Arbeitsdatei 7 erstellt.

(2) Beim Ausführen der Anweisungen der Prozedur soll die über den Schleifenzähler ! angesprochene Zeile ausgegeben werden.

(3) Die Ausführung der Anweisungen der Prozedur wird an dieser Stelle unterbrochen, egal ob der Schleifenzähler ! bereits die obere Grenze erreicht hat oder nicht.

(4) Die Prozedur wird aufgerufen. Dabei soll der Schleifenzähler `!` die Werte `1` bis `3` `(%=1, $=3)` annehmen. Wegen des in der Prozedur stehenden @RETURN kommt es jedoch nicht zu einem Hochzählen des Schleifenzählers.

9.100 @RUN – Aufruf einer Anwenderroutine

Mit der Anweisung @RUN wird eine vom Anwender geschriebene Routine (Anwenderroutine) ausgeführt (siehe Handbuch Unterprogramm-Schnittstellen [1]).

Operation	Operanden	F-Modus, L-Modus
@RUN	**E**NTRY=entry [,**M**ODLIB=modlib] [,**U**NLOAD] [,string]	

entry — Einsprungpunkt der Anwenderroutine.

modlib — Name der Bibliothek, in der der Modul abgelegt ist, der den Einsprungpunkt enthält. Die Bibliothek muss existieren, sonst wird die Anweisung mit der Fehlermeldung `EDT5372` abgewiesen.

Wird kein Modul mit dem Einsprungpunkt in der angegebenen Bibliothek gefunden, wird zuerst in den alternativen Bibliotheken `BLSLIBnn` gesucht, dann in der privaten Tasklib bzw. in der System-Tasklib `$TASKLIB`.

Ist keine Bibliothek angegeben, wird zunächst in der privaten Tasklib und dann in der System-Tasklib `$TASKLIB` gesucht.

Bei Misserfolg wird die Fehlermeldung `EDT5372` ausgegeben.

UNLOAD — Gibt an, dass die Ladeeinheit, die den Einsprungpunkt enthält, nach der Rückkehr in den EDT entladen werden soll. Der Operand `UNLOAD` wirkt, als ob eine separate Anweisung @UNLOAD UNIT=`entry` gegeben worden wäre, d.h. er führt nur dann zum Entladen, wenn der angegebene Einsprungpunkt gleichzeitig der Name einer Ladeeinheit ist (siehe unten). Wenn etwa der Einsprungpunkt in einer schon geladenen Ladeeinheit anderen Namens gefunden wurde, kann `UNLOAD` nicht ausgeführt werden. In diesem Fall wird die Meldung `EDT1907` ausgegeben.

string — Zeichenfolge, die an das aufgerufene Programm übergeben wird.

Die Implementierung externer Anwenderroutinen und die Parameterübergabe an diese sind im Abschnitt „Anwenderroutinen-@RUN“ [1] genauer beschrieben.

Ist für die Anwenderroutine keine Initialisierungsroute definiert, wird die Anweisung mit der Meldung `EDT5469` abgewiesen. Liefert die zugehörige Initialisierungsroutine einen Returncode, wird die Anweisung mit der Meldung `EDT5470` abgewiesen, falls die Initialisierungsroutine die Version nicht unterstützt, und mit der Meldung `EDT5471` falls die Initialisierungsroutine einen sonstigen Fehler meldet.

Führt die Bearbeitung der @RUN-Anweisung zu einem eigenen Ladevorgang, wird dem Binder-Lader-System ein zu dem angegebenen Einsprungpunkt gleich lautender UNIT-Name bekannt gemacht.

Dieser UNIT-Name kann bei der @UNLOAD-Anweisung angegeben werden, um alle zusammen mit dem Einsprungpunkt geladenen Ladeeinheiten zu entladen. Ebenso bezieht sich der UNLOAD Operand der @RUN-Anweisung auf diesen UNIT-Namen. Wird der Einsprungpunkt bereits innerhalb einer anderen Ladeeinheit gefunden, führt @RUN also nicht zu einem eigenen Ladevorgang, kann der Einsprungpunkt nur zusammen mit dieser Ladeeinheit entladen werden.

Die Anweisung @RUN gehört zu den sicherheitsrelevanten Anweisungen des EDT (siehe hierzu auch den Abschnitt „Zugriffsschutz“ auf Seite 103). In bestimmten privilegierten Kennungen wird die Anweisung @RUN abgewiesen. Dies trifft auch für nichtunterbrechbare Systemprozeduren im Dialogbetrieb bzw. bei der Eingabe von einer Datei (Lesen mit RDATA von SYSDTA, Abarbeiten einer EDT-Startprozedur) zu, es sei denn, die Anweisung @RUN wird von der geschützten Prozedur selbst gegeben (SYSDTA=SYSCMD).

Hinweis

Für den Operanden entry, der Namen bis zu 32 Zeichen akzeptiert, ist Groß-/Kleinschreibung relevant.

Achtung

Das Format der Anweisung und die Schnittstelle, mit der die Routine gerufen wird, sind im Unicode-Modus anders als im Kompatibilitäts-Modus.

9.101 @SAVE – Schreiben als ISAM-Datei

Mit der Anweisung @SAVE wird der Inhalt der aktuellen Arbeitsdatei ganz oder teilweise als ISAM-Datei auf Platte geschrieben.

Operation	Operanden	F-Modus, L-Modus
@SAVE	[file] [(ver)] [lines[,...]] [:cols[,...]:] [{ **U**PDATE [**R**ENUMBER [line [(inc)]]] [**O**VERWRITE] }]	

file — Name der ISAM-Datei, die geschrieben werden soll. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen oder die spezielle Angabe `'/'` sein.

Fehlt der Operand `file`, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag, danach der globale @FILE-Eintrag oder als letztes der implizite lokale @FILE-Eintrag (z.B. aus der @GET-Anweisung) als Dateiname herangezogen (siehe auch @FILE-Anweisung). Ist weder ein lokaler noch globaler @FILE-Eintrag vorhanden, wird die Anweisung @SAVE mit der Fehlermeldung `EDT5484` abgewiesen.

Ist die angegebene Datei nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Wenn der Dateikettungsname `EDTISAM` einer Datei zugeordnet ist, genügt die Angabe `'/'`, um diese Datei zu schreiben (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

ver — Versionsnummer der zu überschreibenden Datei. Wird für eine existierende Datei die falsche Versionsnummer angegeben, wird die Anweisung mit der Meldung `EDT4985` abgewiesen. Bei einer noch nicht existierenden Datei wird die Angabe ignoriert und grundsätzlich die Version `001` geschrieben.

lines — Einer oder mehrere Zeilenbereiche, die in die ISAM-Datei geschrieben werden sollen. Werden dabei Zeilen mehrfach angegeben, so werden sie auch mehrfach geschrieben.

Wird `lines` nicht angegeben, wird die gesamte Datei geschrieben.

cols
: Einer oder mehrere Spaltenbereiche, durch die der zu schreibende Bereich jedes Satzes festgelegt wird. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Die Spaltenangaben beziehen sich auf die *Zeichen* in der aktuellen Arbeitsdatei. Werden Spaltenwerte angegeben, die die Länge eines Arbeitsdateisatzes überschreiten, so werden dafür Leerzeichen in die Datei geschrieben.

  Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge geschrieben.

UPDATE
: Die Angabe von `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 aktuellen Arbeitsdatei existieren, und die im Bereich `lines` liegen. Die restlichen Zeilen der ISAM-Datei bleiben erhalten.

  Dieser Operand wird ignoriert, wenn keine ISAM-Datei mit dem angegebenen Namen existiert.

RENUMBER
: Für die abzuspeichernden Zeilen werden neue ISAM-Satzschlüssel gebildet. Die Zeilennummerierung in der Arbeitsdatei bleibt unverändert. Werden Zeilen mehrfach ausgegeben (durch Überlappungen in den Bereichsangaben), werden diese auch mehrfach (mit unterschiedlichen Satzschlüsseln) in die Datei aufgenommen.

  Wird `RENUMBER` nicht angegeben, entstehen die ISAM-Schlüssel aus den Zeilennummern der abzuspeichernden Zeilen. Soll die Datei später mit @GET ... NORESEQ eingelesen werden und dabei die gleiche Zeilennummerierung wie zum Zeitpunkt des Abspeicherns aufweisen, darf `RENUMBER` *nicht* angegeben werden.

  Wird bei der Neunummerierung die maximal mögliche Zeilennummer (entspr. der Schlüssellänge) überschritten, wird die Anweisung mit der Fehlermeldung `EDT5252` abgebrochen.

line
: Startnummer für die neu zu bildenden ISAM-Satzschlüssel. Wird `line` nicht angegeben, so wird `1` angenommen.

inc
: Schrittweite der neu zu bildenden ISAM-Satzschlüssel. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

OVERWRITE
: Eine vorhandene Datei gleichen Namens wird ohne Rückfrage überschrieben. Existiert die angegebene Datei noch nicht, ist `OVERWRITE` wirkungslos.

Wird weder `UPDATE` noch `OVERWRITE` angegeben und existiert bereits eine Datei mit dem gleichen Namen, reagiert der EDT im Dialogbetrieb mit der Frage:

```
%  EDT0903 FILE 'file' IS IN THE CATALOG, FCBTYPE = fcbtyp
%  EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)?
```

Wird die Meldung mit `Y` beantwortet, wird die bestehende Datei als ISAM-Datei mit dem Inhalt der aktuellen Arbeitsdatei überschrieben. Wird dagegen die Meldung mit `N` beantwortet, wird die Datei nicht geschrieben und die Meldung `EDT0293` ausgegeben.
Im Stapelbetrieb wird die Datei immer überschrieben.

Beim Überschreiben einer existierenden Datei durch @SAVE ohne den Operanden `UPDATE` ändern sich möglicherweise Datei-Typ und Datei-Attribute. Die Datei wird als ISAM-Datei mit Standardattributen (z.B. variable Satzlänge) geschrieben, sofern nicht vorher ein entsprechendes `/SET-FILE-LINK`-Kommando mit dem Dateikettungsnamen `EDTISAM` und den abweichenden Attributen gegeben worden ist (siehe Kapitel „Dateibearbeitung“ auf Seite 137). Dateien vom Typ `PAM` oder `BTAM` können nicht überschrieben werden.

Die Datei wird nur während des Schreibvorgangs temporär geöffnet.

Der Zeichensatz, mit dem das Schreiben erfolgt, hängt davon ab, ob die Datei überschrieben, neu angelegt oder erweitert wird (siehe Operand `UPDATE`).

Wird die Datei überschrieben oder neu angelegt, werden die Daten im Zeichensatz der Arbeitsdatei geschrieben und für die Datei wird dieser Zeichensatz im Katalog eingetragen.

Wird die Datei erweitert, werden die Daten vom Zeichensatz der Arbeitsdatei in den Zeichensatz umcodiert, der im Katalogeintrag der Datei spezifiziert ist.
Ist im Katalog der Datei `*NONE` eingetragen, wird `EDF03IRV` angenommen (siehe auch Abschnitt „Zeichensätze“ auf Seite 48). Enthält die Arbeitsdatei Zeichen, die im Zeichensatz der zu schreibenden Datei ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht geschrieben und die Fehlermeldung `EDT5453` ausgegeben.
Die gilt nicht für ungültige Zeichen außerhalb des zu schreibenden Zeilen- oder Spaltenbereichs. Diese werden ignoriert.

Enthält die Arbeitsdatei Zeilen, die für die zu schreibende Datei zu lang sind (z.B. bei fester Satzlänge der Datei) oder entstehen durch die Konvertierung solche Sätze (bei Unicode-Zeichensätzen möglich), dann wird der Schreibvorgang mit der Meldung `EDT5444` abgebrochen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Wird `file` angegeben, kann @SAVE im F-Modus aus Kompatibilitätsgründen auch weiterhin mit `S` abgekürzt werden.

9.102 @SCALE – Spaltenzähler ausgeben

Die Anweisung @SCALE schaltet im F-Modus die Anzeige eines Spaltenzählers (Zeilenlineal) für die aktuelle Arbeitsdatei im Arbeitsfenster ein oder aus (siehe auch den Abschnitt „Das Arbeitsfenster“ auf Seite 107).

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

ON — Schaltet die Anzeige des Spaltenzählers ein (Standardwert).

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).

Falls ein Tabulator definiert ist (siehe Anweisung @TABS), wird eine weitere Bildschirmzeile ausgegeben, in der die aktuellen Positionen des Tabulators mit `'I'` angezeigt werden. In dieser Zeile wird auch das Tabulatorzeichen selbst in der Kurzanweisungsspalte abgebildet.

OFF — Schaltet die Anzeige des Spaltenzählers und der eventuell angezeigten Tabulatorpositionen aus.

Bei Beginn des EDT-Laufs ist die Spaltenzähleranzeige für alle Arbeitsdateien ausgeschaltet.

Die Spaltenzähleranzeige wird für eine *Arbeitsdatei* ein- oder ausgeschaltet. Wenn die Arbeitsdatei gleichzeitig in mehreren Datenfenstern am Bildschirm angezeigt wird, wirkt die Anweisung daher in beiden Datenfenstern.

Ist das Datenfenster zu klein, um neben der Spaltenzähleranzeige und einer evtl. anzuzeigenden Informationszeile oder Tabulatorzeile noch mindestens eine Datenzeile anzuzeigen, wird ein Teil der Anzeigen ausgeblendet. Das Datenfenster muss dann entsprechend vergrößert werden. Dabei werden die ausgeblendeten Anzeigen in der Reihenfolge Informationszeile, Spaltenzähleranzeige und Tabulatoranzeige wieder eingeblendet.

Im `EDIT-LONG`-Modus (siehe Anweisung @EDIT) wird der Spaltenzähler nicht ausgegeben, im HEX-Modus hat die Anweisung @SCALE keine Wirkung. In beiden Fällen wird die Einstellung jedoch wirksam, sobald der entsprechende Modus verlassen wird.

Anstelle von @SCALE kann mit der gleichen Funktionalität auch die Anweisung @PAR SCALE benutzt werden. Zusätzlich lässt sich @PAR SCALE gezielt für eine Arbeitsdatei oder global für alle Arbeitsdateien benutzen und ist auch im L-Modus und damit in EDT-Prozeduren erlaubt.

Beispiel

```
   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 ···············································************************
                                                                              :
                                                                              :
                                                                              :
@scale on·····················································0001.00:00001(00)
```

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 ···············································************************
```

9.103 @SDFTEST – Syntaxprüfung durch SDF

Mit der Anweisung @SDFTEST kann geprüft werden, ob ein Zeilenbereich syntaktisch korrekte SDF-Kommandos bzw. syntaktisch korrekte SDF-Anweisungen enthält.

Für die Überprüfung der SDF-Syntax von SDF-Anweisungen kann ein Programmname bekannt gemacht werden.

Ist die SDF-Option `GUIDANCE=MIN|MED|MAX` eingestellt, 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, so wird bei einer fehlerhaften SDF-Syntax die Fehlermeldung `EDT4310` ausgegeben.

Ist die SDF-Syntax korrekt bzw. wurde sie korrigiert, wird der Text in die Arbeitsdatei aufgenommen. Das Format, in dem die Anweisung übernommen wird, wird durch die SDF-Option `LOGGING` bestimmt (siehe Beschreibung des Kommandos `/MODIFY-SDF-OPTIONS` und Beschreibung im Handbuch SDF [6]).

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

Operation	Operanden	F-Modus, L-Modus
@SDFTEST	[lines[,...]] [**PROG**RAM [= progname [**I**NTERNAL | **E**XTERNAL]]]	

lines — Einer oder mehrere Zeilenbereiche, in denen die SDF-Syntax von SDF-Kommandos und ggf. auch von SDF-Anweisungen überprüft werden soll.

Ist `lines` nicht angegeben, soll die SDF-Syntax aller SDF-Kommandos und ggf. auch aller SDF-Anweisungen der Arbeitsdatei überprüft werden.

PROGRAM= — Bewirkt, dass auch die SDF-Syntax von SDF-Anweisungen analysiert wird.

Ist `PROGRAM` nicht angegeben, dann wird nur die SDF-Syntax von SDF-Kommandos analysiert.

progname — Name des Programms, dessen Anweisungen gemäß der SDF-Syntaxdatei-Hierarchie auf Anweisungs-Syntax geprüft werden sollen.

Ist `progname` nicht angegeben, dann wird der Name verwendet, der durch die Anweisung @PAR SDF-PROGRAM voreingestellt wurde. Existiert keine Voreinstellung, wird die Anweisung @SDFTEST mit der Meldung `EDT5320` zurückgewiesen. Ist der Programmname in der aktuellen SDF-Syntaxdatei-Hierarchie nicht bekannt, wird die Anweisung @SDFTEST mit der Meldung `EDT5321` zurückgewiesen.

INTERNAL Der Programmname ist der maximal 8-stellige interne Name. Der interne Name kann mit SDF-A ermittelt werden, falls er nicht dem Namen des Programms entspricht.

EXTERNAL Der Programmname ist der maximal 30-stellige externe Name (z.B. LMS, SDF-A oder HSMS).

Wird als Namenstyp weder INTERNAL noch EXTERNAL angegeben, wird der Namenstyp verwendet, der durch eine Anweisung @PAR SDF-NAME-TYPE voreingestellt ist. Nach dem Starten des EDT ist der Namenstyp INTERNAL voreingestellt.

Der EDT unterscheidet 3 Arten von Satzinhalten:

1. Sätze, 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 ist durch den aktuellen Benutzer und die aktuelle Umgebung bestimmt.

2. Sätze, die mit '//' beginnen:

   Diese werden an SDF zur Anweisungsüberprüfung übergeben, falls PROGRAM angegeben wurde.

3. sonstige Datenzeilen:

   Sätze, die weder mit '/' noch mit '//' beginnen, werden ignoriert.

Zeilen, die mit '/' beginnen und als letztes Zeichen ein Fortsetzungszeichen ('-') enthalten, werden mit der Folgezeile, sofern diese ebenfalls mit '/' beginnt, verkettet und bei der Bearbeitung der Anweisung @SDFTEST gemeinsam an SDF übergeben. Die Fortsetzungszeilen müssen in keinem der angegebenen Zeilenbereiche enthalten sein. Es genügt, wenn die erste Zeile in einem der angegebenen Zeilenbereiche enthalten ist. Ist PROGRAM angegeben, trifft diese Vorgehensweise auch auf diejenigen Zeilen zu, die mit '//' beginnen.

Das geprüfte Kommando bzw. die Anweisung überschreibt das alte Kommando bzw. die Anweisung inklusive aller Folgezeilen in der Arbeitsdatei. Wenn das Kommando bzw. die Anweisung bei der Prüfung durch SDF verändert wurde (z.B. weil in einem vorangegangenen /MODIFY-SDF-OPTIONS-Kommando LOGGING=INVARIANT eingestellt wurde), werden die betroffenen Zeilen neu formatiert und evtl. in mehrere Fortsetzungszeilen aufgeteilt. Das Fortsetzungszeichen wird in der 72. Spalte gesetzt. Wenn nötig, werden die nachfolgenden Zeilen umnummeriert. Die Zeilennummernvergabe erfolgt dabei nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Können die von SDF erzeugten Zeilen nicht eingefügt werden, wird die Anweisung mit der Meldung EDT5364 bzw. EDT5365 abgebrochen.

Im F-Modus wird nach Abschluss aller Prüfungen die Meldung EDT0285 ausgegeben, wenn keine Fehler auftraten oder wenn nach Fehlern die Verarbeitung fortgesetzt wurde. Im L-Modus werden für Anweisungen (nicht Kommandos) die SDF-Ausgaben nach SYSOUT ausgegeben.

Enthält ein geprüftes Kommando bzw. eine Anweisung Fehler und wird kein erfolgreicher Korrektur-Dialog geführt, so gibt der EDT die Meldung EDT4310 aus. Im Dialogbetrieb wird außerdem abgefragt, ob die Prüfung fortgesetzt werden soll:

```
%  EDT4310 SDF: SYNTAX ERROR IN LINE (&00)
%  EDT0911 CONTINUE PROCESSING? REPLY(Y=YES; N=NO)?
```

Wird die Meldung im Dialogbetrieb mit N beantwortet, so wird die @SDFTEST-Anweisung mit der Meldung EDT5324 abgebrochen und im F-Modus die fehlerhafte Zeile an der obersten Fensterposition angezeigt. Wird dagegen die Meldung mit Y beantwortet, so wird mit der Syntaxkontrolle bei der nächsten noch nicht untersuchten Zeile fortgeführt. Im Stapelbetrieb wird die Syntaxkontrolle immer fortgesetzt.

Hat die Arbeitsdatei mit den zu prüfenden Zeilen einen anderen Zeichensatz als EDF03IRV muss man Besonderheiten beim Umgang mit SDF beachten, insbesondere sind Zeichen die nicht zum EBCDIC-Kern gehören natürlich nur in Literalen oder Kommentaren zulässig. Darüber hinaus führt SDF einen Korrekturdialog immer in dem mit /MODIFY-TERMINAL-OPTIONS eingestellten Zeichensatz und interpretiert die ihm übergebenen Byte-Sequenzen auch immer in diesem Zeichensatz.

Daher konvertiert der EDT die Anweisungen oder Kommandos vor Übergabe an SDF in den mit /MODIFY-TERMINAL-OPTIONS eingestellten Zeichensatz, falls der aktuell eingestellte GUIDANCE-MODE einen Korrekturdialog ermöglicht. Gelingt das nicht wird die @SDFTEST-Anweisung mit der Meldung EDT5327 abgebrochen.

Ist kein Korrekturdialog möglich, konvertiert der EDT nach anderen (großzügigeren) Regeln. Hat die Arbeitsdatei einen EBCDIC-Zeichensatz, wird dieser ohne Konvertierung benutzt. Hat die Arbeitsdatei einen ISO-Zeichensatz, wird der entsprechende EBCDIC-Referenzzeichensatz verwendet. In allen anderen Fällen verwendet der EDT den Zeichensatz UTFE. Ist die Konvertierung nicht möglich, wird die @SDFTEST-Anweisung mit der Meldung EDT5453 abgebrochen.

Wenn SDF Daten zurückliefert, werden diese wieder in den Zeichensatz der Arbeitsdatei umgewandelt. Ist das nicht möglich, wird die @SDFTEST-Anweisung mit der Meldung EDT5453 abgebrochen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Hinweis

Bei GUIDANCE=EXPERT werden von SDF gemeldete Fehler von EDT nur in Form der Meldung EDT4310 angezeigt. Zur genaueren Fehleranalyse empfiehlt es sich GUIDANCE=MIN, MED oder MAX einzustellen.

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.

Ist mit /MODIFY-TERMINAL-OPTIONS der Zeichensatz UTFE eingestellt, ist das Bildschirmlayout beim Korrekturdialog von SDF verschoben, wenn Zeichen vorkommen, die außerhalb von EDF03IRV liegen. Das liegt daran, dass SDF derzeit Unicode nicht unterstützt, stellt aber normalerweise keine funktionelle Einschränkung dar.

Ein Kommando oder eine Anweisung darf sowohl bei der Ein- als auch bei der Ausgabe eine maximale Länge von 16379 Byte nicht übersteigen. Andernfalls wird die Anweisung @SDFTEST mit der Meldung EDT5325 bzw. EDT5326 abgebrochen.

9.104 @SEARCH-OPTION – Voreinstellung für Suchen mit @ON

Mit der Anweisung @SEARCH-OPTION kann zum einen eingestellt werden, ob bei der Suche nach Zeichenfolgen mit der @ON-Anweisung Groß- und Kleinbuchstaben im Suchbegriff unterschieden werden sollen und zum anderen kann global ein Spaltenbereich definiert werden, auf den die Suche eingeschränkt wird, wenn in der @ON-Anweisung keine explizite Angabe zum Spaltenbereich erfolgt.

Operation	Operanden	F-Modus, L-Modus
@SEARCH-OPTION	{ **CASELESS**-SEARCH [= { **ON** / **OF**F }] / **COL**UMN-RANGE [= cols] } [,...]	

CASELESS-SEARCH=

Der Operand legt fest, ob bei der Suche mit @ON Groß- und Kleinbuchstaben im Suchbegriff unterschieden werden sollen.

ON — Es wird nicht unterschieden, ob die Zeichen im Suchbegriff mit dem gefundenen Text bezüglich der Groß- oder Kleinschreibung übereinstimmen, d.h. bei der Suche nach `'string'` werden auch die Zeichenfolgen `'String'`, `'STRING'` oder `'StrIng'` als Treffer erkannt (siehe auch Abschnitt „Suchen mit @ON“ auf Seite 81).

OFF — Die Groß- oder Kleinschreibung eines Zeichens wird bei der Suche beachtet. Dies ist die Voreinstellung beim Start des EDT.

COLUMN-RANGE=

Der Operand definiert global einen Spaltenbereich, auf den die Suche eingeschränkt wird, wenn in der @ON-Anweisung keine explizite Angabe zum Spaltenbereich erfolgt.

cols — Zusammenhängender Spaltenbereich, auf den die Suche mit der Anweisung @ON eingeschränkt werden soll. Enthält die Bereichsangabe nur eine einzelne Spaltenangabe, so wird damit der Bereich von dieser Spalte bis 32768 spezifiziert. Ist die erste Spaltenangabe größer als die zweite, wird die Anweisung mit der Meldung `EDT3922` abgewiesen.

Wird kein Spaltenbereich angegeben, wird die Einstellung auf den beim Start des EDT eingestellten Bereich `1-32768` zurückgesetzt.

Der EDT verwendet zur Zuordnung von Klein- zu Großbuchstaben die Systemkomponente XHCS. Welche Zeichen als Groß-Klein-Paare behandelt werden, hängt daher von der Definition der jeweiligen Zeichensatz-Attribute in XHCS ab.

Die Option `CASELESS-SEARCH=ON` wirkt unabhängig von der mit @PAR LOWER getroffenen Einstellung. Im F-Modus mit @PAR LOWER=OFF werden daher evtl. Treffer angezeigt, bei denen die gefundenen Zeichen als Schmierzeichen dargestellt sind.

9.105 @SEPARATE – Zeile umbrechen

Mit der Anweisung @SEPARATE werden die angegebenen Zeilen in mehrere Zeilen umgebrochen. Die Umbruchstelle wird durch ein Satztrennzeichen oder durch eine Spaltenposition angegeben.

Operation	Operanden	F-Modus, L-Modus
@SEPARATE	[lines [,...]] [**AT** { strchar | col }]	

lines — Einer oder mehrere Zeilenbereiche, deren Zeilen umgebrochen werden sollen. Fehlt der Operand `lines`, so werden alle Zeilen der Datei bearbeitet.

AT — Der Operand leitet die Definition der Umbruchstelle ein.

Fehlt der Operand `AT`, so bestimmt das mit der Anweisung @PAR SEPARATOR voreingestellte Satztrennzeichen den Umbruch (siehe Anweisung @PAR SEPARATOR). Wenn kein Satztrennzeichen voreingestellt ist, wird die Fehlermeldung `EDT4952` ausgegeben.

strchar — Der Operand legt das Satztrennzeichen für den Umbruch fest. Er besteht aus einem beliebigen Zeichen, das in Hochkommas anzugeben ist. Das Zeichen kann auch in Form einer Ersatzdarstellung für Unicode-Zeichen angegeben werden.

col — Der Operand gibt die Nummer der Spalte an, bei der umgebrochen werden soll.

Wird der Umbruch mit Hilfe eines Satztrennzeichens durchgeführt und befinden sich in einer Zeile des angegebenen Zeilenbereichs keine Satztrennzeichen, so wird die Zeile nicht umgebrochen.

Befinden sich in einer Zeile des angegebenen Zeilenbereichs dagegen ein oder mehrere Satztrennzeichen, so wird die Zeile von links nach rechts nach dem ersten Auftreten des Satztrennzeichens durchsucht.

Alle Zeichen vor dem ersten Satztrennzeichen verbleiben in der ursprünglichen Zeile, während alle Zeichen nach dem Satztrennzeichen (auch noch weitere eventuell auftretende Satztrennzeichen) als neue Zeile in die Arbeitsdatei eingefügt werden.
Das Satztrennzeichen, bei dem umgebrochen wurde, wird entfernt, d.h., es kommt weder in der ursprünglichen Zeile noch in der neu eingefügten Zeile vor.

Enthält die neu eingefügte Zeile noch weitere Satztrennzeichen, so wird die soeben beschriebene Vorgehensweise auf diese Zeile erneut angewandt. Dies wird solange fortgesetzt, bis eine neu eingefügte Zeile kein Satztrennzeichen mehr enthält.

Treten in der Zeile mehrere Satztrennzeichen hintereinander auf bzw. beginnt oder endet die Zeile mit einem oder mehreren Satztrennzeichen, so werden Leersätze (Satzlänge=0) erzeugt.

Wird die Umbruchstelle durch eine Spaltennummer festgelegt, so werden alle Zeichen ab dieser Position (inklusive dem Zeichen in der angegebenen Spalte) von der ursprünglichen Zeile abgetrennt und als neue Zeile in die Arbeitsdatei eingefügt. Hat die neu eingefügte Zeile mindestens noch genau so viel oder mehr Spalten, als in der Anweisung durch den Operanden col angegeben, so wird auch diese Zeile an der Spalte `col` umgebrochen. Dieser Vorgang wird solange fortgesetzt, bis eine neu eingefügte Zeile weniger Spalten hat, als durch den Operanden `col` angegeben.

Hat die ursprüngliche Zeile bereits weniger Spalten als durch den Operand col angegeben oder wird in der Anweisung `col=1` angegeben, wird die Zeile nicht verändert.

Die Nummerierung der beim Umbruch neu entstehenden Zeilen geschieht nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Beispiel 1

Eine Druckliste soll eine kleinere Breite erhalten:

```
@SEPARATE 5-100 AT 41
```

Die Zeilen 5 bis 100 werden auf eine Länge von 40 Zeichen gekürzt. Die Zeilenreste werden hinter jede Zeile in die Datei eingefügt.

Beispiel 2

In Datensätzen sind Zeilenvorschub-Zeichen (`= U'000A'` in `UTF16`) enthalten, die ausgewertet werden sollen (es wird für dieses Beispiel angenommen, dass als Fluchtsymbol für die Ersatzdarstellung von Unicode-Zeichen mit der Anweisung @PAR ESCAPE-CHARACTER das Zeichen `%` vereinbart wurde):

```
@SEPARATE & AT '%U000A'
```

9.106 @SEQUENCE (Format 1) – Zeilen nummerieren

Die Anweisung @SEQUENCE (Format 1) bewirkt, dass der EDT in jede Zeile eines zusammenhängenden Zeilenbereichs eine Zahl schreibt.
In die erste Zeile des Zeilenbereichs wird eine vorgegebene, maximal 8-stellige Zahl geschrieben (eventuell mit führenden Nullen), die auch die Stellenzahl aller folgenden Zahlen festlegt. Alle folgenden Zahlen sind jeweils die Summe aus der vorhergehenden Zahl und einer vorgegebenen Schrittweite. Würde dabei eine Zahl mit einer größeren Stellenzahl als die der Startzahl entstehen, werden von rechts nur so viele Stellen genommen, wie die Startzahl hat.

Diese Anweisung überschreibt den etwaigen Inhalt der Spalten, in die die Zahlen geschrieben werden.

Operation	Operanden	F-Modus, L-Modus
@SEQUENCE	[{lines | svars}] [: [col] [: [n1] [(n2)]]]	

lines — In jede Zeile des angegebenen Zeilenbereichs schreibt der EDT eine Zahl.

svars — In jede Zeichenfolgevariable des angegebenen Bereichs von Zeichenfolgevariablen schreibt der EDT eine Zahl.

col — Der Operand gibt die Spalte an, in der die erste Ziffer der zu schreibenden Zahl stehen soll. Hat eine Zeile des angegebenen Zeilenbereichs weniger Spalten, als durch den Operand `col` spezifiziert, so werden die Spalten zwischen dem bisherigen Zeilenende und der Spalte `col` mit Leerzeichen aufgefüllt.

Fehlt der Operand `col`, schreibt der EDT die erste Ziffer in Spalte 73.

n1 — Der Operand gibt die Ganzzahl an, die der EDT in die erste Zeile des betrachteten Zeilenbereichs als Dezimalzahl schreibt. Der Operand `n1` darf maximal 8 Stellen haben (eventuell mit führenden Nullen). Die Zahlen, die in die folgenden Zeilen geschrieben werden, haben die gleiche Stellenanzahl.

Fehlt der Operand `n1`, schreibt der EDT in die erste betrachtete Zeile die Zahl `00000100`.

n2 Der Operand gibt die ganzzahlige Schrittweite zur Bildung der folgenden Zahlen an. Diese sind jeweils die Summe aus der vorhergehenden Zahl und der Schrittweite, wobei von rechts nur so viele Stellen genommen werden, wie die Startzahl `n1` aufweist.

Fehlt der Operand `n2`, nimmt der EDT als Schrittweite den Wert `100`.

Fehlt sowohl der Operand `lines` als auch der Operand `svars`, schreibt der EDT in jede Zeile der aktuellen Arbeitsdatei eine Zahl.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Hinweis

Ob die durch die Anweisung @SEQUENCE (Format 1) erzeugte Zahlenfolge aufsteigend, absteigend oder konstant ist, hängt von der Wahl des Startwerts und der Schrittweite ab. So erzeugt der Startwert `0100` und die Schrittweite `100` die aufsteigende Zahlenfolge `0100`, `0200`, `0300`, usw., zumindest bis zum Wert `9900`.
Danach springt der Wert auf `0000` und es geht danach wieder mit `0100`, `0200` usw. weiter.
Wählt man als Startwert die Zahl `999` und als Schrittweite die Zahl `998`, entsteht eine absteigende Folge von Zahlen und zwar `999`, `997`, `995`, usw. Nachdem die Zahl `001` erreicht wurde ist die nächste Zahl wieder `999` und Zahlenfolge beginnt von vorn.

Das gleiche Ergebnis erhält man auch, wenn man im vorangehenden Beispiel die Schrittweite z.B. auf `3998` setzt, da die führende `3` bei jeder neuen Zahlenbildung wegfällt.
Eine alternierende Zahlenfolge erhält man z.B. durch den Startwert `3` und die Schrittweite `5: 3`, `8`, `3`, `8`, usw. Eine konstante Zahlenfolge erhält man am einfachsten durch die Wahl der Schrittweite `0`.

9.107 @SEQUENCE (Format 2) – Zeilennummern übernehmen

Die Anweisung @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. Der EDT überschreibt dabei den etwaigen Inhalt der 8 Spalten, in die er die Zeilennummer schreibt.

Operation	**Operanden**	**F-Modus, L-Modus**
@SEQUENCE	[lines] : [col] : **L**INE	

lines — In jede Zeile des angegebenen Zeilenbereichs schreibt der EDT die zugehörige Zeilennummer. Fehlt der Operand `lines`, schreibt der EDT in jede Zeile der aktuellen Arbeitsdatei die zugehörige Zeilennummer.

col — Der Operand gibt die Spalte an, in der die erste Ziffer der zugehörigen Zeilennummer stehen soll. Hat eine Zeile des angegebenen Zeilenbereichs weniger Spalten, als durch den Operand `col` spezifiziert, so werden die Spalten zwischen dem bisherigen Zeilenende und der Spalte `col` mit Leerzeichen aufgefüllt.

Fehlt der Operand `col`, schreibt der EDT die erste Ziffer der Zeilennummer in Spalte 73.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Beispiel

```
   0.00 AUF<·····························································
   1.11 DIE<·····························································
  88.76 REIHENFOLGE<·····················································
  88.76 KOMMT<···························································
5555.00 ES<·····························································
9876.54 AN<·····························································
9877.54 ·····························································
sequence :20: line·····································0000.00:00001(00)
```

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.

9.108 @SEQUENCE (Format 3) – Zeilennummern überprüfen

Die Anweisung @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 im Fall von Unicode-Zeichensätzen gemäß ihrer `UTF16`-Codierung, ansonsten gemäß der dem Zeichensatz der Arbeitsdatei entsprechenden Codierung als Dualzahl. Liegt die zu untersuchende Spalte rechts vom Zeilenende, nimmt der EDT als Spalteninhalt ein Leerzeichen an.
Der EDT prüft, ob die ermittelten Dualzahlen eine aufsteigende Folge bilden. Er gibt alle Zeilen aus, in denen er eine Dualzahl ermittelt, die gleich oder kleiner ist, als die aus der vorhergehenden Zeile. Im Dialogbetrieb erfolgt die Ausgabe nach `SYSOUT`, im Stapelbetrieb nach `SYSLST`.

Operation	Operanden	F-Modus, L-Modus
@SEQUENCE	[{ lines | svars }] : [col] : **C**HECK [int]	

lines — In jeder Zeile des angegebenen Zeilenbereichs wird der angegebene Spaltenbereich untersucht.

svars — In jeder Zeichenfolgevariablen des angegebenen Bereichs von Zeichenfolgevariablen wird der angegebene Spaltenbereich untersucht.

col — Der Operand gibt die Spalte an, in der das erste zu überprüfende Zeichen steht. Liegt der zu überprüfende Spaltenbereich ganz oder teilweise hinter dem Zeilenende, wird für jede hinter dem Zeilenende liegende Spalte ein Leerzeichen angenommen.

Fehlt der Operand `col`, beginnt der EDT die Überprüfung in Spalte 73.

int — Der Operand gibt an, wie viele Spalten (`1..8`) betrachtet werden sollen. Fehlt der Operand `int`, betrachtet der EDT 8 Spalten.

Wird ein Bereich von Zeichenfolgevariablen angegeben, muss für alle Zeichenfolgevariablen entweder ein Unicode-Zeichensatz oder ein 7-Bit- bzw. 8-Bit-Zeichensatz eingestellt sein. Ist dies nicht der Fall, wird die Fehlermeldung `EDT5473` ausgegeben und die Abarbeitung der Anweisung abgebrochen.
Fehlt sowohl der Operand `lines` als auch der Operand `svars`, überprüft der EDT jede Zeile der aktuellen Arbeitsdatei.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

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·····································0001.00:00001(00)
```

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·····································0001.00:00001(00)
```

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.

9.109 @SET (Format 1) – Versorgen von Ganzzahlvariablen mit Werten

Mit diesem Format der Anweisung @SET wird einer Ganzzahlvariablen ein Wert zugewiesen. Dieser Wert kann als Ergebnis eines Ausdrucks, durch Umwandlung einer abdruckbaren Zahl in eine Ganzzahl, durch Umwandlung des Inhalts einer Zeilennummervariablen in eine Ganzzahl, aus der Länge einer Zeile oder aus dem Binärwert einer Zeichenfolge entstehen.

Operation	**Operanden**	**F-Modus, L-Modus**
@SET	ivar = { intex | lvar | **L**ENGTH line | **L**ENGTH svarex | **S**UBSTR string | **ST**RING string [,**C**ODE = name] }	

ivar Ganzzahlvariable (`#I0..#I20`), der ein Wert zugewiesen werden soll.

intex Ganzzahliger Ausdruck. Wird bei einer Rechenoperation der negative oder positive Maximalwert (-2^{31}, $2^{31}-1$) überschritten, wird die Anweisung mit der Meldung `EDT4946` abgewiesen.

lvar Zeilennummervariable (`#L0..#L20`), deren Wert der Ganzzahlvariablen zugewiesen werden soll. Bei der Konvertierung in eine Ganzzahl wird der Inhalt der Zeilennummervariablen mit `10000` multipliziert und zugewiesen.

LENGTH line Zeilennummer einer Zeile, deren Länge der Ganzzahlvariablen zugewiesen werden soll. Ist die Zeile leer, wird der Wert `0` zugewiesen. Existiert die Zeile nicht, wird der Ganzzahlvariablen aus Kompatibilitätsgründen ebenfalls der Wert `0` zugewiesen.

LENGTH svarex
Zeichenfolgevariable, deren Länge der Ganzzahlvariablen zugewiesen werden soll.

SUBSTR string

Zeichenfolge, die eine Ganzzahl angibt, die der Ganzzahlvariablen zugewiesen werden soll. Ein in der Zeichenfolge enthaltenes Plus- oder Minuszeichen wird bei der Konvertierung berücksichtigt. Fehlt das Vorzeichen, wird + angenommen. Enthält die Zeichenfolge Leerzeichen, werden diese bei der Umwandlung unterdrückt.

Ist die Zeichenfolge keine Ganzzahl, wird die Anweisung mit der Fehlermeldung EDT5477 abgewiesen. Liegt die Ganzzahl nicht im zulässigen Bereich ($-2^{31}, 2^{31}-1$) wird die Anweisung mit der Meldung EDT4946 abgewiesen.
Die leere Zeichenfolge ist nicht erlaubt und führt zum Fehler EDT3907.

STRING string

Zeichenfolge. Der Binärwert der ersten 4 Byte der Zeichenfolge wird der Ganzzahlvariablen zugewiesen. Enthält die Zeichenfolge weniger als 4 Byte, wird linksbündig mit Nullen aufgefüllt.
Die leere Zeichenfolge ist nicht erlaubt und führt zum Fehler EDT3907.

name Zeichensatz, in dem die Zeichenfolge interpretiert werden soll. Sie wird vor der Zuweisung in diesen Zeichensatz konvertiert. Die ersten 4 Byte werden dann ohne Rücksicht auf Zeichengrenzen zugewiesen.
Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Anweisung mit der Fehlermeldung EDT5453 abgewiesen.

Fehlt der Operand, wird der Zeichensatz der angegebenen Zeichenfolge (je nach Quelle) verwendet.

Bei dieser Anweisung darf der Anweisungsname ganz entfallen. Im F-Modus darf zusätzlich auch das Anweisungssymbol fehlen.

Beispiel

```
   1.     @SET #I0 = SUBSTR '123' ---------------------------------------- (1)
   1.     @SET #L0 = 1.01 ------------------------------------------------ (2)
   1.     @SET #I1 = #L0 ------------------------------------------------- (3)
   1.     @CREATE 1 'AB' ------------------------------------------------- (4)
   1.     @SET #I2 = LENGTH 1 -------------------------------------------- (5)
   1.     @SET #I3 = 2124 + #I0 + #I1 - #I2 ------------------------------ (6)
   1.     @SET #I4 = STRING '123' ---------------------------------------- (7)
   1.     @SET #I5 = STRING 'A',CODE=UTF16 ------------------------------- (8)
   1.     @STATUS = I ---------------------------------------------------- (9)
#I00= 0000000123 #I01= 0000010100 #I02= 0000000002
#I03= 0000012345 #I04= 0015856371 #I05= 0000000065
#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
```

(1) Der Ganzzahlvariablen `#I0` wird der Wert `123` zugewiesen.

(2) Der Zeilennummervariablen `#L0` wird der Wert `0001.0100` zugewiesen.

(3) Der Ganzzahlvariablen `#I1` wird der Wert `10100`, Zeilennummer `1.01 * 10000`, zugewiesen.

(4) Die Zeile 1 wird mit dem Inhalt `AB` erzeugt.

(5) Der Ganzzahlvariablen `#I2` wird der Wert `2`, die Länge der Zeile 1, zugewiesen.

(6) Der Ganzzahlvariablen `#I3` wird der Wert des Ausdrucks zugewiesen.

(7) Der Ganzzahlvariablen `#I4` wird der Wert `X'00F1F2F3' = 15856371` zugewiesen.

(8) Der Ganzzahlvariablen `#I5` wird der Wert zugewiesen, der der Unicode-Codeposition des Zeichens `'A'` entspricht.

(9) Der Inhalt der Ganzzahlvariablen wird ausgegeben.

9.110 @SET (Format 2) – Versorgen von Zeichenfolgevariablen mit Werten

Mit diesem Format der Anweisung @SET wird einer Zeichenfolgevariablen ein Wert zugewiesen. Dieser Wert kann zum einen der Binärwert einer Ganzzahlvariablen, einer Zeilennummer oder der Name einer Zeichenfolgevariablen sein. Zum zweiten kann eine Zeichenfolge zugewiesen werden. In beiden Fällen wird die Zeichenfolgevariable neu angelegt.

Operation	**Operanden**	**F-Modus, L-Modus**
@SET	svarex { = **I**NTERNAL { ivar / line / svar } / = string [,**C**ODE=name] }	

svarex Zeichenfolgevariable (#S0..#S20), der ein Wert zugewiesen werden soll.

INTERNAL ivar
Ganzzahlvariable (#I0..#I20), deren Inhalt der Zeichenfolgevariablen binär zugewiesen werden soll. Das Ergebnis ist in der Regel nicht abdruckbar. Der Zeichensatz ist EDF041.

INTERNAL line
Zeilennummer, die der Zeichenfolgevariablen als Binärwert zugewiesen werden soll. Dabei werden die 8 Ziffern einer Zeilennummer jeweils binär einem Halbbyte zugewiesen. Das Ergebnis ist in der Regel nicht abdruckbar. Der Zeichensatz ist EDF041.

INTERNAL svar
Zeichenfolgevariable (#S0..#S20), deren Name der Zeichenfolgevariablen als Wert zugewiesen werden soll. Der Zeichensatz ist EDF041.

string Zeichenfolge, die der Zeichenfolgevariablen zugewiesen werden soll.

name Zeichensatz, den die Zeichenfolgevariable erhalten soll. Die Zeichenfolge string wird vor der Zuweisung in diesen Zeichensatz konvertiert. Enthält sie Zeichen, die im Ziel-Zeichensatz ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Anweisung mit der Fehlermeldung EDT5453 abgewiesen.
Fehlt der Operand, erhält die Zeichenfolgevariable den Zeichensatz der zuzuweisenden Zeichenfolge.

Bei dieser Anweisung darf der Anweisungsname ganz entfallen. Im F-Modus darf zusätzlich auch das Anweisungssymbol fehlen.

Beispiel

```
1.     @SET #I0 = -2122153084 ---------------------------------------- (1)
1.     @SET #S0 = INTERNAL #I0 --------------------------------------- (2)
1.     @SET #S1 = INTERNAL 4081.4082 --------------------------------- (3)
1.     @SET #S2 = INTERNAL #S0 --------------------------------------- (4)
1.     @SET #S3 = 'ABC' ---------------------------------------------- (5)
1.     @PRINT #S00.-#S03 --------------------------------------------- (6)
  #S00 abcd
  #S01  a b
  #S02 #S00
  #S03 ABC
```

(1) Der Ganzzahlvariablen #I0 wird der Wert -2122153084 zugewiesen.

(2) Der Zeichenfolgevariablen #S0 wird der Wert X'81828384' zugewiesen.

(3) Der Zeichenfolgevariablen #S1 wird der Wert X'40814082' zugewiesen.

(4) Der Zeichenfolgevariablen #S2 wird der Wert '#S00' zugewiesen.

(5) Der Zeichenfolgevariablen #S3 wird der Wert 'ABC' zugewiesen.

(6) Die Zeichenfolgevariablen #S00-#S03 werden ausgegeben.

9.111 @SET (Format 3) – Versorgen von Zeilennummervariablen mit Werten

Mit diesem Format der Anweisung @SET wird einer Zeilennummervariablen ein Wert zugewiesen. Dieser Wert kann entstehen aus: der Angabe einer Zeilennummer, dem Wert einer Ganzzahlvariablen, der Angabe einer Zeilennummer als Zeichenfolge oder dem Binärwert der ersten 4 Byte einer Zeichenfolge.

Operation	Operanden	F-Modus, L-Modus
@SET	lvar = { line | ivar | **S**UBSTR string | **ST**RING string[,**C**ODE=name] }	

lvar Zeilennummervariable (#L0..#L20), der ein Wert zugewiesen werden soll.

line Zeilennummer, die der Zeilennummervariablen zugewiesen werden soll.

ivar Ganzzahlvariable (#I0..#I20), deren Inhalt der Zeilennummervariablen in umgewandelter Form zugewiesen werden soll. Der zulässige Wertebereich für ivar beträgt dabei 1 bis 99999999. Dies führt zu den Zeilennummern 0.0001 bis 9999.9999. Bei der Konvertierung wird also der Wert der Ganzzahl durch 10000 dividiert und der entsprechende Wert der Zeilennummervariablen zugewiesen. Liegt der Wert der Ganzzahlvariablen nicht im gültigen Bereich, wird die Anweisung mit der Meldung EDT5475 abgewiesen.

SUBSTR string

Zeichenfolge, die nach Umwandlung in eine Zeilennummer der Zeilennummervariablen zugewiesen werden soll. Stellt die Zeichenfolge keine gültige Zeilennummer dar, wird die Anweisung mit der Meldung EDT5477 abgewiesen. Enthält die Zeichenfolge Leerzeichen, werden diese bei der Umwandlung unterdrückt.
Die leere Zeichenfolge ist nicht erlaubt und führt zum Fehler EDT3907.

STRING string

Zeichenfolge, deren Binärwert als Zeilennummer interpretiert und der Zeilennummervariablen zugewiesen wird. Dabei werden die 8 Halbbytes, der ersten 4 Bytes der Zeichenfolge, jeweils als Dezimalziffer interpretiert.

Enthält die Zeichenfolge nach dem Konvertieren weniger als 4 Byte, wird linksbündig mit Nullen aufgefüllt.

Entspricht der Binärwert eines Halbbytes in den ersten 4 Bytes der Zeichenfolge nicht einer Dezimalziffer `0..9`, wird die Anweisung mit der Meldung `EDT4928` abgewiesen.
Die leere Zeichenfolge ist nicht erlaubt und führt zum Fehler `EDT3907`.

name — Zeichensatz, in dem die Zeichenfolge interpretiert werden soll. Sie wird vor der Zuweisung in diesen Zeichensatz konvertiert. Die ersten 4 Byte werden dann ohne Rücksicht auf Zeichengrenzen zugewiesen.

Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Anweisung mit der Fehlermeldung `EDT5453` abgewiesen.

Fehlt der Operand, wird der Zeichensatz der angegebenen Zeichenfolge (je nach Quelle) verwendet.

Bei dieser Anweisung darf der Anweisungsname ganz entfallen. Im F-Modus darf zusätzlich auch das Anweisungssymbol fehlen.

Beispiel

```
   1.     @SET #L0 = 1.01 ------------------------------------------------ (1)
   1.     @SET #I0 = #L0 ------------------------------------------------- (2)
   1.     @SET #L1 = #I0 ------------------------------------------------- (3)
   1.     @SET #L2 = SUBSTR '1.01' --------------------------------------- (4)
   1.     @SET #L3 = STRING X'00010100' ---------------------------------- (5)
   1.     @STATUS = L ---------------------------------------------------- (6)
#L00=   1.0100    #L01=   1.0100    #L02=   1.0100
#L03=   1.0100    #L04=   0.0000    #L05=   0.0000
#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) Der Zeilennummervariablen `#L0` wird der Wert `0001.0100` zugewiesen.

(2) Der Ganzzahlvariablen `#I1` wird der Wert `10100`, Zeilennummer `1.01 * 10000`, zugewiesen.

(3) Der Zeilennummervariablen `#L1` wird der Wert `0001.0100` zugewiesen.

(4) Der Zeilennummervariablen `#L2` wird der Wert `0001.0100` zugewiesen.

(5) Der Zeilennummervariablen `#L3` wird der Wert `0001.0100` zugewiesen.

(6) Der Inhalt der Zeilennummervariablen wird ausgegeben.

9.112 @SET (Format 4) – Werte von Variablen ablegen

Mit diesem Format der Anweisung @SET kann der Inhalt einer Ganzzahlvariablen, der Name einer Zeichenfolgevariablen oder der Inhalt einer Zeilennummervariablen ab einer bestimmten Spalte in abdruckbarer Form in eine Arbeitsdateizeile oder eine Zeichenfolgevariable eingefügt werden.

Operation	**Operanden**	**F-Modus, L-Modus**
@SET	{ svarex | lvar } [,col] = **C**HAR { ivar | svar | lvar1 }	

svarex Zeichenfolgevariable (#S0..#S20), in die ein Wert eingefügt werden soll. Vorhandene Zeichen in den betroffenen Positionen werden überschrieben.

lvar Zeilennummervariable (#L0..#L20), die die Zeile angibt, in die ein Wert geschrieben werden soll. Existiert die Zeile noch nicht, wird sie neu angelegt. Vorhandene Zeichen in den betroffenen Positionen werden überschrieben.

col Spalte, ab der in die Zeile oder Zeichenfolgevariablen geschrieben werden soll. Der Standardwert von col ist 1. Liegt col hinter dem Zeilenende, wird die Zeile bis zur Spalte col mit Leerzeichen aufgefüllt.

CHAR ivar Ganzzahlvariable (#I0..#I20), deren Inhalt in die Zeile ab der Spalte col als Zeichenfolge eingefügt werden soll. 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.

CHAR svar Zeichenfolgevariable (#S00..#S20), deren Name in die angegebene Zeile oder Zeichenfolgevariable eingefügt werden soll.

CHAR lvar1 Zeilennummervariable, deren Wert in abdruckbarer Form in die angegebene Zeile oder Zeichenfolgevariable eingefügt 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.

Wird durch das Einfügen die maximale Länge von 32768 überschritten, wird die Anweisung mit der Meldung EDT5474 abgewiesen.

Wird die Information in eine Zeile eingefügt, hängt der Zeichensatz von der Arbeitsdatei ab. Hat die aktuelle Arbeitsdatei bereits einen Zeichensatz, wird der Wert in diesem Zeichensatz eingefügt. Ist die Arbeitsdatei leer und hat sie den Zeichensatz *NONE, erhält sie vor dem Einfügen den Zeichensatz EDF041.

Wird die Information in eine Zeichenfolgevariable eingefügt, wird sie vor dem Einfügen in den Zeichensatz der Zeichenfolgevariablen konvertiert.

Bei dieser Anweisung darf der Anweisungsname ganz entfallen. Im F-Modus darf zusätzlich auch das Anweisungssymbol fehlen.

Beispiel

```
1.     @SET #L0 = 1 ------------------------------------------------ (01)
1.     @SET #I0 = 123 ---------------------------------------------- (02)
1.     @SET #L0 = CHAR #I0 ----------------------------------------- (03)
1.     @SET #L0 ,13 = CHAR #S0 ------------------------------------- (04)
1.     @SET #L0 ,18 = CHAR #L0 ------------------------------------- (05)
1.     @SET #S0 = CHAR #I0 ----------------------------------------- (06)
1.     @SET #L0 = 47.11 -------------------------------------------- (07)
1.     @SET #S1 = CHAR #L0 ----------------------------------------- (08)
1.     @SET #S2 ,5 = CHAR #S0 -------------------------------------- (09)
1.     @PRINT 1,#S0-#S2 -------------------------------------------- (10)
1.0000  0000000123 #S00    1.0000
  #S00  0000000123
  #S01   47.1100
  #S02     #S00
```

(01) Der Zeilennummervariablen #L0 wird der Wert 0001.0000 zugewiesen.

(02) Der Ganzzahlvariablen #I0 wird der Wert 123 zugewiesen.

(03) In die Zeile 1 wird ab Spalte 1 die Zeichenfolge ' 0000000123' eingefügt.

(04) In die Zeile 1 wird ab Spalte 13 die Zeichenfolge '#S00' eingefügt.

(05) In die Zeile 1 wird ab Spalte 18 die Zeichenfolge '   1.0000' eingefügt.

(06) Der Zeichenfolgevariablen #S0 wird der Wert ' 0000000123' zugewiesen.

(07) Der Zeilennummervariablen #L0 wird der Wert 47.11 zugewiesen.

(08) Der Zeichenfolgevariablen #S1 wird der Wert ' 47.1100' zugewiesen.

(09) Der Zeichenfolgevariablen #S2 wird ab Spalte 5 der Wert '#S00' zugewiesen d.h. ihr Wert ist '   #S00'.

(10) Zeile 1 und die Zeichenfolgevariablen #S0..#S2 werden ausgegeben.

9.113 @SET (Format 5) – Datum und Uhrzeit

Mit diesem Format der Anweisung @SET wird Datum oder Uhrzeit ab einer gewünschten Spalte in einer Zeichenfolgevariablen oder in einer Arbeitsdateizeile abgelegt.

Operation	Operanden	F-Modus, L-Modus
@SET	{ svarex | lvar } [,col] = { **D**ATE [**I**SO[**4**]] | **T**IME }	

svarex — Zeichenfolgevariable (#S0..#S20), in der Datum oder Uhrzeit eingefügt werden sollen. Vorhandene Zeichen in den betroffenen Positionen werden überschrieben.

lvar — Zeilennummervariable (#L0..#L20), die angibt, in welcher Zeile Datum oder Uhrzeit eingefügt werden sollen. Existiert die Zeile noch nicht, wird sie neu angelegt. Vorhandene Zeichen in den betroffenen Positionen werden überschrieben.

col — Spalte, ab der Datum bzw. Uhrzeit abgelegt werden soll. Ist `col` nicht angegeben, wird ab Spalte 1 eingefügt. Liegt `col` hinter dem Zeilenende, wird die Zeile bis zur Spalte `col` mit Leerzeichen aufgefüllt.

DATE — Das aktuelle Datum wird in der angegebenen Zeichenfolgevariablen bzw. Zeile in der gewünschten Form eingefügt. Ist ISO nicht angegeben, wird die Form `mm/dd/yyjjj` verwendet. Durch `mm` wird der Monat, durch `dd` der Tag, durch `yy` das Jahr und durch `jjj` der Jahrestag angegeben.

ISO — gibt an, dass das Datum im Format `yy-mm-ddjjj` ausgegeben werden soll.

ISO4 — gibt an, dass das Datum im Format `yyyy-mm-ddjjj` ausgegeben werden soll.

TIME — Die Uhrzeit wird in der angegebenen Zeichenfolgevariablen bzw. Zeile in der Form `hhmmss` abgelegt. Durch `hh` werden die Stunden, durch `mm` die Minuten und durch `ss` die Sekunden angegeben.

Wird durch das Einfügen die maximale Länge von 32768 überschritten, wird die Anweisung mit der Meldung `EDT5474` abgewiesen.

Wird die Information in eine Zeile eingefügt, hängt der Zeichensatz von der Arbeitsdatei ab. Hat die aktuelle Arbeitsdatei bereits einen Zeichensatz, wird der Wert in diesem Zeichensatz eingefügt. Ist die Arbeitsdatei leer und hat sie den Zeichensatz `*NONE`, erhält sie vor dem Einfügen den Zeichensatz `EDF041`.

Wird die Information in eine Zeichenfolgevariable eingefügt, wird sie vor dem Einfügen in den Zeichensatz der Zeichenfolgevariablen konvertiert.

Bei dieser Anweisung darf der Anweisungsname ganz entfallen. Im F-Modus darf zusätzlich auch das Anweisungssymbol fehlen.

Beispiel

```
1.     @SET #L0 = 1 ------------------------------------------------- (1)
1.     @SET #L0 = DATE ---------------------------------------------- (2)
1.     @SET #L0 ,13 = DATE ISO -------------------------------------- (3)
1.     @SET #S0 = TIME ---------------------------------------------- (4)
1.     @SET #S0 ,13 = DATE ISO4 ------------------------------------- (5)
1.     @PRINT 1 ----------------------------------------------------- (6)
1.0000 02/07/06038 06-02-07038
1.     @PRINT #S00 -------------------------------------------------- (7)
  #S00 165941      2006-02-07038
```

(1) Der Zeilennummervariablen #L0 wird der Wert 0001.0000 zugewiesen.

(2) In die Zeile 1 wird ab Spalte 1 das Datum eingefügt.

(3) In die Zeile 1 wird ab Spalte 13 das Datum im ISO-Format eingefügt.

(4) In die Zeichenfolgevariable #S00 wird ab Spalte 1 die Uhrzeit eingefügt.

(5) In die Zeichenfolgevariable #S00 wird ab Spalte 13 das Datum im ISO4-Format eingefügt.

(6) Zeile 1 wird ausgegeben

(7) Zeichenfolgevariable #S00 wird ausgegeben.

9.114 @SET (Format 6) – Verändern der aktuellen Schrittweite und Zeilennummer

Die Anweisung @SET definiert die aktuelle Zeilennummer und die aktuelle Schrittweite oder es werden frühere Werte für die Zeilennummer und die Schrittweite wieder hergestellt.

Operation	Operanden	F-Modus, L-Modus
@SET	[line [(inc)] [:text]]]	

line — Der Operand gibt die neue aktuelle Zeilennummer an.

inc — Der Operand gibt die neue aktuelle Schrittweite an. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

text — EDT-Anweisung oder eine Dateneingabe, die nach der Festlegung der neuen aktuellen Zeile und Schrittweite ausgeführt bzw. in die neue aktuelle Zeile eingefügt wird. Die Zeichenfolge wird so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

Wird `text` nicht angegeben (aber der Doppelpunkt), wird eine Leerzeile (Zeile der Länge 0) eingefügt. Wird weder der Operand `text` noch der Doppelpunkt angegeben, wird nur die neue aktuelle Zeile und Schrittweite festgelegt.

Mit jeder Anweisung @SET line [(inc)] werden eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite festgelegt. Die bisherigen aktuellen Werte für Zeilennummer und Schrittweite für die aktuelle Arbeitsdatei werden in einem Speicherbereich abgelegt, in dem maximal drei Wertepaare (Zeilennummer/Schrittweite) Platz haben.

Die früher in diesem Speicherbereich abgelegten Wertepaare rutschen dabei jeweils um einen Platz nach hinten. Ist der Speicherbereich voll, so geht das letzte (älteste) Wertepaar verloren.

Sind bei der Anweisung @SET keine Operanden angegeben, so wird das zuletzt im Speicherbereich abgelegte Wertepaar zur aktuellen Zeilennummer und zur aktuellen Schrittweite.
Die Verschiebung der Wertepaare im Speicherbereich hängt dabei davon ab, ob der Speicherbereich bereits ganz voll war oder nicht. War der Speicherbereich noch nicht ganz voll, so rücken alle Wertepaare um einen Platz nach vorn.
Der letzte Platz im Speicherbereich bleibt leer. Ist der Speicherbereich bei Eingabe der Anweisung @SET ohne Operanden bereits leer, so werden die bisherige aktuelle Zeilennummer und die bisherige aktuelle Schrittweite beibehalten und es wird die Meldung `EDT4964` ausgegeben.

Ist der Speicherbereich bei Eingabe der Anweisung @SET ohne Operanden bereits voll, so wird wie bereits zuvor das zuletzt im Speicherbereich abgelegte Wertepaar zur aktuellen Zeilennummer und zur aktuellen Schrittweite.

Im Speicherbereich selbst findet im Gegensatz zu vorher eine Rotation statt, d.h., alle Wertepaare im Speicherbereich, außer dem ersten, rücken einen Platz nach vorn und das bisher erste Wertepaar im Speicherbereich (die neue aktuelle Zeilennummer und die neue aktuelle Schrittweite) werden im letzten Platz des Speicherbereichs abgelegt.

Beim vollständigen Löschen der Arbeitsdatei wird der Speicherbereich geleert.

Bei dieser Anweisung darf der Anweisungsname ganz entfallen. Anders als bei den anderen @SET-Formaten, muss dann aber im F-Modus das Anweisungssymbol angegeben werden.

Bei Verwendung des Operanden `text` im F-Modus ist besondere Vorsicht geboten. Für seine Verarbeitung wird temporär in den L-Modus umgeschaltet und anschließend wieder der F-Modus aktiviert.
Dies geschieht auch, wenn der Operand z.B. eine Anweisung zum Umschalten in den L-Modus enthält. Enthält der Operand Dateneingaben und kommen in diesen das Zeichen Semikolon oder unpaarige Anführungszeichen vor, führt das durch die Zerlegung der Anweisungszeile in Einzelanweisungen möglicherweise zu unerwarteten Effekten.

9.115 @SETF – Arbeitsdatei wechseln und positionieren

Mit der Anweisung @SETF kann man mit oder ohne Wechsel der aktuellen Arbeitsdatei das Arbeitsfenster für eine Arbeitsdatei vertikal oder horizontal positionieren.

Operation	Operanden	F-Modus, L-Modus
@SETF **@#**	[{$0..$22 | **GL**OBAL | (0..22)}] [{line | vpos}] [{:col: | hpos}]	

$0..$22 Arbeitsdatei, in der positioniert wird. Die aktuelle Arbeitsdatei bleibt unverändert. Wenn keine weiteren Operanden angegeben werden, wird in der angegebenen Arbeitsdatei auf die (logisch) erste Zeile und erste Spalte positioniert.

0..22 Arbeitsdatei, in der positioniert wird. Es wird vor der Positionierung die angegebene Arbeitsdatei als aktuelle eingestellt. Vor dem Umschalten in die angegebene Arbeitsdatei wird eine eventuell vorhandene mit @PROC erzeugte Schachtelung der Arbeitsdateien aufgehoben.
Wenn keine weiteren Operanden angegeben werden, wird nur die angegebene Arbeitsdatei als aktuelle eingestellt, die Zeilen- und Spaltenposition in der angegebenen Arbeitsdatei bleiben dann unverändert.

Eine *aktive* Arbeitsdatei kann nicht zur aktuellen Arbeitsdatei werden, beim Versuch wird die Meldung `EDT4959` ausgegeben.

GLOBAL Es wird gleichzeitig in allen Arbeitsdateien positioniert. Wenn keine weiteren Operanden angegeben werden, wird in allen Arbeitsdateien auf die (logisch) erste Zeile und erste Spalte positioniert.

line Absolutangabe der vertikalen Position. Es ist die Zeilennummer anzugeben, die in der ersten Zeile des Arbeitsfensters angezeigt werden soll. Falls keine Zeile mit dieser Zeilennummer vorhanden ist, so wird auf die existierende Zeile mit der nächsten höheren Zeilennummer positioniert. Gibt es keine solche Zeile, so wird auf die existierende Zeile mit der höchsten Zeilennummer positioniert.

vpos
: Relative vertikale Positionieranweisung. Es können `+[n]`, `++`, `-[n]`, `--` sowie `+([m[,...]])`, `++([m[,...]])`, `-([m[,...]])` und `--([m[,...]])` angegeben werden.

  Dabei ist `m` 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, siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden hier nicht ausgewertet.

  Die relative Positionieranweisung wirkt in gleicher Weise, wie die entsprechende Anweisung (`+[n]`, `++` usw.) im F-Modus, wenn diese mit DÜE übertragen wird (siehe entsprechende Anweisung). Die Positionierung mit @SETF kann aber zusätzlich auch im L-Modus (etwa in Prozeduren) oder bei Steuerung des EDT über die Unterprogramm-Schnittstelle Verwendung finden.

col
: Absolutangabe der horizontalen Position. Es ist die Spaltennummer anzugeben, die als erste Spalte des Arbeitsfensters angezeigt werden soll.

hpos
: Relative horizontale Positionieranweisung. Es können `>[n]`, `<[n]` und `<<` angegeben werden.

  Die relative Positionieranweisung wirkt in gleicher Weise, wie die entsprechende Anweisung (`>[n]`, `<<` usw.) im F-Modus, wenn diese mit DÜE übertragen wird (siehe dort). Die Positionierung mit @SETF kann aber zusätzlich auch im L-Modus (etwa in Prozeduren) oder bei Steuerung des EDT über die Unterprogramm-Schnittstelle Verwendung finden.

Wird @SETF ganz ohne Operanden angegeben, wird in der aktuellen Arbeitsdatei auf die (logisch) erste Zeile und erste Spalte positioniert. Wird @SETF ohne Positionier-Operanden angegeben, so wird bei @SETF (n) nur die aktuelle Arbeitsdatei gewechselt und alle Positionen bleiben unverändert. Bei @SETF $n oder @SETF GLOBAL werden die betroffenen Arbeitsdateien auf die (logisch) erste Zeile und erste Spalte positioniert.

Die Abkürzung # darf nur im F-Modus eingegeben werden. Es muss mindestens ein Operand angegeben werden, andernfalls wird die #-Anweisung (Ausgeben der letzten Anweisung) ausgeführt.

Die Einstellung der Fensterposition wird getrennt für das obere und untere (mögliche) Datenfenster jeder Arbeitsdatei gespeichert. Für die Wirkung der Positionierung ist die Sichtbarkeit des betroffenen Fensters und evtl. das aktuelle Eingabefenster von Bedeutung. Dies gilt sinngemäß auch für Eingaben im L-Modus, wobei *Sichtbarkeit* und *Eingabefenster* sich auf den Zustand nach einem möglichen Wechsel in den F-Modus (mit der Anweisung @EDIT FULL) beziehen. Die entsprechende Information kann im L-Modus auch durch @STATUS=PAR(..) angezeigt werden.

Für nicht sichtbare Arbeitsdateien wird grundsätzlich das obere Fenster positioniert.

Ist eine betroffene Arbeitsdatei in genau einem Fenster (oben oder unten) sichtbar, so wird genau dieses Fenster positioniert, unabhängig davon, welches das Eingabefenster ist.

Ist eine Arbeitsdatei bei geteiltem Bildschirm in beiden Fenstern sichtbar, dann wird nur das Eingabefenster positioniert.

Hinweis

Werden mit @SETF Einstellungen für Arbeitsdateien vorgenommen (Operand `GLOBAL` oder `$0..$22`), werden diese nicht mehr temporär zur aktuellen Arbeitsdatei gemacht, wie im Kompatibilitätsmodus. Die dort beschriebenen Einschränkungen und Nebenwirkungen treten also im Unicode-Modus nicht auf.

Beispiel

Es wird vorausgesetzt, dass Arbeitsdatei 1 bereits Datensätze enthält und dass @PAR LOWER=ON eingestellt ist.

```
  23.00 ·····························································
@setf (1) 4 :3:·····························································0000.00:00001(02)
```

Der EDT soll in die Arbeitsdatei 1 wechseln und dort auf die Zeilennummer 4 und die Spalte 3 positionieren.

```
   4.00 d leider auch Theologie<·····························································
   5.00 rchaus studiert, mit heißem Bemühn.<·····························································
   6.00  steh ich nun ich armer Tor!<·····························································
   7.00 d bin so klug als wie zuvor;<·····························································
```

```
·····························································0004.00:00003(01)
```

Der Wechsel in Arbeitsdatei 1 und die Positionierung sind erfolgt.

9.116 @SETJV – Jobvariable katalogisieren und Wert zuweisen

Die Anweisung @SETJV trägt eine Jobvariable in den Katalog ein und/oder weist einer Jobvariablen einen Wert zu.

Operation	Operanden	F-Modus, L-Modus
@SETJV	{ [string] = string1[,...] [,**C**ODE=name] / string }	

string
: Zeichenfolge, die einen voll qualifizierten Jobvariablen-Namen angibt. Der Name muss den syntaktischen Regeln für Jobvariablen-Namen genügen, der EDT prüft diese aber nicht vollständig.

  Ist die Jobvariable noch nicht im Katalog vorhanden, wird sie mit den Standardfunktionen des DCLJV-Makros katalogisiert.

  Der Jobvariablen wird der Kettungsname *EDTLINK zugeordnet. Über ihn kann sie in späteren Anweisungen angesprochen werden. Die Zuweisung erfolgt, bevor die entsprechenden Systemschnittstellen aufgerufen werden. Treten bei der Wertzuweisung dann Fehler auf, bleibt die erfolgte Zuordnung aber bestehen.

  Wird string nicht angegeben, wird die Jobvariable mit dem Kettungsnamen *EDTLINK angesprochen. In diesem Fall muss string1 angegeben werden. Ist keine Jobvariable mit dem Kettungsnamen *EDTLINK verknüpft, wird die Anweisung mit der Meldung EDT5289 abgewiesen. Ist auch string1 nicht angegeben, wird die Anweisung mit der Meldung EDT3908 abgewiesen.

  Ist die Jobvariable nicht zugreifbar, wird die Anweisung mit der Meldung EDT4208 abgebrochen.

string1
: Eine oder mehrere Zeichenfolgen, die der Jobvariablen zugewiesen werden sollen.

  Sind mehrere Zeichenfolgen angegeben, werden sie zu einem Zwischenergebnis verkettet. Haben alle beteiligten Zeichenfolgen den gleichen Zeichensatz, wird dieser der Zeichensatz des Zwischenergebnisses. Sind unterschiedliche Zeichensätze beteiligt, ist der Zeichensatz des Zwischenergebnisses UTFE.

  Ist die Zeichenfolge länger als 256 Bytes, werden nur die ersten 256 Bytes als Wert zugewiesen und die Meldung EDT1936 wird ausgegeben.

Ist `string1` nicht angegeben oder ist `string1` die leere Zeichenfolge, wird eine leere Job-Variable angelegt. Existiert sie schon, wird ihr Wert nicht verändert.

name — Name eines Zeichensatzes, in den das Zwischenergebnis vor der Zuweisung an die Jobvariable konvertiert wird. Ist `name` nicht angegeben, wird `EDF041` verwendet. Der Zeichensatzname muss in XHCS bekannt sein, andernfalls wird die Anweisung mit der Meldung `EDT4980` abgewiesen.

Enthält die zuzuweisende Zeichenfolge Zeichen, die im angegebenen Zeichensatz ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls unterbleibt die Zuweisung und die Fehlermeldung `EDT5453` wird ausgegeben.

Ist das Subsystem Jobvariablen-Support nicht installiert, wird die Anweisung mit der Fehlermeldung `EDT5254` abgewiesen. Details zu Jobvariablen können dem Handbuch JV [9] entnommen werden.

9.117 @SETLIST – Erweitern einer Listenvariablen

Die Anweisung @SETLIST weist einer S-Listenvariablen Elemente zu. Dabei werden Werte aus Zeilen der aktuellen Arbeitsdatei oder aus Zeichenfolgevariablen übernommen.

Sind keine Werte spezifiziert, wird der Inhalt der S-Listenvariablen gelöscht.

Operation	**Operanden**	**F-Modus, L-Modus**
@SETLIST	string { [lines[,...]] [**MA**RK [m]] / svar } [:cols[,...]:] [,] [**M**ODE = { **A**PPEND / **P**REFIX / **O**VERWRITE }] [,**C**ODE =name]	

string — Zeichenfolge, die den gültigen Namen einer S-Listenvariablen angibt. Der Name muss den syntaktischen Regeln für einen S-Variablen-Namen genügen, diese werden aber vom EDT nicht vollständig überprüft.

lines — einer oder mehrere Zeilenbereiche, deren Inhalt in die S-Listenvariable übernommen werden soll. Ist im angegebenen Zeilenbereich keine Zeile vorhanden, wird die Meldung `EDT2903` ausgegeben. Ist kein Zeilenbereich angegeben, werden alle Zeilen der aktuellen Arbeitsdatei berücksichtigt.

MARK — Es sollen nur markierte Zeilen der angegebenen Zeilenbereiche übernommen werden. Ist `MARK` nicht angegeben, werden alle Zeilen übernommen.

m — Nummer der Satzmarkierung (`1..9`), die berücksichtigt werden soll. Ist `m` nicht angegeben, werden nur Datenzeilen mit Markierung 1 berücksichtigt.

svar — Name einer Zeichenfolgevariablen, deren Inhalt als Element aufgenommen werden soll.

cols — Einer oder mehrere Spaltenbereiche, deren Zeichen übernommen werden sollen. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Enthält eine Zeile in einer zu berücksichtigenden Spalte kein Zeichen, wird stattdessen ein Leerzeichen eingefügt.
Alle spezifizierten Zeichen werden in der Reihenfolge der Spaltenangabe ggf. auch mehrfach aneinandergefügt und das Ergebnis in das Listenelement eingefügt.

Ist dieser Operand nicht angegeben, wird die ganze Zeile übernommen.

MODE= legt fest, auf welche Art die Liste erweitert werden soll.

APPEND Die Liste wird am Ende erweitert, d.h. die neuen Listenelemente werden nach dem letzten Element angefügt (Standardwert).

PREFIX Die Liste wird am Anfang erweitert, d.h. die neuen Listenelemente werden vor dem ersten Element eingefügt.

OVERWRITE
Zunächst wird der Inhalt der S-Listenvariablen gelöscht. Anschließend werden die neuen Listenelemente aufgenommen.

Ist im angegebenen Zeilenbereich keine Zeile vorhanden, enthält die S-Listenvariable anschließend keine Elemente mehr. Im F-Modus wird zur Information die Meldung EDT0211 ausgegeben.

name Zeichensatz, in den die zuzuweisende Zeichenfolge konvertiert werden soll, bevor sie zugewiesen wird. Ist name nicht angegeben, wird EDF041 verwendet. Der Zeichensatzname muss in XHCS bekannt sein, andernfalls wird die Anweisung mit der Meldung EDT4980 abgewiesen.

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.

Die Listenvariable muss bereits existieren, sonst wird die Meldung EDT5274 ausgegeben. Ist sie keine Listenvariable, wird die Meldung EDT4910 ausgegeben. Ist die Listenvariable nicht vom Typ STRING oder ANY wird die Anweisung mit der Meldung EDT5343 abgewiesen.

Wird eine S-Variable durch APPEND oder PREFIX erweitert, wird im F-Modus zur Information die Meldung EDT0210 ausgegeben.

Ist im L-Modus die Arbeitsdatei leer und wurde ein Zeilenbereich (keine Zeichenfolgevariable) angegeben, wird die Meldung EDT2903 ausgegeben.

Enthält die zuzuweisende Zeichenfolge Zeichen, die im angegebenen Zeichensatz ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls unterbleibt die Zuweisung und die Anweisung wird mit der Fehlermeldung EDT5453 abgebrochen.

Ist der zuzuweisende Wert länger als 4096 Bytes, werden nur die ersten 4096 Bytes als Wert zugewiesen und die Meldung EDT2403 wird ausgegeben.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Details zu S-Listenvariablen können dem Handbuch SDF-P [7] entnommen werden.

9.118 @SETSW – Auftrags- und Benutzerschalter setzen

Mit der Anweisung @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 (Standardwert).

OFF — Die angegebenen Schalter werden zurückgesetzt.

U — Ist `U` angegeben, so bezieht sich die nachfolgende Angabe von `int1` und ggf. von `int2` auf einen Benutzerschalter der eigenen Benutzerkennung.

Ohne Angabe von `U` bezieht sich die Angabe von `int1` und ggf. von `int2` auf einen Auftragsschalter.

int1 — Nummer des Schalters (`0..31`), der gesetzt oder zurückgesetzt werden soll.

int2 — Alle Schalter von `int1` bis `int2 (0..31)` werden gesetzt bzw. zurückgesetzt. Ist `int2` kleiner als `int1`, so wird die Anweisung mit der Meldung `EDT3216` abgewiesen.

Durch die Anweisung @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

```
@SET #I2 = 6
@SETSW ON = U1-#I2,12-20,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 ---------------------------------------- (1)
   2.     @ @PRINT #S2 N
   3.     @ @RETURN
   4.     @ @PRINT #S3 N
   5.     @END
   1.     @SETSW OFF = 15 ------------------------------------------------ (2)
   1.     @DO 9 ---------------------------------------------------------- (3)
SCHALTER 15 IST AUS
   1.     @SETSW ON = 15 ------------------------------------------------- (4)
   1.     @DO 9 ---------------------------------------------------------- (5)
SCHALTER 15 IST AN
   1.
```

(1) In der Arbeitsdatei 9 wird eine Prozedur abgelegt, die bei gesetztem Auftragsschalter 15 die Zeichenfolgevariable #S3 ausgibt, andernfalls die Zeichenfolgevariable #S2.

(2) Der Auftragsschalter 15 wird zurückgesetzt.

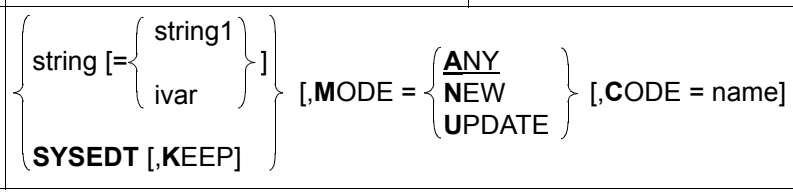
(3) Die Prozedur in der Arbeitsdatei 9 wird ausgeführt.

(4) Der Auftragsschalter 15 wird gesetzt.

(5) Die Prozedur in der Arbeitsdatei 9 wird ausgeführt.

9.119 @SETVAR – Deklarieren einer S-Variablen und Wertzuweisung

Die Anweisung @SETVAR deklariert eine S-Variable und/oder weist einer S-Variablen einen Wert zu.

Operation	Operanden	F-Modus, L-Modus
@SETVAR		

string — Zeichenfolge, die einen gültigen S-Variablen-Namen angibt. Der Name muss den syntaktischen Regeln für einen S-Variablen-Namen genügen, diese werden aber vom EDT nicht vollständig überprüft.

string1 — Zeichenfolge, die der S-Variablen zugewiesen werden soll.

Hat die S-Variable nicht den Typ `STRING` oder `ANY` bzw. ist sie ein Array oder eine Liste, wird die Anweisung mit der Meldung `EDT5342` abgebrochen.

Ist der zuzuweisende Wert länger als 4096 Bytes, werden nur die ersten 4096 Bytes als Wert zugewiesen und die Meldung `EDT2403` wird ausgegeben.

ivar — Ganzzahlvariable (`#I0..#I20`), deren Inhalt der über `string` angegebenen S-Variablen als Wert zugewiesen wird.

Hat die S-Variable nicht den Typ `INTEGER` oder `ANY` bzw. ist sie ein Array oder eine Liste, wird die Anweisung mit der Meldung `EDT5342` abgebrochen.

SYSEDT — Den S-Variablen `SYSEDT-S00..SYSEDT-S20` werden die Inhalte der Zeichenfolgevariablen `#S00` bis `#S20` zugewiesen. Treten dabei Fehler auf, werden entsprechende Meldungen ggf. auch mehrfach ausgegeben. Die Anweisung wird dabei nicht beendet.

Für S-Variable denen keine Zeichenfolge als Wert zugewiesen werden kann (anderer Typ) wird kein Fehler gemeldet, sondern die Zuweisung unterbleibt. Die Behandlung von nicht existierenden S-Variablen hängt von `MODE` ab.

KEEP Ist KEEP angegeben, wird der EDT so eingestellt, dass er die S-Variablen SYSEDT-S00..SYSEDT-S20 bei Beendigung nicht überschreibt. Ist KEEP nicht angegeben, wird die Einstellung zurückgenommen d.h. der EDT weist bei seiner Beendigung die Inhalte der Zeichenfolgevariablen #S00 bis #S20 zu und zwar im Zeichensatz EDF041.

MODE= legt fest, ob die S-Variable schon existieren soll.

- ANY Einer existierenden oder einer neuen S-Variable wird ein Wert zugewiesen.
- NEW Die S-Variable darf noch nicht existieren. Existiert sie schon, wird die Anweisung nicht ausgeführt und die Meldung EDT5272 ausgegeben. Ist SYSEDT angegeben wird die Angabe NEW wie ANY behandelt.
- UPDATE Die S-Variable muss schon existieren. Existiert sie noch nicht, wird die Anweisung nicht ausgeführt und die Meldung EDT5274 ausgegeben. Ist SYSEDT angegeben, wird die Meldung nicht ausgegeben und es wird nur denjenigen S-Variablen ein Wert zugewiesen, die schon existieren und vom Typ STRING oder ANY sind.

name Zeichensatz, in den die zuzuweisende Zeichenfolge konvertiert werden soll bevor sie zugewiesen wird. Ist name nicht angegeben, wird EDF041 verwendet. Ist string aber nicht string1 angegeben, wird der Operand ignoriert. Ansonsten muss der Zeichensatzname in XHCS bekannt sein, andernfalls wird die Anweisung mit der Meldung EDT4980 abgewiesen.

Ist string aber weder string1 noch ivar angegeben, so wird der S-Variablen die leere Zeichenfolge als Wert zugewiesen. Existiert sie noch nicht, wird sie mit Default-Attributen (TYPE=ANY,MULTIPLE-ELEMENTS=*NO,SCOPE=PROCEDURE)angelegt. Existiert die Variable bereits mit dazu unverträglichen Attributen, wird die Anweisung mit der Meldung EDT5342 abgewiesen.

Enthält die zuzuweisende Zeichenfolge Zeichen, die im angegebenen Zeichensatz ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls unterbleibt die Zuweisung und die Fehlermeldung EDT5453 wird ausgegeben.

Details zu S-Variablen können dem Handbuch SDF [6] entnommen werden.

9.120 @SHIH – Anweisungspuffer ausgeben

Mit der Anweisung @SHIH kann der Anweisungspuffer des EDT ausgegeben werden. Nur die im F-Modus eingegebenen Anweisungen werden in den Anweisungspuffer aufgenommen.
Die Blätteranweisungen, die Anweisungen zum Wechseln der Arbeitsdatei sowie die Anweisung @SHIH selbst werden nicht in den Anweisungspuffer aufgenommen.

Operation	**Operanden**	**F-Modus, L-Modus**
@SHIH	[[**T**O] line [(inc)]] [**F**ORWARD]	

line — 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 `line` nicht angegeben, wird das Ergebnis im L-Modus des Dialogbetriebs nach `SYSOUT` ausgegeben (im Stapelbetrieb können keine Ausgaben entstehen) und im F-Modus in die Arbeitsdatei `9` geschrieben. Die Arbeitsdatei `9` wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei `9` eine Datei geöffnet ist, wird die Meldung `EDT5189` ausgegeben und die Anweisung nicht ausgeführt.

inc — Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

FORWARD — Ist dieser Operand angegeben, werden die Anweisungen in der Reihenfolge ihrer Eingabe ausgegeben. Ist er nicht angegeben, werden sie in der umgekehrten Reihenfolge ausgegeben, also die zuletzt eingegebene Anweisung zuerst.

Der Anweisungspuffer kann unabhängig von der Länge der jeweiligen Anweisungen bis zu 2048 Anweisungen enthalten. Ist der Anweisungspuffer leer, dann wird die Anweisung @SHIH mit der Meldung `EDT5376` abgewiesen.

Es wird nicht berücksichtigt, ob eine Anweisung im oberen oder unteren Teil eines geteilten Bildschirmes eingegeben wurde. Die Speicherung im Anweisungspuffer erfolgt in der Reihenfolge von oben nach unten unabhängig von der Arbeitsdatei, auf die die Anweisung angewendet wurde. Anweisungen innerhalb einer Anweisungsfolge (durch ’;’ getrennte Anweisungen) werden einzeln abgelegt.

Bei Ausgabe in die Arbeitsdatei `9` (im F-Modus ohne `line` Operand) wird eine Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Außerdem wird für die Arbeitsdatei 9 implizit ein @LOWER ON abgesetzt.

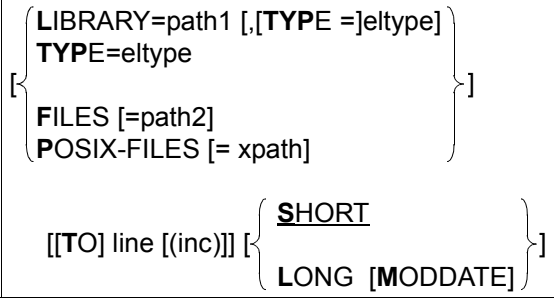
Eine Ausgabe nach `SYSOUT` erfolgt im Zeichensatz, der für diese Systemdatei eingestellt ist. Bei Ausgabe in eine Arbeitsdatei erfolgt die Ausgabe im Zeichensatz der Arbeitsdatei, ist diese leer und hat sie den Zeichensatz `*NONE`, dann im Zeichensatz `UTFE`. Im Zielzeichensatz nicht darstellbare Zeichen werden grundsätzlich durch Leerzeichen ersetzt.

Hinweis

Mit der Kurzanweisung `K` kann eine Anweisung in die Anweisungszeile kopiert werden. Man kann diese in einer anderen Arbeitsdatei zur Ausführung bringen, wenn eine Anweisung zum Wechseln der Arbeitsdatei vorangestellt wird (siehe Anweisung `$0..$22`).

9.121 @SHOW (Format 1) – Ausgeben eines Inhaltsverzeichnisses

Mit der Anweisung @SHOW (Format 1) kann man sich das Inhaltsverzeichnis einer Bibliothek oder eine Liste von Dateien aus dem BS2000-Katalog oder aus einem POSIX-Verzeichnis ausgeben lassen. Dabei kann man den Ort der Ausgabe festlegen. Wahlweise kann man sich zusätzliche Informationen über die Dateien oder Bibliothekselemente ausgeben lassen.

Operation	Operanden	F-Modus, L-Modus
@SHOW		

LIBRARY= Das Inhaltsverzeichnis einer Bibliothek oder eines Elementtyps einer Bibliothek soll ausgegeben werden. Gibt es für ein Element mehrere Versionen, werden alle Versionen angezeigt. Gibt es keine Elemente des angegebenen Elementtyps, wird die Meldung `EDT5287` ausgegeben. Existiert die angegebene Bibliothek nicht oder ist sie nicht wie erforderlich zugreifbar, wird eine entsprechende Meldung ausgegeben.

Fehlt die Angabe von `LIBRARY` (und auch `FILES` und `POSIX-FILES`), wird implizit die mit @PAR LIBRARY voreingestellte Bibliothek verwendet, sofern @PAR LIBRARY spezifiziert wurde, andernfalls wird die Meldung `EDT5181` ausgegeben.

path1 Name der Bibliothek.

TYPE= Es soll nur das Inhaltsverzeichnis aller Elemente eines bestimmten Typs ausgegeben werden. Fehlt die Angabe von TYPE, wird das gesamte Inhaltsverzeichnis der Bibliothek ausgegeben.

eltype Typ des Elements. Zulässige Typangaben sind `S`, `M`, `P`, `J`, `D`, `X`, `R`, `C`, `H`, `L`, `U`, `F`, `*STD` und freie Typnamen mit entsprechendem Basistyp. Die zulässigen Elementtypen und deren Bedeutung sind im Kapitel „Dateibearbeitung“ auf Seite 137 beschrieben.

FILES= Eine Liste von Dateien aus dem BS2000-Katalog soll ausgegeben werden.

path2 Bezeichnet die Dateien, die aufgelistet werden sollen. Der Operand `path2` kann ein vollqualifizierter oder teilqualifizierter Dateiname sein, darf Wildcards enthalten und bis zu 80 Zeichen lang sein.

Fehlt die Angabe von `path2`, wird eine Liste aller Dateien der eigenen Benutzerkennung ausgegeben. Wird keine Datei mit dem angegebenen Namen gefunden, wird die Meldung `EDT5281` ausgegeben.

POSIX-FILES=
Eine Liste von Dateien aus dem POSIX-Verzeichnis soll ausgegeben werden.

xpath Bezeichnet die POSIX-Dateien, die aufgelistet werden sollen. Ist `xpath` ein Verzeichnis, werden alle Dateien in diesem Verzeichnis (ohne Verzeichnisanteil) aufgelistet. Ist `xpath` eine einfache Datei, wird ihr Name wie angegeben angezeigt.

Der Operand `xpath` kann auch als Zeichenfolgevariable angegeben werden. Er *muss* als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

Fehlt die Angabe von `xpath`, wird eine Liste aller Dateien des aktuellen POSIX-Verzeichnisses ausgegeben. Wird keine Datei mit dem angegebenen Namen gefunden oder ist ein Verzeichnis nicht wie erforderlich zugreifbar, wird eine entsprechende Meldung ausgegeben.

line 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 `line` nicht angegeben, wird das Ergebnis im L-Modus des Dialogbetriebs nach `SYSOUT` ausgegeben, im Stapelbetrieb nach `SYSLST` ausgegeben und im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei 9 eine Datei geöffnet ist, wird die Meldung `EDT5189` ausgegeben und die Anweisung nicht ausgeführt.

inc Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

SHORT Dies ist der Standardwert für den Umfang der Informationsausgabe. Die Bedeutung von SHORT ist unterschiedlich für die einzelnen Dateitypen.

Für Bibliotheken wird für jedes Element ausgegeben:

Spalte	**Überschrift**	**Bedeutung**
2–5	TYP	Elementtyp
7–38	ELEMENT	Elementname
42–53	VERSION	Versionsbezeichnung oder @ für die höchstmögliche Version
56–59	VAR	Variantennummer
63–72	DATE	Userdatum (Format YYYY-MM-DD)

Die Liste ist nach Typnamen, Elementnamen und Versionsnamen alphabetisch sortiert. Bei Typnamen > 4, bei Elementnamen > 32 bzw. Versionsbezeichnungen > 12 Zeichen besteht ein Eintrag aus 2 Zeilen.

Bei Ausgabe nach SYSOUT oder SYSLST wird die Ausgabe mit einer Überschrift (siehe Tabelle) eingeleitet. Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne line Operand) wird die Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Bei Ausgabe in eine Arbeitsdatei mit dem Operanden line wird keine Überschrift ausgegeben.

Für BS2000-Dateien wird eine Datei pro Zeile ausgegeben. Es werden nur die Dateinamen ergänzt und die Katalogkennung und die Benutzerkennung ausgegeben. Die Liste ist nach den Dateinamen alphabetisch sortiert. Eine Überschrift wird nicht angezeigt.

Für POSIX-Dateien wird eine Datei pro Zeile ausgegeben. Es werden nur die Dateinamen ausgegeben. Die Liste ist nach den Dateinamen alphabetisch sortiert. Eine Überschrift wird nicht angezeigt.

LONG Für die Dateien werden zusätzliche Informationen ausgegeben. Die Bedeutung von LONG ist unterschiedlich für die einzelnen Dateitypen.

Für Bibliotheken wird für jedes Element eine Zeile mit folgendem Inhalt ausgegeben:

Spalte	Überschrift	Bedeutung
1-8	TYP	Elementtyp
10-73	ELEMENT L=path1	Elementname
75-98	VERSION	Versionsbezeichnung oder @ für die höchstmögliche Version
100-103	VAR	Variantennummer
105-114	DATE	Userdatum bzw. Datum der letzten Änderung (Format YYYY-MM-DD)
116-123	CODESET	Zeichensatz

Die Liste ist nach Typnamen, Elementnamen und Versionsnamen alphabetisch sortiert.

Bei Ausgabe nach SYSOUT oder SYSLST wird die Ausgabe mit einer Überschrift (siehe Tabelle) eingeleitet, wobei der Name der Bibliothek in der Überschrift enthalten ist. Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne line Operand) wird die Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Bei Ausgabe in eine Arbeitsdatei mit dem Operanden line wird keine Überschrift ausgegeben.

Für BS2000-Dateien wird für jede Datei eine Zeile mit folgendem Inhalt ausgegeben:

Spalte	Überschrift	Bedeutung
1-10	SIZE	Anzahl der PAM-Seiten
11	P	Datei auf privatem oder gemeinschaftlichem Datenträger (*/␣)
12-65	FILENAME	Dateiname mit CATID und USERID
67-76	LAST PP	Letzte benutzte PAM-Seite
78-87	CR-DATE bzw. MOD-DATE	Erstellungsdatum bzw. Datum der letzten Änderung (Format YYYY-MM-DD)
89	S	SHARE-Attribut (Y/N/S)
90	A	ACCESS-Attribut (W/R)
92-95	FCB	FCB-Typ (SAM/ISAM/PAM/BTAM/NONE)

Spalte	Überschrift	Bedeutung
97	R	READ-PASS-Attribut (Y/N)
98	W	WRITE-PASS-Attribut (Y/N)
100-107	CODESET	Zeichensatz

Die Liste ist nach den Dateinamen alphabetisch sortiert.

Bei Ausgabe nach SYSOUT oder SYSLST wird die Ausgabe mit einer Überschrift (siehe Tabelle) eingeleitet. Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne line Operand) wird die Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Bei Ausgabe in eine Arbeitsdatei mit dem Operanden line wird keine Überschrift ausgegeben.

Für POSIX-Dateien wird für jede Datei eine Zeile mit folgendem Inhalt ausgegeben:

Spalte	Überschrift	Bedeutung
1	T	Typ (D für Verzeichnis, L für Symbolischer Verweis, B für Gerätedatei(Block), C für Gerätedatei(Zeichen), M für Datei mit verteiltem Zugriff, P für FIFO, S für Semaphore Datei oder F für Datei)
2-10	ACCESS	Zugriffsrechte (RWX für *user*, *group*, *others*)
12-31	SIZE	Größe der Datei in Byte
33-42	MOD-DATE	Datum der letzten Änderung (Format YYYY-MM-DD)
ab 44	FILENAME	Dateiname (Groß-Kleinschreibung ist relevant)

Die Liste ist nach den Dateinamen alphabetisch sortiert.

Bei Ausgabe nach SYSOUT oder SYSLST wird die Ausgabe mit einer Überschrift (siehe Tabelle) eingeleitet. Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne line Operand) wird die Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Bei Ausgabe in eine Arbeitsdatei mit dem Operanden line wird keine Überschrift ausgegeben.

MODDATE Für BS2000-Dateien wird anstelle des Erstellungsdatums das Datum der letzten Änderung ausgegeben. Das Datumsformat bleibt gleich. In der Überschrift wird MOD–DATE ausgegeben.
Für Bibliotheken wird anstelle des Userdatums das Datum der letzten Änderung ausgegeben.

Für POSIX-Dateien wird der Operand ignoriert.

Eine Ausgabe nach SYSOUT oder SYSLST erfolgt im Zeichensatz, der für diese Systemdateien eingestellt ist.
Bei Ausgabe in eine Arbeitsdatei erfolgt die Ausgabe im Zeichensatz der Arbeitsdatei, ist diese leer und hat sie den Zeichensatz *NONE, dann im Zeichensatz EDF041.
Im Zielzeichensatz nicht darstellbare Zeichen werden grundsätzlich durch Leerzeichen ersetzt.

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 to 100(10)·····························0001.00:00001(00)
```

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  C    PROG                                 @             0003   2006-01-11
110.00  C    PROG1                                @             0004   2006-01-11
120.00  D    E.TEXT1                              @             0006   2005-12-05
130.00  D    E.TEXT2                              @             0013   2005-12-05
140.00  D    E.TEXT3                              100           0011   2003-12-13
150.00  D    E.TEXT3                              101           0121   2003-12-20
160.00  D    E.TEXT3                              102           0007   2004-04-11
170.00  J    ASSEMB                               @             0099   2006-01-11
180.00  J    FILE-TRANSFER                        @             0045   2006-01-11
190.00  J    PROC1                                1             0001   2005-12-05
200.00  D    DAS-IST-EIN-ELEMENT-MIT-EINEM-SE     @             0000   2005-08-17
210.00       HR-LANGEN-NAMEN
220.00  D    DAS-IST-EIN-ELEMENT-MIT-EINER-LA     URALT.VERSIO  0000   2003-08-17
230.00       NGEN-VERSIONSANGABE                  N
240.00 FREE DAS-IST-EIN-ELEMENT-MIT-EINEM-FR      @             0000   2003-08-17
250.00 TYPX EIEN-TYPNAMEN
251.00 ·····································································
252.00 ·····································································
253.00 ·····································································
254.00 ·····································································
255.00 ·····································································
show files=*text* long;edit long·····························0001.00:00001(00)
```

Ausführliche Informationen über alle BS2000-Dateien, deren Namen die Zeichenfolge `TEXT` enthalten, werden in Arbeitsdatei 9 ausgegeben. Um die Informationen komplett im Datenfenster zu sehen, wird `EDIT LONG` angegeben.

```
0000000003 :N:$USER.BSP.TEXT                                  0000000002 200
7-01-12 NW ISAM NN *NONE   <···················································
0000000006 :N:$USER.TEXT.PROG                                 0000000006 200
7-01-12 YR PAM  NY *NONE   <···················································
0000000015 :N:$USER.TEXT1                                     0000000014 200
7-01-12 NW SAM  NN EDF041  <···················································
0000000021 :N:$USER.TEXT2                                     0000000020 200
7-01-12 NW SAM  NN EDF03IRV<···················································
                                                                        :
                                                                        :
                                                                        :
show posix-files=data long;index on·····························0001.00:00001(09)
```

Ausführliche Informationen über die POSIX-Datei data im aktuellen POSIX-Verzeichnis werden in Arbeitsdatei 9 ausgegeben. Der `EDIT LONG` Modus wird wieder ausgeschaltet.

```
   1.00 FRW-RW-RW-  00000000000000000004  2006-11-08  data <·····················
   2.00 ·····························································
                                                                 
                                                                 
                                                                 
                                                                 
                                                                 
·····························································0001.00:00001(09)
```

9.122 @SHOW (Format 2) – Ausgeben der unterstützten Zeichensätze

Mit der Anweisung @SHOW (Format 2) kann eine Liste der von XHCS unterstützten Zeichensätze ausgegeben werden. Im Dialogbetrieb werden zusätzlich die von der Datensichtstation unterstützten Zeichensätze gekennzeichnet.

Operation	**Operanden**	**F-Modus, L-Modus**
@SHOW	**C**CS [[**T**O] line [(inc)]]	

line
: 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 `line` nicht angegeben, wird das Ergebnis im L-Modus des Dialogbetriebs nach `SYSOUT` ausgegeben, im Stapelbetrieb nach `SYSLST` ausgegeben und im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei 9 eine Datei geöffnet ist, wird die Meldung `EDT5189` ausgegeben und die Anweisung nicht ausgeführt.

inc
: Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne `line` Operand) wird eine Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden).

Es wird eine Liste der von XHCS unterstützten Zeichensätze ausgegeben. Im Gegensatz zum Verhalten des EDT V16.6 bzw. zum Verhalten im Kompatibilitäts-Modus kann jeder der dort aufgeführten Zeichensätze im Unicode-Modus als Zeichensatz für eine Arbeitsdatei eingestellt werden.

Jede Zeile der Liste enthält neben dem Namen des Zeichensatzes zusätzliche Kennzeichen, nämlich

- `P` wenn es sich um einen partiellen Zeichensatz handelt (siehe Handbuch XHCS [8]),
- `E` wenn es sich um einen EBCDIC-Zeichensatz handelt,
- `I` wenn es sich um einen ISO-Zeichensatz handelt und
- `*` wenn der Zeichensatz zu den von der Datensichtstation akzeptierten Zeichensätzen gehört (nur im Dialogbetrieb).
  Die mit `*` gekennzeichneten Zeichensätze sind also genau diejenigen, die mit der Anweisung @CODENAME (Format 2) als Kommunikations-Zeichensatz einstellbar sind.

Eine Ausgabe nach `SYSOUT` oder `SYSLST` erfolgt im Zeichensatz, der für diese Systemdateien eingestellt ist. Bei Ausgabe in eine Arbeitsdatei erfolgt die Ausgabe im Zeichensatz der Arbeitsdatei, ist diese leer und hat sie den Zeichensatz `*NONE`, dann im Zeichensatz `EDF041`. Im Zielzeichensatz nicht darstellbare Zeichen werden grundsätzlich durch Leerzeichen ersetzt.

9.123 @SORT – Sortieren von Zeilenbereichen

Mit der Anweisung @SORT kann man zusammenhängende Zeilenbereiche in der aktuellen Arbeitsdatei aufsteigend oder absteigend sortieren. Durch die Angabe eines Spaltenbereichs kann der für die Sortierung relevante Teil des Satzes eingeschränkt werden.

Operation	**Operanden**	**F-Modus, L-Modus**
@SORT	[lines] [{ :cols[:] / :**R** (cols*) }] [{ **A** / D }]	

lines — Zeilenbereich, dessen Daten sortiert werden sollen. Wird kein Zeilenbereich angegeben, werden alle Datensätze der aktuellen Arbeitsdatei sortiert.

cols — Spaltenbereich, dessen Zeichen zur Sortierung berücksichtigt werden sollen. Im Folgenden wird dieser Spaltenbereich als *Sortierfeld* bezeichnet.

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 so behandelt, als ob sie ein leeres Sortierfeld enthielte.

Die Sortierfelder zweier Zeilen werden zeichenweise von links nach rechts verglichen. Wird dabei das Ende eines Sortierfeldes (z.B. wegen Zeilenendes) erreicht (ohne Differenz), so gilt die Zeile mit dem kürzeren Sortierfeld als kleiner.

Wird kein Spaltenbereich angegeben, umfasst das Sortierfeld die gesamte Zeile.

cols* — Spaltenbereich ausgehend vom Satzende. Die Zählung vom Satzende Richtung Satzanfang bestimmt nur, welche Zeichen zum Sortierfeld gehören. Innerhalb des Sortierfeldes wird trotzdem von links nach rechts verglichen. Die Angabe `1-10` bedeutet also z.B. die letzten zehn Zeichen jeder Zeile.

A — Es wird aufsteigend sortiert (*ascending*).

D — Es wird absteigend sortiert (*descending*).

Zeilen mit gleichem Sortierfeld behalten ihre ursprüngliche Reihenfolge in der Arbeitsdatei.

Wenn der Arbeitsdatei ein 7-Bit- oder 8-Bit-Zeichensatz zugeordnet ist, wird die Reihenfolge der einzelnen Zeichen durch ihren als Binärzahl aufgefassten Bytecode bestimmt. Wenn der Arbeitsdatei ein Unicode-Zeichensatz zugeordnet ist, wird die Reihenfolge der einzelnen Zeichen durch den als Binärzahl aufgefassten UTF16-Code des Zeichens bestimmt. Das von XHCS verwaltete Sortiergewicht für die einzelnen Zeichen wird nicht berücksichtigt. Die Sortierreihenfolge bei Sortierung mit dem EDT kann also von der Sortierreihenfolge bei Sortierung mit dem Programm SORT abweichen.

Die Anweisung @SORT verwendet eine Kombination von *Quicksort* und *Bubblesort*.

Die Anweisung @SORT kann nicht für eine durch die Anweisung @OPEN (Format 2) real geöffnete Datei gegeben werden.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Beispiele

```
@SORT :1-15
```

sortiert alle Sätze der aktuellen Arbeitsdatei aufsteigend nach dem Inhalt der Spalten 1 bis 15.

```
@SORT &:1-15
```

sortiert alle Sätze des durch das Bereichssymbol & definierten Bereichs (siehe @RANGE-Anweisung) in 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.

9.124 @SPLIT – Ausgeben von 2 Arbeitsfenstern

Mit der Anweisung @SPLIT wird im F-Modus die Darstellung eines zweiten Arbeitsfensters am Bildschirm ein- bzw. ausgeschaltet. Jedes Arbeitsfenster hat eine eigene Anweisungszeile (siehe auch den Abschnitt „Das Arbeitsfenster“ auf Seite 107).

Operation	Operanden	F-Modus
@SPLIT	{ n { (0..22) | $0..$22 } | **O**FF }	

n — legt die Zeilenanzahl für das untere Arbeitsfenster inklusive Anweisungszeile fest. Für `n` ist mindestens `2` anzugeben und höchstens zwei weniger als die von der Datensichtstation je nach Bildschirmformat maximal erlaubte Zeilenzahl (siehe @VDT-Anweisung).

0..22 | $0..$22 — Nummer der Arbeitsdatei, die im unteren Arbeitsfenster gezeigt werden soll. Im oberen Arbeitsfenster wird die Arbeitsdatei gezeigt, in der die Anweisung gegeben wurde. Es ist erlaubt, für beide Arbeitsfenster die gleiche Arbeitsdatei einzustellen, so können etwa unterschiedliche Zeilenbereiche einer Arbeitsdatei gleichzeitig bearbeitet werden.

OFF — Das Arbeitsfenster, in dessen Anweisungszeile die Anweisung eingegeben wurde, wird vollständig gezeigt. Die Angabe von `0` (Null) anstelle von `OFF` wird nur noch aus Kompatibilitätsgründen unterstützt.

Bei Beginn des EDT-Laufs ist im F-Modus das Arbeitsfenster ungeteilt.

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 nach der Abarbeitung der Anweisungen 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.

Anstelle von @SPLIT kann mit der gleichen Funktionalität auch die Anweisung @PAR SPLIT benutzt werden. Zusätzlich lässt sich @PAR SPLIT gezielt für eine Arbeitsdatei oder global für alle Arbeitsdateien benutzen und ist auch im L-Modus und damit in EDT-Prozeduren erlaubt.

Beispiel

```
   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 ···············································································
:                                                                              :
:                                                                              :
:                                                                              :
@split 10(2)·····················································0001.00:00001(00)
```

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 ···············································································
·····························································0001.00:00001(00)
   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 ···············································································
·····························································0001.00:00001(02)
```

9.125 @STAJV – Information über Jobvariablen ausgeben

Die Anweisung @STAJV gibt Eigenschaften von Jobvariablen aus bzw. schreibt sie in eine Arbeitsdatei.

Operation	Operanden	F-Modus, L-Modus
@STAJV	[string] [**T**O line [(inc)]] [{ [**S**HORT] / **LONG** [ISO**4**] }]	

string — Zeichenfolge, die die Namen der Jobvariablen bezeichnet, deren Eigenschaften ausgegeben werden sollen. Es sind alle Angaben erlaubt, die auch im BS2000-Kommando `/SHOW-JV-ATTRIBUTES` gegeben werden dürfen. Die Angabe kann also auch teilqualifiziert oder mit Wildcards erfolgen.
Der EDT nimmt keine vollständige Überprüfung der Syntax vor.

Ist keine Jobvariable vorhanden, die die Angabe bezeichnet, wird die Meldung `EDT4982` ausgegeben.

Ist der Name der Jobvariablen voll qualifiziert, wird die Katalogkennung (CATID) nur mit ausgegeben, wenn sie in ihm schon vorhanden war.

Ist `string` nicht angegeben, werden alle Jobvariablen der aktuellen Kennung ausgegeben.

line — Zeilennummer, ab der die Informationen über die Jobvariablen 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 `line` nicht angegeben, wird das Ergebnis im L-Modus des Dialogbetriebs nach `SYSOUT` ausgegeben, im Stapelbetrieb nach `SYSLST` ausgegeben und im F-Modus in die Arbeitsdatei `9` geschrieben. Die Arbeitsdatei `9` wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei `9` eine Datei geöffnet ist, wird die Meldung `EDT5189` ausgegeben und die Anweisung nicht ausgeführt.

inc — Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

SHORT Es werden nur die Jobvariablen-Namen ausgegeben (Standardwert).

LONG Zusätzlich zu den Jobvariablen-Namen werden weitere Kataloginformationen ausgegeben:

Spalte	Überschrift	Bedeutung
1-7	SIZE	Länge des aktuellen Wertes in Bytes
8	M	Angabe, ob es sich um eine auftragsüberwachende Jobvariable handelt (*/␣)
9-62	JOBVARIABLE NAME	Jobvariablen-Name mit USERID und evtl. CATID (s.o.)
64-71	CR-DATE	Erstellungsdatum (YY-MM-DD)
73	S	SHARE-Attribut (Y/N)
75	A	ACCESS-Atribut (W/R)
77	R	READ-PASS-Attribut (Y/N)
79	W	WRITE-PASS-Attribut (Y/N)

Die Liste ist nach den Jobvariablen-Namen alphabetisch sortiert.

Bei Ausgabe nach SYSOUT oder SYSLST wird die Ausgabe mit einer Überschrift (siehe Tabelle) eingeleitet. Bei Ausgabe in die Arbeitsdatei 9 (im F-Modus ohne line Operand) wird die Überschrift in der Informationszeile ausgegeben (kann mit @PAR INFORMATION=ON sichtbar gemacht werden). Bei Ausgabe in eine Arbeitsdatei mit dem Operanden line wird keine Überschrift ausgegeben.

ISO4 Das Erstellungsdatum (CR-DATE) wird in der Form YYYY-MM-DD ausgegeben. Die nachfolgenden Felder (siehe Tabelle) verschieben sich entsprechend.

Eine Statusabfrage auf Systemjobvariablen ($SYSJV.) wird mit der Meldung EDT3087 abgewiesen.

Ist das Subsystem *Jobvariablen-Support* nicht installiert, wird die Anweisung mit der Fehlermeldung EDT5254 abgewiesen. Details zu Jobvariablen können dem Handbuch JV [9] entnommen werden.

Beispiel

```
   1.00 90-06-27178<·····························································
   2.00 GUTEN MORGEN, HEUTE IST DER 27.06.1990<·····························
   3.00 ·····························································

setjv 'heute'=2·····························0001.00:00001(00)
```

Der Jobvariablen HEUTE wird die Zeichenfolge zugeordnet, die in Zeile 2 steht.

```
   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·····0001.00:00001(00)
```

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 N W N N
·····························································

stajv'heu*'long4····························0001.00:00001(09)
```

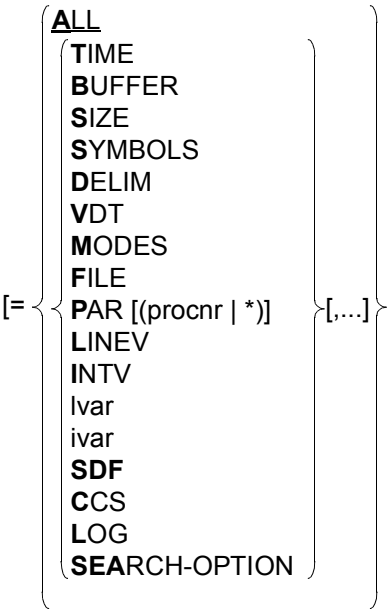
Informationen über die Jobvariable werden angezeigt.

```
SIZE   M    JOBVARIABLE NAME                                    CR-DATE    S A R
0000038 :XYZA:$EW.HEUTE                                         1990-06-27 N W N
·····························································

·····························0001.00:00001(09)
```

Informationen über die Jobvariable werden angezeigt.

9.126 @STATUS – Aktuelle Einstellungen und Variableninhalte anzeigen

Mit der Anweisung @STATUS können Einstellungen des EDT und der Systemumgebung sowie die Werte von Zeilennummer- und Ganzzahlvariablen ausgegeben werden.

Operation	Operanden	F-Modus, L-Modus
@STATUS		

ALL — Es werden alle Informationen der Parameter `TIME`, `BUFFER`, `SIZE`, `SYMBOLS`, `DELIM`, `VDT`, `MODES`, `LOG`, `SEARCH-OPTION`, `CCS` und `FILE` ausgegeben. Zusätzlich werden die Benutzerkennung (USERID), die Prozessfolgenummer (TSN) und der aktuelle Betriebsmodus des EDT ausgegeben (siehe Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21).

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-Anweisungen.

BUFFER — Ausgegeben werden die aktuelle Größe des physikalischen Ein-Ausgabe-Puffers für die Datensichtstation (siehe `SETBF`-Makro) und die Spaltennummer, ab der die Eingaben im L-Modus auf zulässige Länge geprüft werden sollen (siehe @CHECK-Anweisung). Im Stapelbetrieb wird nur die Spaltennummer ausgegeben.

SIZE	Dieser Operand wird nur noch aus Kompatibilitätsgründen unterstützt. Es wird immer der Wert `0` ausgegeben.
SYMBOLS	Ausgegeben werden das aktuelle Anweisungssymbol (siehe @:-Anweisung), die gültigen Literalbegrenzer (siehe @QUOTE-Anweisung), die gültigen Musterzeichen (siehe @SYMBOLS-Anweisung), das aktuelle Bereichssymbol mit dem vereinbarten Bereich (siehe @RANGE-Anweisung), das aktuelle Füllzeichen in hexadezimaler Form (siehe @SYMBOLS-Anweisung) und das aktuelle Ersatzzeichen, das bei Umcodierungen ungültige Zeichen ersetzt (siehe die Anweisung @PAR SUBSTITUTION-CHARACTER).
DELIM	Gibt die vereinbarte Menge der Textbegrenzerzeichen aus (siehe @DELIMIT-Anweisung).
VDT	Gibt die Anzahl der Bildschirmzeilen und -spalten (siehe @VDT-Anweisung) und das aktuell mit @PAR OPTIMIZE eingestellte Verhalten bei der Bildschirmausgabe aus. Bei der Datensichtstation 9763 wird zusätzlich das aktuelle Bildschirmformat ausgegeben (siehe @VDT-Anweisung).
MODES	Gibt die Voreinstellungen aus, die mit den Anweisungen @BLOCK, @CHECK, @INPUT, @TABS, @EDIT und @VTCSET vereinbart werden können. Zusätzlich werden die Einstellung der Syntaxkontrolle im L-Modus, der Ausführungsmodus (siehe @SYNTAX-Anweisung) und die mit der @AUTOSAVE-Anweisung einstellbaren Werte ausgegeben.
FILE	Gibt den globalen Dateinamen aus, der durch die letzte @FILE-Anweisung vereinbart wurde. Wurde dort auch eine Versionsnummer angegeben, wird diese mit ausgegeben. Für jede Arbeitsdatei wird zudem (wenn vorhanden) ein lokaler @FILE-Eintrag, der implizit (mit @READ oder @GET) oder explizit (mit @FILE .. LOCAL) definiert wurde sowie der Name einer geöffneten Datei bzw. der Bibliotheks- und Elementname eines geöffneten Bibliothekselements ausgegeben. Der Operand `FILE` führt zu keiner Ausgabe, wenn diese leer wäre.

PAR (procnr | *)

Für die angegebene Arbeitsdatei wird zunächst (falls vorhanden) ein lokaler @FILE-Eintrag sowie der Name einer geöffneten Datei bzw. eines Bibliothekselements im gleichen Format wie bei @STATUS=FILE ausgegeben. Anschließend werden der für diese Arbeitsdatei eingestellte Zeichensatz (siehe @CODENAME-Anweisung), die für die angegebene Arbeitsdatei aktuellen Werte aller mit der @PAR-Anweisung änderbaren arbeitsdateispezifischen Einstellungen sowie die Werte der arbeitsdateispezifischen symbolischen Zeilennummervariablen (`*`, `?`, `%` und `$`) ausgegeben. Schließlich wird für jedes (mögliche) Bildschirmfenster der angegebenen Arbeitsdatei die Zeilen- und Spaltenposition (siehe @SETF-Anweisung) und die Wahl für die Zeilennummernanzeige (siehe @PAR INDEX) angezeigt, sowie die Zeilenzahl des (nach Umschalten in den F-Modus) sichtbaren Bereichs (siehe @PAR SPLIT) und ein Kennzeichen, ob sich (nach Umschalten in den F-Modus) der Cursor in diesem Bildschirmfenster befindet.

Ist keine Arbeitsdatei angegeben, wird eine Liste mit den genannten Informationen für alle Arbeitsdateien ausgegeben. Ist `*` angegeben, werden die Informationen für die aktuelle Arbeitsdatei ausgegeben.

LINEV — Zeigt die Inhalte aller Zeilennummervariablen `#L0..#L20`.

INTV — Zeigt die Inhalte aller Ganzzahlvariablen `#I0..#I20`.

lvar — Der Inhalt der genannten Zeilennummervariablen wird ausgegeben.

ivar — Der Inhalt der genannten Ganzzahlvariablen wird ausgegeben.

SDF — Es werden die aktuellen SDF-Einstellungen angezeigt. Außerdem wird für jede Arbeitsdatei (nur wenn vorhanden) der Name des mit @PAR SDF-PROGRAM oder @SDFTEST eingestellten Programms ausgegeben, sowie ein Kennzeichen, ob es sich um einen internen oder externen Namen (gemäß Vereinbarung durch @PAR SDF-NAME-TYPE) handelt. Das Format der Ausgabe ist mit der entsprechenden Ausgabe für @STATUS=PAR identisch.

CCS — Es wird der eingestellte Kommunikations-Zeichensatz (`TERMACT`, siehe @CODENAME-Anweisung) sowie für jede Arbeitsdatei, die einen Zeichensatz hat und für jede Zeichenfolgevariable der aktuell eingestellte Zeichensatz ausgegeben. Ferner werden die Zeichensätze für die Systemdateien `SYSDTA`, `SYSOUT` und `SYSLST` sowie im Dialogbetrieb der mit `/MODIFY-TERMINAL-OPTIONS` eingestellte Zeichensatz (`TERMDEF`) und die Einstellung des Auto-Mechanismus ausgegeben.

LOG — Es werden die eingestellten Werte für die Protokollierung ausgegeben (siehe @LOG-Anweisung)

SEARCH-OPTION

Gibt die Voreinstellungen für die Suchfunktion (@ON-Anweisung) aus, die mit der Anweisung @SEARCH-OPTION vereinbart wurden.

line

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 `line` nicht angegeben, wird das Ergebnis im L-Modus des Dialogbetriebs nach `SYSOUT` ausgegeben, im Stapelbetrieb nach `SYSLST` ausgegeben und im F-Modus in die Arbeitsdatei 9 geschrieben. Die Arbeitsdatei 9 wird vor ihrer Verwendung gelöscht. Wenn in der Arbeitsdatei 9 eine Datei geöffnet ist, wird die Meldung `EDT5189` ausgegeben und die Anweisung nicht ausgeführt.

inc

Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

Wird kein Operand angegeben, wird `ALL` angenommen.

Die Reihenfolge der Ausgabe bei gleichzeitiger Angabe von mehreren Operanden ist fest vom EDT vorgegeben und insbesondere nicht von der Reihenfolge bei der Angabe der Operanden abhängig. Wird der gleiche Operand mehrfach angegeben, führt dies nicht zu einer Vervielfachung der Ausgabe für diesen Operanden.

Eine Ausgabe nach `SYSOUT` oder `SYSLST` erfolgt im Zeichensatz, der für diese Systemdateien eingestellt ist. Bei Ausgabe in eine Arbeitsdatei erfolgt die Ausgabe im Zeichensatz der Arbeitsdatei, ist dieser `*NONE` im Zeichensatz `EDF041`. Alle Zeichen, die bei der Umcodierung in den jeweiligen Ausgabe-Zeichensatz eventuell auf ein ungültiges Zeichen abgebildet werden könnten (etwa die Werte für @PAR SEPARATOR oder @PAR STRUCTURE), werden zusätzlich zur Darstellung als Zeichen (bzw. dann eventuell als Leerzeichen) auch noch hexadezimal in der Form `U'xxxx'` ausgegeben, wobei die Codierung des Zeichens in `UTF16` angezeigt wird.

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 line(inc)` ignoriert, d.h. die Ausgabe erfolgt stattdessen nach `SYSOUT`.

9.127 @SUFFIX – Anfügen von Zeichenfolgen

Mit der @SUFFIX-Anweisung wird an das Ende jeder Zeile bzw. Zeichenfolgevariablen jedes angegebenen Bereiches eine Zeichenfolge angefügt (siehe auch @PREFIX).

Operation	Operanden	F-Modus, L-Modus
@SUFFIX	{ lines / svars } [,...] **W**ITH string	

lines — Einer oder mehrere Zeilenbereiche, in denen am Zeilenende Text angefügt werden soll. Es werden nur existierende Zeilen bearbeitet.

svars — Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen am Ende Text angefügt werden soll.

string — Zeichenfolge, die an das Ende jeder Zeile bzw. Zeichenfolgevariablen jedes angegebenen Bereiches angefügt wird. Die Angabe einer leeren Zeichenfolge ist ebenfalls gestattet.

Die Zeichenfolge wird in den Zeichensatz der Arbeitsdatei bzw. der Zeichenfolgevariablen konvertiert. Enthält die Zeichenfolge Zeichen, die im Ziel-Zeichensatz nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), anderenfalls wird die @SUFFIX-Anweisung abgebrochen und die Fehlermeldung `EDT5453` ausgegeben.

Überschreitet eine Zeile bzw. eine Zeichenfolgevariable durch das Einfügen die maximale Länge von 32768 Zeichen, dann wird das Einfügen nicht durchgeführt und stattdessen die Meldung `EDT5474` ausgegeben.

Beim Auftreten von Fehlern während der Verarbeitung (`EDT5453` oder `EDT5474`) wird die Anweisung abgebrochen. Evtl. zu diesem Zeitpunkt schon erfolgreich modifizierte Zeilen und/oder Zeichenfolgevariablen bleiben modifiziert.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Beispiel

```
   1.00 UND<·····························································
   2.00 NOCH<····························································
   3.00 EINMAL<··························································
   4.00 UND<·····························································
   5.00 UND<·····························································
   6.00 ·····························································

suffix 4-5 with ' NOCH '·····································0001.00:00001(00)
```

Den Zeilen 4 und 5 wird die Zeichenfolge NOCH angefügt.

```
   1.00 UND<·····························································
   2.00 NOCH<····························································
   3.00 EINMAL<··························································
   4.00 UND NOCH <·····················································
   5.00 UND NOCH <·····················································
   6.00 ·····························································

suffix 4-5 with 3·····························································0001.00:00001(00)
```

Den Zeilen 4 und 5 wird der Inhalt der Zeile 3 angefügt.

```
   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·····················0001.00:00001(00)
```

Zunächst werden den Zeilen 4 und 5 je 5 Leerzeichen und anschließend den Zeilen 4 und 5 der Inhalt der Zeile 4 angefügt.

```
   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 ·····························································
```

9.128 @SYMBOLS – Symbole definieren

Mit der Anweisung @SYMBOLS können die Jokersymbole *asterisk* und *slash* zur Suche mit Platzhaltern vereinbart werden (siehe Abschnitt „Verwendung von Musterzeichen im Suchbegriff“ auf Seite 83).

Mit dem Operanden `FILLER` wird ein Füllzeichen vereinbart (siehe Abschnitt „Datenfenster“ auf Seite 110).

Operation	Operanden	F-Modus, L-Modus
@SYMBOLS	[{ **A**STERISK [= strspec1] / **S**LASH [= strspec2] / **F**ILLER [= strchar] } [,...]]	

ASTERISK= Vereinbart das Musterzeichen für eine beliebig lange, auch leere Zeichenfolge. Beim Start des EDT ist `'*'` voreingestellt. Ist der Operand nicht angegeben, wird das Musterrzeichen nicht verändert, es sei denn, es ist überhaupt kein Operand angegeben, dann wird der Standardwert `'*'` wieder eingesetzt.

strspec1 Sonderzeichen, das als Musterzeichen vereinbart werden soll. Ist kein Wert angegeben, wird der Standardwert `'*'` wieder eingesetzt.

SLASH= Definiert das Musterzeichen für genau ein Zeichen. Beim Start des EDT ist `'/'` voreingestellt. Ist der Operand nicht angegeben, wird das Musterzeichen nicht verändert, es sei denn, es ist überhaupt kein Operand angegeben, dann wird der Standardwert `'/'` wieder eingesetzt.

strspec2 Sonderzeichen, das als Musterzeichen definiert werden soll. Ist kein Wert angegeben, wird der Standardwert `'/'` wieder eingesetzt.

FILLER= Vereinbart ein Füllzeichen, das bei Dateneingabe an der Datensichtstation im F-Modus durch ein Leerzeichen ersetzt wird. Beim Start des EDT ist `X'00'` voreingestellt. Ist der Operand nicht angegeben, wird das Füllzeichen nicht verändert, es sei denn, es ist überhaupt kein Operand angegeben, dann wird der Standardwert `X'00'` wieder eingesetzt.

strchar Beliebiges Zeichen, das als Füllzeichen vereinbart werden soll.

Wird die Anweisung @SYMBOLS ohne Operand eingegeben, werden alle 3 Zeichen auf ihren Standardwert zurückgesetzt.

Die Zeichen spec1 und spec2 müssen verschieden voneinander sein, sonst wird die Anweisung mit der Meldung EDT3181 abgewiesen, und sie müssen von den in @QUOTE definierten Zeichen verschieden sein, sonst wird die Anweisung mit der Meldung EDT3180 abgewiesen.

Ist spec1 oder spec2 keines der zulässigen Sonderzeichen, wird die Anweisung @SYMBOLS mit der Fehlermeldung EDT3952 abgewiesen.

9.129 @SYNTAX – Einstellen des Test-Modus

Mit der Anweisung @SYNTAX kann für die Eingabe im L-Modus der Test-Modus ein- oder ausgeschaltet werden. Bei eingeschaltetem Test-Modus werden Anweisungen bei der Eingabe im L-Modus nicht ausgeführt (eine syntaktische Prüfung von Anweisungen findet unabhängig vom eingestellten Test-Modus immer statt).

Operation	Operanden	F-Modus, L-Modus
@SYNTAX	**TEST**MODE [= { **ON** / **OF**F }]	

TESTMODE= Mit dem Operand TESTMODE wird eingestellt, ob die eingegebenen Anweisungen ausgeführt werden oder nicht. Der Test-Modus ist nur bei Eingaben im L-Modus wirksam.

ON Die Anweisungen werden syntaktisch geprüft, aber nicht ausgeführt (auch keine externe Anweisungsroutine, die als Anweisungsfilter arbeitet). Ausnahmen hiervon sind nachfolgend aufgeführt. Im L-Modus eingegebene Datenzeilen werden nicht in die Arbeitsdatei übernommen. Eingaben, die mit mindestens einem EDT-Anweisungssymbol beginnen, werden syntaktisch geprüft.

Folgende Anweisungen werden auch bei eingeschaltetem Test-Modus immer ausgeführt:

- @HALT, @LOG, @RETURN, @SYNTAX und @: (Umdefinieren des EDT-Anweisungssymbols)
- @STATUS, die Ausgabe erfolgt aber bei eingeschaltetem Test-Modus immer nach SYSOUT

Folgende Anweisungen und Operanden werden nicht geprüft:

- Aufruf externer Anweisungsroutinen (Anweisungen mit Benutzeranweisungssymbol)
- Der Operand text bei den Anweisungen @+, @-, @IF und @SET (Format 6). Enthält die Anweisung sonst keinen Fehler, wird im Dialog die Meldung EDT0110 statt der Meldung EDT0100 ausgegeben.
- Anweisungen mit indirekter Operandenangabe.

OFF Der Test-Modus wird ausgeschaltet.

Der Test-Modus ist beim Start des EDT immer ausgeschaltet.

Mit der Anweisung @STATUS kann die aktuelle Einstellung für den Test-Modus ausgegeben werden.

Im L-Modus wird bei eingeschaltetem Test-Modus zusätzlich zur fehlerhaften Anweisung die Stelle mit ： markiert, an der der Fehler erkannt wurde. Bei Anweisungen ohne Fehler wird im Dialog die Meldung `EDT0100` ausgegeben.

Hinweis

Soll bei den Anweisungen @, @+, @- und @SET (Format 6) der Operand `text` syntaktisch geprüft werden, so muss die Anweisung in zwei Anweisungen aufgeteilt werden, z.B. @3:@... sollte in die zwei Anweisungen @3 und @... aufgeteilt werden.

9.130 @SYSTEM – Systemkommandos absetzen

Mit der Anweisung @SYSTEM kann man entweder den EDT-Lauf unterbrechen (wie mit K2) oder ein Betriebssystemkommando zur Ausführung bringen, ohne dass der EDT-Lauf unterbrochen wird.

Operation	**Operanden**	**F-Modus, L-Modus**
@SYSTEM	[string [**T**O line [(inc)]]]	

string — Zeichenfolge, die das Kommando enthält, das ausgeführt werden soll. Es wird ein BS2000-Kommando erwartet, das sofort ausgeführt wird. Anschließend wird der EDT-Lauf fortgesetzt, sofern es sich nicht um ein Kommando handelt, das zum Entladen des Programms führt (siehe unten). In diesem Fall wird auch der EDT entladen, ohne noch einmal die Kontrolle zu bekommen.

Wird `string` nicht angegeben, wird der EDT-Lauf unterbrochen. Durch das Kommando `/RESUME-PROGRAM` oder `/INFORM-PROGRAM` kann der EDT-Lauf an der Stelle fortgesetzt werden, an der er durch @SYSTEM unterbrochen wurde (siehe Abschnitt „Unterbrechen des EDT-Laufs“ auf Seite 95). In nichtunterbrechbaren Systemprozeduren ist ein Unterbrechen des EDT-Laufs im Dialog nicht möglich.

line — Zeilennummer, ab der eine eventuelle Ausgabe des Systemkommandos in die aktuelle Arbeitsdatei eingefügt werden soll. Es können nur solche Ausgaben eingefügt werden, die vom Betriebssystem nach `SYSOUT` ausgegeben werden (darunter fällt z.B. nicht die formatierte Ausgabe des `/SHOW-FILE`-Kommandos). Die Kommando-Ausgaben werden umgelenkt, d.h. die Ausgabe erfolgt nur noch in die aktuelle Arbeitsdatei und nicht mehr nach `SYSOUT`. Dabei wird die Ausgabe vom Zeichensatz `EDF041` in den Zeichensatz der aktuellen Arbeitsdatei konvertiert.

Enthält die Ausgabe Zeichen, die im Zeichensatz der aktuellen Arbeitsdatei ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Fehlermeldung `EDT5453` ausgegeben.

Wird `line` nicht angegeben, erfolgt die Ausgabe des Systemkommandos gemäß dessen Spezifikation (normalerweise nach `SYSOUT`). Falls das Kommando formatierte Ausgaben erzeugt hat (z.B. `/SHOW-FILE`) muss das Arbeitsfenster evtl. mit K3 neu aufgebaut werden (siehe Abschnitt „Funktionstasten im F-Modus“ auf Seite 128).

inc — Schrittweite, aus der die auf `line` folgenden Zeilennummern gebildet werden. Wird `inc` nicht angegeben, wird die implizit durch `line` gegebene Schrittweite verwendet (siehe Abschnitt „Implizite Schrittweitenvergabe“ auf Seite 36).

Die Anweisung @SYSTEM gehört zu den sicherheitsrelevanten Anweisungen des EDT (siehe hierzu auch Abschnitt „Zugriffsschutz“ auf Seite 103). In nichtunterbrechbaren Systemprozeduren im Dialog und bei Eingabe von einer Datei (Lesen mit `RDATA` von `SYSDTA` ungleich `SYSCMD`, Abarbeiten einer Start-Prozedur) wird die Anweisung abgewiesen.

Das Kommando kann mit und ohne Schrägstrich am Anfang geschrieben werden. Vor der Übergabe an das Betriebssystem wird der Kommandostring vom Zeichensatz der Eingabequelle in den Zeichensatz `UTFE` umcodiert.

Es dürfen nur Kommandos angegeben werden, die mit dem `CMD`-Makro abgesetzt werden dürfen. Ist ein Kommando an der `CMD`-Schnittstelle oder in der aktuellen SDF-Syntaxdatei nicht erlaubt, wird es mit der Fehlermeldung `EDT4300` (die auch den Meldungsschlüssel aus dem Kommando-Returncode enthält) abgewiesen. Die für Verwendung mit dem `CMD`-Makro zugelassenen Kommandos sind im Handbuch Makroaufrufe an den Ablaufteil [12] beschrieben.

Einige Systemkommandos führen bei Verwendung über die `CMD`-Schnittstelle zum Entladen des Programms (z.B. `/EXIT-JOB`, `/LOGOFF`, `/HELP-SDF`, `/CALL-PROCEDURE`, `/START-PROGRAM`, `/LOAD-PROGRAM` oder ein mittels SDF-A definiertes und durch Kommandoprozeduren implementiertes Anwender-Kommando). Eine vollständige Übersicht dieser Kommandos findet man ebenfalls im Handbuch Makroaufrufe an den Ablaufteil [12]. Das Entladen des EDT durch ein derartiges Kommando sollte vermieden werden, da weder offene Dateien geschlossen werden, noch die Möglichkeit gegeben wird, ungesicherte Arbeitsdateien zurück zu schreiben.

Beispiel

```
   1.00 ····························································
   2.00 ····························································
   3.00 ····························································
   4.00 ····························································
   5.00 ····························································
   6.00 ····························································




@SYSTEM '/SHOW-SYSTEM-INFORMATION' TO 1;@ON & F NOT 'OSD' DEL··0000.00:00010(00)
```

Aus der Ausgabe des Kommandos `/SHOW-SYSTEM-INFORMATION` soll Information über die OSD-Version extrahiert werden.

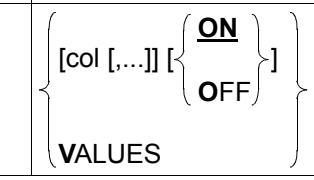
```
  12.00                OSD-BC-VERSION  = V05.0C0000<·····················
  27.00 VM2000-MONITOR-  OSD-BC-VERSION  = *NONE<···························
  25.00 ····························································
  26.00 ····························································
  27.00 ····························································
  28.00 ····························································




·····························································0012.00:00001(00)
```

Die relevante Information wird in der Arbeitsdatei abgelegt.

9.131 @TABS (Format 1) – Hardwaretabulatoren definieren und ausgeben

Mit dem Format 1 der @TABS-Anweisung können für das Positionieren mit dem Hardwaretabulator Tabulatorpositionen definiert und ihre aktuellen Werte ausgegeben werden. Der Hardwaretabulator wirkt nur im F-Modus.

Die Einstellungen des Hardware- und Softwaretabulators werden getrennt gespeichert. Es kann allerdings nur einer der beiden Tabulatoren aktiv sein. Wird der Hardwaretabulator aktiviert, wird daher automatisch ein evtl. aktiver Softwaretabulator deaktiviert.

Operation	Operanden	F-Modus, L-Modus
@TABS		

col — Gibt, durch Komma voneinander getrennt, eine oder mehrere Tabulatorpositionen für das Positionieren mit dem Hardwaretabulator an. Die Anzahl der möglichen Tabulatorpositionen ist durch die Länge einer Anweisung und möglicherweise durch die Hardware beschränkt. Der EDT positioniert auf die Tabulatorpositionen in der angegebenen Reihenfolge. Sind die Tabulatorpositionen nicht aufsteigend geordnet, wird die @TABS-Anweisung nicht ausgeführt und die Fehlermeldung `EDT4940` ausgegeben.

Werden die Tabulatorpositionen so definiert, das für die Darstellung eines Bildschirms mehr Felder pro Zeile benötigt werden, als an der jeweiligen Datensichtstation erlaubt sind, wird die @TABS-Anweisung mit der Fehlermeldung `EDT5463` abgewiesen (an den Datensichtstationen 975x sind beispielsweise nur 64 Felder pro Zeile erlaubt).

Ist `col` nicht angegeben, dann bleiben die aktuellen Werte der Tabulatorpositionen des Hardwaretabulators unverändert.

ON — Die Funktion des Hardwaretabulators wird aktiviert (Standardwert).

Werden durch die aktuelle oder eine vorhergehende Anweisung Tabulatorpositionen definiert, dann kann durch die Eingabe von TAR die Schreibmarke zur nächsten definierten Tabulatorposition rechts von der aktuellen Position der Schreibmarke positioniert werden. Dies entspricht der *Vorwärts-Strategie* bei der Auswertung von Softwaretabulatoren.

Sind keine Tabulatorpositionen definiert, dann wird die @TABS-Anweisung mit der Fehlermeldung EDT4941 abgewiesen.

OFF — Die Funktion des Hardwaretabulators wird ausgeschaltet. Die definierten Tabulatorpositionen bleiben erhalten und können mit der Anweisung @TABS ON wieder aktiviert werden.

VALUES — Die aktuellen Tabulatorpositionen des Hardwaretabulators werden ausgegeben. Ist keine Tabulatorposition definiert, erfolgt keine Ausgabe.

Die auszugebenden Tabulatorpositionen sind fünfstellig. Pro Ausgabezeile werden bis zu 11 Tabulatorpositionen ausgegeben und danach erfolgt ein Umbruch.

Die Ausgabe erfolgt im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST.

Beim Start des EDT ist der Hardwaretabulator ausgeschaltet.

Im EDIT-LONG-Modus kann mit dem Hardwaretabulator nur auf diejenigen Tabulatorpositionen positioniert werden, deren Werte kleiner als die Bildschirmbreite sind. Tabulatorpositionen, deren Werte größer als die Bildschirmbreite sind, werden ignoriert.

Ist der Hardwaretabulator aktiviert und ist mit der Anweisung @SCALE ON im F-Modus die Anzeige des Spaltenzählers eingeschaltet, dann wird eine weitere Bildschirmzeile ausgegeben, in der die aktuellen Tabulatorpositionen mit 'I' angezeigt werden. Die Anzeige erfolgt nicht im EDIT-LONG-Modus.

Ist der Hardwaretabulator aktiviert, kann nur innerhalb der Tabulatorpositionen mit EFG und AFG ein- bzw. ausgefügt werden, da der Hardwaretabulator über Felder der Datensichtstation realisiert ist.

Beispiel

```
  23.00 ·····································································
tabs 10,16,40·····················································0000.00:00001(00)
```

Als Tabulatorpositionen für den Hardwaretabulator werden die Werte 10,16 und 40 definiert.

```
   1.00          BALR  14,15                  IN UNTERPROGRAMM UND ZURUECK<····
   2.00 SPRUNG   DC C  ' IN ORDNUNG'<·····································
   3.00 ·····································································
```

Durch die Eingabe von TAB an der Datensichtstation wird auf die Tabulatorpositionen 10,16 und 40 positioniert.

9.132 @TABS (Format 2) – Softwaretabulatoren definieren und ausgeben

Mit dem Format 2 der @TABS-Anweisung können für das Positionieren mit Softwaretabulatoren Tabulatorzeichen und Tabulatorpositionen definiert, deren aktuelle Werte ausgegeben und der Softwaretabulator aktiviert bzw. deaktiviert werden.

Sind Softwaretabulatoren aktiviert, dann erfolgt die Tabulatorexpansion bei der Eingabe von Datenzeilen von einer Datensichtstation (nicht im Stapelbetrieb oder bei Eingabe aus Prozeduren) im L-Modus (auch bei indirekter Eingabe über den Text-Operand einiger Anweisungen) oder der Eingabe/Modifikation von Datenzeilen im Datenfenster des F-Modus durch den EDT. Ist der Softwaretabulator definiert (nicht notwendig aktiviert), kann man die Tabulatorexpansion für existierende Zeilen durch @TABS (Format 3) veranlassen. Die Tabulatorexpansion erfolgt von links nach rechts in den Datenzeilen, wobei zwei Expansions-Strategien eingestellt werden können.

Die übliche *Vorwärts-Strategie* bewirkt, dass ein Tabulatorzeichen durch so viele Leerzeichen (mindestens eines) ersetzt wird, dass das darauf folgende Zeichen auf der nächsten rechts von der aktuellen Position liegenden Tabulatorposition steht.

Die eher ungewöhnliche *Positionier-Strategie* bewirkt, dass das n-te Tabulatorzeichen auf die n-te Tabulatorposition positioniert, unabhängig davon, ob diese links oder rechts von der aktuellen Position liegt. Liegt sie rechts von der aktuellen Position, wird das Tabulatorzeichen wie bei der *Vorwärts-Strategie* durch entsprechend viele Leerzeichen ersetzt. Liegt sie dagegen links von der aktuellen Position (oder genau darauf), wird das Tabulatorzeichen gelöscht und die aktuelle Position auf die Tabulatorposition gesetzt. Nachfolgende Zeichen werden dann normalerweise in der eingegebenen Zeile davor stehende Zeichen überschreiben. Man kann eine Meldung ausgeben lassen, wenn das geschieht.

In beiden Strategien werden Tabulatorzeichen, zu denen keine Tabulatorposition mehr ermittelt werden kann, nicht ersetzt und bleiben als normale Textzeichen erhalten.

Die Einstellungen des Software- und Hardwaretabulators werden getrennt gespeichert. Es kann allerdings nur einer der beiden Tabulatoren aktiv sein. Wird der Softwaretabulator aktiviert, wird daher automatisch ein evtl. aktiver Hardwaretabulator deaktiviert.

Operation	Operanden	F-Modus, L-Modus
@TABS	: : { [char [:] col [,...]] [{ **C**HECK / **F**ORWARD / **N**OCHECK } [col1]] / **ON** / **O**FF / **V**ALUES }	

char — Zeichen, das vom EDT bei nachfolgenden Eingaben als Tabulatorzeichen interpretiert werden soll. Der Softwaretabulator wird dabei implizit aktiviert. Als Tabulatorzeichen sollten weder das Anweisungssymbol noch das Separatorzeichen angegeben werden. Das Zeichen ; kann bei der Eingabe der Anweisungen im F-Modus nicht verwendet werden, da es als Trennzeichen für Anweisungen interpretiert wird. Das Tabulatorzeichen `char` und `col` müssen durch : getrennt werden, falls `char` eines der Zeichen `C`, `F`, `O`, `V` oder `N` oder eine Ziffer ist.

Ist `char` nicht angegeben, aber eine Strategie, dann wird nur die Strategie geändert.

Ist weder `char` noch eine Strategie angegeben, dann werden die aktuellen Einstellungen für den Softwaretabulator als nicht definiert gesetzt und der Softwaretabulator deaktiviert.

col — Gibt, durch Komma voneinander getrennt, die Tabulatorpositionen für das Positionieren mit dem Tabulatorzeichen an. Die Anzahl der möglichen Tabulatorpositionen ist nur durch die Länge einer Anweisung beschränkt. Sind die Tabulatorpositionen nicht aufsteigend geordnet, wird die @TABS-Anweisung nicht ausgeführt und die Fehlermeldung `EDT4940` ausgegeben.

CHECK — Der EDT verwendet die in der Einleitung beschriebene *Positionier-Strategie* und gibt eine Meldung aus, wenn auf eine Tabulatorposition positioniert wird, die links von der aktuellen Position in der Zeile liegt. Falls dies bei der Eingabe von Text an der Datensichtstation auftritt, wird die Meldung `EDT2902` ausgegeben. Falls dies bei der Auswertung von Tabulatorzeichen in einer Arbeitsdatei mit der Anweisung @TABS (Format 3) vorkommt, wird die Meldung `EDT4312` ausgegeben. Zusätzlich werden in diesem Falle, beginnend mit der Zeile, auf die sich die Meldung bezieht, keine weiteren Tabulatorexpansionen durchgeführt.

Wird `CHECK` angegeben, aber nicht `char`, so wird die Strategie eingestellt, unabhängig davon, ob ein Softwaretabulator definiert und aktiv ist. Sie gilt, bis sie wieder explizit geändert wird.

FORWARD — Der EDT verwendet die in der Einleitung beschriebene *Vorwärts-Strategie*.

Wird `FORWARD` angegeben, aber nicht `char`, so wird die Strategie eingestellt, unabhängig davon, ob ein Softwaretabulator definiert und aktiv ist. Sie gilt, bis sie wieder explizit geändert wird.

NOCHECK Der EDT verwendet die in der Einleitung beschriebene *Positionier-Strategie* und überprüft nicht, ob eine Tabulatorexpansion auf eine Position links von der aktuellen Zeilenposition erfolgt.

Wird NOCHECK angegeben, aber nicht char, so wird die Strategie eingestellt, unabhängig davon, ob ein Softwaretabulator definiert und aktiv ist.

Sie gilt, bis sie wieder explizit geändert wird.

Beim Start des EDT ist NOCHECK voreingestellt.

col1 Gibt die Anzahl der Zeichen (1..32768) pro Zeile für die Zeilenlängenprüfung im L-Modus an. Der EDT überprüft die Anzahl der Zeichen in einer Zeile, die neu eingegeben wurde. Insbesondere sollen Überschreitungen der vorgegebenen Zeilenlänge als Folge von eventuell stattfindenden Tabulatorexpansionen angezeigt.

Ist eine Zeile länger als col1, wird diese Zeile zwar angelegt, aber der EDT macht mit der Meldung EDT2901 darauf aufmerksam, dass die vorgegebene Anzahl an Zeichen pro Zeile überschritten wurde.

Ist col1 nicht angegeben, dann bleibt der eingestellte Wert unverändert.

Beim Start des EDT ist der Wert für die Zeilenlängenprüfung im L-Modus auf 32768 Zeichen eingestellt.

ON Die Funktion des Softwaretabulators wird aktiviert. Es muss dazu bereits ein Softwaretabulator definiert sein.

Sind keine Tabulatorpositionen definiert, dann wird die @TABS-Anweisung mit der Fehlermeldung EDT4941 abgewiesen.

OFF Die Funktion des Softwaretabulators wird ausgeschaltet. Die definierten Tabulatorpositionen bleiben erhalten und können mit der Anweisung @TABS ::ON wieder aktiviert werden.

VALUES Das aktuelle Tabulatorzeichen und die zugehörigen Tabulatorpositionen des Softwaretabulators werden ausgegeben. Ist kein Tabulatorzeichen definiert, erfolgt keine Ausgabe.

Die auszugebenden Tabulatorpositionen sind fünfstellig. Pro Ausgabezeile werden bis zu 11 Tabulatorpositionen ausgegeben und danach erfolgt ein Umbruch. Das Tabulatorzeichen wird in den Fortsetzungszeilen nicht ausgegeben.

Die Ausgabe erfolgt im Dialogbetrieb nach SYSOUT und im Stapelbetrieb nach SYSLST.

Überschreitet eine Zeile durch die Tabulatorexpansion die maximale Länge von 32768 Zeichen, wird auf die maximale Länge abgeschnitten und und im L-Modus die Meldung EDT1903 bzw. im F-Modus die Meldung EDT2400 ausgegeben.ausgegeben.

Sind Tabulatorpositionen definiert und ist mit der Anweisung @SCALE ON im F-Modus die Anzeige des Spaltenzählers eingeschaltet, dann wird eine weitere Bildschirmzeile ausgegeben, in der die aktuellen Tabulatorpositionen mit 'I' angezeigt werden. In dieser Zeile wird auch das Tabulatorzeichen selbst in der Kurzanweisungsspalte abgebildet.

Das Tabulatorzeichen wirkt global für alle Arbeitsdateien. Bei der Dateneingabe im F-Modus erfolgt die Tabulatorexpansion in der gesamten Zeile und nicht nur im angezeigten Ausschnitt.

Wird nur eine Strategie angegeben (z.B. @TABS::FORWARD), wird diese Strategie eingestellt, auch wenn kein Softwaretabulator definiert ist. Wird *keine* Strategie angegeben, bleibt die *eingestellte* Strategie erhalten. Beim Start des EDT ist NOCHECK voreingestellt.

Hinweis

Durch die Angabe von @TABS:: werden die aktuellen Einstellungen für den Software-Tabulator als nicht definiert gesetzt. Ist gar kein Software-Tabulator definiert (aber z.B. ein Hardware-Tabulator) bleibt diese Anweisung wirkungslos.

Der Wert für die Zeilenlängenprüfung im L-Modus sollte besser durch die Anweisung @CHECK verändert werden. Hier wird sie nur noch aus Kompatibilitätsgründen unterstützt.

Die Hauptanwendung der @TABS-Anweisung ist das spaltengerechte Erstellen von Dateien, z.B. von Quelldateien. Beispielsweise ist bei der Erstellung einer Quelldatei für ein Assemblerprogramm die Anweisung @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 Spalte 72, mit der man Fortsetzungzeilen kennzeichnet, nicht versehentlich überschrieben wird.

Achtung

Überzählige Tabulatorzeichen in einer Zeile sind bei der *Positionier-Strategie* gefährlich. Kommt es zur erneuten Tabulatorexpansion, z.B. durch eine Korrektur in der Zeile im F-Modus, werden diese plötzlich wirksam, meist auf nicht erwünschte Weise.

Beispiel

```
  23.00 ·····························································
lower off;tabs ::[: 10,16,40·····································0000.00:00001(00)
```

Als Tabulatorzeichen wird [vereinbart. Die Tabulatorpositionen seien 10, 16 und 40.

```
   1.00 [balr[14,15[unterprogramm dann zurueck<·····································
   2.00 sprung[dc[c' in ordnung'<·····················································
   3.00 ·····························································
```

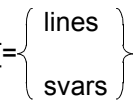
Text wird zusammen mit 3 Tabulatorzeichen eingegeben.

```
   1.00          BALR  14,15                  UNTERPROGRAMM DANN ZURUECK<·····
   2.00 SPRUNG   DC    C  ' IN ORDNUNG'<·····································
   3.00 ·····························································
```

Es werden Tabulatorexpansionen auf die Spalten 10, 16 und 40 durchgeführt.

9.133 @TABS (Format 3) – Softwaretabulatoren in Arbeitsdateien expandieren

Mit dem Format 3 der @TABS-Anweisung werden Softwaretabulatoren in Arbeitsdateien und Zeichenfolgevariablen expandiert, falls ein Tabulatorzeichen und entsprechende Tabulatorpositionen definiert sind (siehe @TABS, Format 2).

Operation	Operanden	F-Modus, L-Modus
@TABS	**R**ANGE [=  [,...]]	

lines Einer oder mehrere Zeilenbereiche, in denen die Tabulatorzeichen expandiert werden sollen.

svars Einer oder mehrere Bereiche von Zeichenfolgevariablen, in denen die Tabulatorzeichen expandiert werden sollen.

Ist weder `lines` noch `svars` angegeben, werden die Tabulatorzeichen in allen Zeilen der aktuellen Arbeitsdatei expandiert.

Ist kein Softwaretabulator definiert, wird die Anweisung mit der Fehlermeldung `EDT4953` abgewiesen. Der Softwaretabulator muss aber nicht aktiv sein.

Überschreitet eine Zeile oder Zeichenfolgevariable durch die Tabulatorexpansion die maximale Länge von 32768 Zeichen, wird die Tabulatorexpansion abgebrochen und die Meldung `EDT4314` ausgegeben. Eine Überprüfung der durch @TABS (Format 2) bzw. @CHECK (Format 1) festgelegten Maximallänge erfolgt nicht.

Ist für den Software-Tabulator die *Positionier-Strategie* mit Prüfung (CHECK in Format 2) eingestellt und würde eine *Rückwärts-Positionierung* erfolgen, wird die Tabulatorexpansion abgebrochen und die Meldung `EDT4312` ausgegeben.

9.134 @TMODE – Prozesseigenschaften ausgeben

Mit der Anweisung @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 (YYYY-MM-DD)
TIME	Zeit
ANWEISUNGSSYMBOL	aktuelles Anweisungssymbol (z.B. @)
TERMINAL	Typ des Bildschirms

Beispiel

```
  23.00 ·····························································
tmode·····························································0000.00:00001(00)
```

Informationen über die Prozesseigenschaften werden angefordert.

```
  22.00 ·····························································
%  EDT0300 0582 BOND     007         15.1635 2005-09-27 15:12:19 @ 9763
···························································0000.00:00001(00)
```

9.135 @UNLOAD – Entladen eines Moduls

Mit der Anweisung @UNLOAD können Ladeeinheiten oder einzelne Module, die mit @USE oder @RUN geladen wurden, wieder entladen werden. Gleichzeitig werden Anweisungsroutinen, die der entladenen Einheit zugewiesen sind, deaktiviert und deren Benutzeranweisungssymbole ungültig gesetzt.

Operation	Operanden	F-Modus, L-Modus
@UNLOAD	{ **U**NIT=entry **M**ODULE=entry (name) }	

UNIT=entry Die Ladeeinheit (`UNIT`) mit dem Namen `entry` soll komplett entladen werden. Der EDT teilt beim Laden von externen Anweisungsroutinen mit @USE bzw. von Anwenderroutinen mit @RUN dem Binder-Lader-System den Namen des angegebenen Moduls bzw. des Einsprungpunkts (`ENTRY`) als `UNIT`-Name mit. Durch Angabe dieses Namens bei @UNLOAD wird die komplette Ladeeinheit, einschließlich aller evtl. durch Autolink nachgeladenen Module entladen.

MODULE=entry
Der Modul mit dem Namen `entry` soll entladen werden. Im Gegensatz zum Verhalten bei Angabe von `UNIT` wird hier nur der angegebene Modul entladen.

name Name des Moduls, der entladen werden soll. Dieses Format wird nur noch aus Kompatibilitätsgründen unterstützt.

Wenn die Anweisung @UNLOAD zum Entladen einer Ladeeinheit bzw. eines Moduls geführt hat, bereinigt der EDT anschließend die Liste der vereinbarten Anweisungsroutinen. Dabei wird eine Anweisungsroutine nur über die bei `entry` bzw. `name` angegebene Zeichenkette identifiziert, ohne die Ladestruktur zu berücksichtigen. Alle Anweisungsroutinen mit dem angegebenen Namen werden deaktiviert und deren Benutzeranweisungssymbole werden ungültig gesetzt. Anweisungsroutinen, die mit @USE ...,ENTRY=* definiert wurden, werden dabei nicht berücksichtigt, d.h. das entsprechende Benutzeranweisungssymbol bleibt gültig und die entladene Ladeeinheit wird beim nächsten Aufruf mit dem entsprechenden Einsprungpunkt ggf. neu geladen.

Die Anweisung @UNLOAD gehört zu den sicherheitsrelevanten Anweisungen des EDT (siehe hierzu auch Abschnitt „Zugriffsschutz“ auf Seite 103). In nichtunterbrechbaren Systemprozeduren im Dialog und bei Eingabe von einer Datei (Lesen mit `RDATA` von `SYSDTA` ungleich `SYSCMD`, Abarbeiten einer Start-Prozedur) wird die Anweisung abgewiesen.

Der Versuch, Module zu entladen, die intern vom EDT nachgeladen wurden, wird mit der Meldung `EDT1907` abgewiesen.

Die gleiche Meldung wird auch ausgegeben, wenn der Modul aus anderen Gründen nicht entladen werden kann. Mögliche Ursachen sind ein falscher Modulname, die Angabe eines Moduls, der mehrfach benutzbar geladen ist sowie die Angabe eines `CSECT`- oder `ENTRY`-Namens, der bei der @USE-bzw. @RUN-Anweisung nicht zu einem Ladevorgang geführt hat.

Hinweis

Bei Angabe von `entry` sind bis zu 32 Zeichen lange Namen erlaubt, Groß-und Kleinschreibung wird unterschieden. Für `name` sind nur 8 Zeichen erlaubt, eingegebene Kleinbuchstaben werden in Großbuchstaben umgewandelt.

Achtung

Der EDT löscht bei @UNLOAD nur die Benutzeranweisungssymbole, denen die angegebene Ladeeinheit bzw. der angegebene Modul direkt zugewiesen war. Benutzeranweisungssymbole, die sich auf andere Einsprungpunkte (`ENTRY`) innerhalb der Ladeeinheit beziehen, bleiben erhalten und verweisen anschließend auf ungültige Adressen. Der Anwender ist selbst dafür verantwortlich, dass derartige Benutzeranweisungssymbole nach dem Entladen nicht mehr verwendet werden.

9.136 @UNSAVE – SAM- oder ISAM-Datei löschen

Die Anweisung @UNSAVE löscht eine SAM- oder ISAM-Datei und ihren Katalogeintrag.

Operation	**Operanden**	**F-Modus, L-Modus**
@UNSAVE	file [(ver)]	

file Name der zu löschenden Datei. Der Name muss dem SDF-Datentyp `<filename 1..54>` entsprechen.

Die symbolische Bezeichnung `'/'` für eine Datei, für die der `LINK`-Name `EDTSAM` bzw. `EDTISAM` durch das `/SET-FILE-LINK`-Kommando vergeben wurde, ist hier nicht zulässig.

ver Versionsnummer der zu löschenden Datei. Stimmt die angegebene Versionsnummer nicht mit der Versionsnummer der Datei überein, wird die Anweisung mit der Meldung `EDT4985` abgewiesen.

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

9.137 @USE – Definieren externer Anweisungsroutinen

Mit der Anweisung @USE kann man Benutzeranweisungen definieren, indem man ein Benutzeranweisungssymbol und die zugehörige Anweisungsroutine definiert (siehe Handbuch Unterprogramm-Schnittstellen [1]).

Operation	**Operanden**	**F-Modus, L-Modus**
@USE	**C**OMMAND='[spec]' [,**E**NTRY=entry | * [,**M**ODLIB=modlib] ; name | * [,modlib])]	

spec — Sonderzeichen, das als Benutzeranweisungssymbol für eine externe Anweisungsroutine dient. Das Benutzeranweisungssymbol muss verschieden vom aktuellen EDT-Anweisungssymbol sein.

Für die spezielle Verwendung der externen Anweisungsroutine als Anweisungsfilter muss statt eines Benutzeranweisungssymbols die leere Zeichenfolge angegeben werden. Dies ist allerdings nur beim Aufruf des EDT als Unterprogramm zulässig (siehe Abschnitt „Spezialanwendung als Anweisungsfilter“ in [1]).

entry — Einsprungpunkt der externen Anweisungsroutine. Der Modul, der den Einsprungpunkt enthält, wird sofort geladen.

name — Einsprungpunkt der externen Anweisungsroutine. Die Operandensyntax mit Angabe von Klammern wird nur noch aus Kompatibilitätsgründen unterstützt.

* — Der Name des Einsprungpunktes der Anweisungsroutine wird bei der Eingabe der benutzerdefinierten Anweisung als Anweisungsname angegeben. Der Modul, der den Einsprungpunkt enthält, wird erst bei der erstmaligen Ausführung der benutzerdefinierten Anweisung geladen.
Ebenso wird der UNIT-Name, mit dessen Hilfe die Ladeeinheit wieder entladen werden kann (siehe unten), erst zu diesem Zeitpunkt gebildet.

modlib Name der Bibliothek, in der der Modul abgelegt ist, der den Einsprungpunkt enthält. Die Bibliothek muss existieren, sonst wird die Anweisung mit der Fehlermeldung EDT5372 abgewiesen.

Wird der Modul in der angegebenen Bibliothek nicht gefunden, wird zuerst in den alternativen Bibliotheken BLSLIBnn gesucht, dann in der privaten Tasklib bzw. in der System-Tasklib $TASKLIB.

Ist keine Bibliothek angegeben, wird zunächst in der privaten Tasklib und dann in der System-Tasklib $TASKLIB gesucht.

Bei Misserfolg wird die Fehlermeldung EDT5372 ausgegeben.

Es können maximal 5 verschiedene Benutzeranweisungssymbole vereinbart werden. Der Versuch, ein sechstes Benutzeranweisungssymbol zu vereinbaren, wird mit der Meldung EDT5373 abgewiesen.

Sind außer dem Benutzeranweisungssymbol keine weiteren Operanden angegeben, wird die mit dem angegebenen Benutzeranweisungssymbol zuvor definierte Anweisungsroutine deaktiviert.

Führt die Bearbeitung der @USE-Anweisung zu einem eigenen Ladevorgang, wird dem Binder-Lader-System ein zu dem bei entry bzw. name angegebenen Einsprungpunkt gleich lautender UNIT-Name bekannt gemacht.

Dieser UNIT-Name kann bei der @UNLOAD-Anweisung angegeben werden, um alle zusammen mit dem Einsprungpunkt geladenen Ladeeinheiten zu entladen. Wird der Einsprungpunkt bereits innerhalb einer anderen Ladeeinheit gefunden, führt @USE also nicht zu einem eigenen Ladevorgang, kann der Einsprungpunkt nur zusammen mit dieser Ladeeinheit entladen werden.

Wenn anstelle von entry oder name der Wert * angegeben wird, findet dieser Mechanismus erst bei Eingabe der benutzerdefinierten Anweisungen Anwendung. Das heißt, dem Binder-Lader-System werden als UNIT-Namen die Namen aller Einsprungpunkte bekannt gemacht, die zu einem eigenen Ladevorgang geführt haben.

Der EDT lädt einen Einsprungpunkt nur dann nach, wenn dieser nicht schon in einer vorher geladenen Ladeeinheit gefunden wurde. Dies gilt auch, wenn der EDT selbst als Unterprogramm eines Benutzerprogramms geladen wurde, für Einsprungpunkte in diesem Benutzerprogramm.

Es ist also nicht möglich, Einsprungpunkte gleichen Namens in unterschiedlichen Ladeeinheiten parallel zu benutzen. Trotzdem kann es vorkommen, dass beim Laden einer Anweisungsroutine vom Binder-Lader-System doppelte Namen entdeckt werden. In diesem Fall wird die @USE-Anweisung bzw. die Benutzeranweisung, die zum Nachladen geführt hat, mit der Meldung EDT4208 abgewiesen. Wenn der EDT sich nicht im F-Modus befindet, wird zusätzlich noch die Fehlermeldung des Binder-Lader-Systems ausgegeben, die es erlaubt, den Namen des doppelt vergebenen Symbols zu identifizieren.

Man kann diese Situation vermeiden, wenn man die mit einer früheren @USE-Anweisung geladene Ladeeinheit vor dem Absetzen einer weiteren @USE-Anweisung zuerst mit der Anweisung @UNLOAD entlädt. Das zugehörige Benutzeranweisungssymbol wird dann ebenfalls zurückgenommen.

Wird das Benutzeranweisungssymbol lediglich mittels @USE COMMAND='spec' zurückgenommen, oder wird ein Benutzeranweisungssymbol zum zweiten Mal zugewiesen, findet demgegenüber kein implizites Entladen der ursprünglich zugewiesenen Ladeeinheit statt.

Werden in der @USE-Anweisung EDT-eigene Einsprungpunkte oder Modulnamen angegeben, so wird die @USE-Anweisung mit der Meldung `EDT4933` abgewiesen.

Bei Angabe des Operanden `ENTRY=*` bzw. (`*,modlib`) wird bei der Bildung des Namens des Einsprungpunktes aus dem ersten Teil der Benutzeranweisung grundsätzlich eine Umwandlung in Großbuchstaben vorgenommen, der Rest der Benutzeranweisung wird in Abhängigkeit der Einstellung bei @PAR LOW in Großbuchstaben umgewandelt oder nicht. Bei Angabe von (`*,modlib`) werden zur Bildung des Namens des Einsprungpunktes aus Kompatibilitätsgründen nur bis zu 8 Zeichen berücksichtigt, sonst bis zu 32 Zeichen.

Einzelheiten zur Implementierung externer Anweisungsroutinen oder Anweisungsfilter und zur Parameterübergabe an diese sind im Abschnitt „Aufruf einer benutzerdefinierten Anweisung“ in [1] beschrieben.

Die Anweisung @USE gehört zu den sicherheitsrelevanten Anweisungen des EDT (siehe hierzu auch den Abschnitt „Zugriffsschutz“ auf Seite 103). In bestimmten privilegierten Kennungen wird die Anweisung @USE abgewiesen. Dies trifft auch für nichtunterbrechbare Systemprozeduren im Dialogbetrieb bzw. bei der Eingabe von einer Datei (Lesen mit `RDATA` von `SYSDTA`, Abarbeiten einer EDT-Startprozedur) zu, es sei denn, die Anweisung @USE wird von der geschützten Prozedur selbst gegeben (`SYSDTA=SYSCMD`).

Hinweis

Bei Angabe von `entry` sind bis zu 32 Zeichen lange Namen erlaubt, Groß- und Kleinschreibung wird unterschieden. Für `name` sind nur 8 Zeichen erlaubt, eingegebene Kleinbuchstaben werden in Großbuchstaben umgewandelt.

Beispiel 1

Der Modul, der den Einsprungpunkt JOBVAR in der Bibliothek PRIVLIB enthält, kann mit den Parametern CATJV <name>, ERAJV <name>, GETJV <name>,<ln> bzw. SETJV <name>,<ln> aufgerufen werden. Der Name des Einsprungpunkts JOBVAR wird direkt in der @USE-Anweisung angegeben.

```
@USE COMMAND = '*',ENTRY=JOBVAR,MODLIB=PRIVLIB ----------------------------(1)
*CATJV JV.TEST ------------------------------------------------------------ (2)
*SETJV JV.TEST,3 ---------------------------------------------------------- (3)
```

(1) Das Benutzeranweisungssymbol * wird vereinbart und der Modul JOBVAR wird geladen.

(2) Der Modul JOBVAR wird mit dem Parameter 'CATJV JV.TEST' aufgerufen.

(3) Der Modul JOBVAR wird mit dem Parameter 'SETJV JV.TEST,3' aufgerufen.

Beispiel 2

In der Bibliothek PRIVLIB befinden sich die beiden Module SORT und HELP. Der Name des Einsprungpunkts ist nich definiert, bis die benutzerdefinierte Anweisung tatsächlich aufgerufen wird..

```
@USE COMMAND = '*',ENTRY=*,MODLIB=MODLIB ------------------------------- (1)
*SORT 20-100 -------------------------------------------------------------- (2)
*HELP EDT5100 ------------------------------------------------------------- (3)
```

(1) Das Benutzeranweisungssymbol * wird vereinbart. Es wird aber noch kein Modul geladen.

(2) Der Modul SORT wird geladen und mit dem Parameter '20-100' aufgerufen.

(3) Der Modul HELP wird geladen und mit dem Parameter 'EDT5100' aufgerufen.

9.138 @VDT – Bildschirmformat steuern

Mit der Anweisung @VDT kann im F-Modus das Bildschirmformat eingestellt werden.

Operation	Operanden	F-Modus, L-Modus
@VDT	[{**F1** / **F2** / **F3** / **F4**}]	

F1 Stellt das Bildschirmformat auf 24 Zeilen und 80 Spalten. Beim Starten des EDT bzw. wenn kein Operand angegeben ist, wird dieses Format ebenfalls eingestellt.

F2 Stellt das Bildschirmformat auf 27 Zeilen und 132 Spalten.

F3 Stellt das Bildschirmformat auf 32 Zeilen und 80 Spalten.

F4 Stellt das Bildschirmformat auf 43 Zeilen und 80 Spalten.

Die Bildschirmformate `F2`, `F3` und `F4` werden nur von der DSS 9763 unterstützt. Wenn die DSS ein angegebenes Format nicht unterstützt, wird die Anweisung mit der Fehlermeldung `EDT4945` abgewiesen.

Wird die Anweisung @VDT in einer Anweisungsfolge oder im `BLOCK`-Modus in einem Anweisungsblock eingegeben, wird sie nach Abarbeitung aller anderen Anweisungen als letzte ausgeführt.

Die Anweisung @VDT beendet implizit die Darstellung mit zwei Arbeitsfenstern.

Im Stapelbetrieb wird die Anweisung ohne Fehlermeldung ignoriert. Im L-Modus und bei Ausgaben nach `SYSOUT` (wenn `SYSOUT` einer Datensichtstation zugewiesen ist) ist immer das Bildschirmformat `F1` eingestellt.

9.139 @VTCSET – Bildschirmausgabe steuern

Die Anweisung @VTCSET bestimmt, ob bei der Ausgabe nach SYSOUT (z.B. durch eine der Anweisungen @PRINT oder @ON..PRINT) die Line-Modus-Steuerzeichen, die eventuell in den auszugebenden Datenzeilen enthalten sind, unverändert übermittelt werden oder in Schmierzeichen umgesetzt werden.

Operation	Operanden	F-Modus, L-Modus
@VTCSET	[{**ON** / **OF**F}]	

ON Legt fest, dass die Ausgabe ungeprüft erfolgt und daher keine Ersetzung durch Schmierzeichen erfolgt. Dies ist auch die Einstellung beim Start des EDT.

OFF Bewirkt, dass Line-Modus-Steuerzeichen in Datenzeilen bei der Ausgabe nach SYSOUT durch Schmierzeichen ersetzt werden.

Als Schmierzeichen wird das mit /MODIFY-TERMINAL-OPTIONS SUBSTITUTE-CHARACTER=... festgelegte Zeichen verwendet.

Die Einstellung @VTCSET OFF ist sinnvoll, wenn durch die als Steuerzeichen interpretierten Daten die Bildschirmausgabe zerrissen wird, dies kann nur bei Ausgaben nach SYSOUT geschehen, wenn SYSOUT der Datensichtstation zugewiesen ist. Für die formatierte Ausgabe des Arbeitsfensters im F-Modus oder für Ausgaben nach SYSLST (@LIST) bzw. im Stapelbetrieb hat die Anweisung @VTCSET keine Auswirkungen.

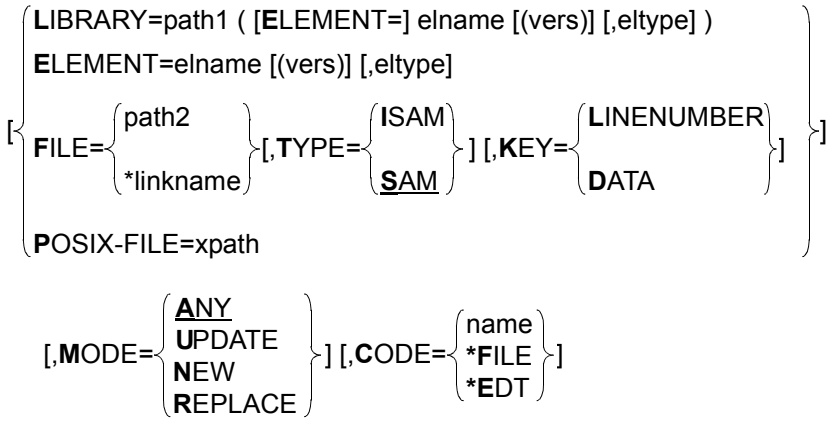
Ausgaben nach SYSOUT erfolgen generell im für SYSOUT eingestellten Zeichensatz (entweder dem der Datensichtstation oder dem der Datei bzw. des Bibliothekselements, dem SYSOUT zugewiesen ist). Die Ausgabe wird also evtl. umcodiert.

Eine Ersetzung der Line-Mode-Steuerzeichen (siehe Handbuch Makroaufrufe an den Ablaufteil [12], Makro VTCSET) wird nach dieser Umcodierung vorgenommen.
Da alle vom EDT unterstützten Datensichtstationen nur Zeichensätze unterstützen, bei denen die Steuerzeichen binär gleich codiert sind, spielt die Reihenfolge der Zeichenersetzung nur für die Darstellung des Schmierzeichens eine Rolle.

9.140 @WRITE (Format 1) – Schreiben einer Datei

Mit der Anweisung @WRITE (Format 1) wird eine Datei neu erzeugt und der Inhalt der aktuellen Arbeitsdatei in die neue Datei geschrieben, eine existierende Datei mit dem Inhalt der aktuellen Arbeitsdatei überschrieben oder der Inhalt der aktuellen Arbeitsdatei in eine mit @OPEN (Format 1) geöffnete Datei zurück geschrieben. Eine geöffnete Datei bleibt bei @WRITE geöffnet. Beim Überschreiben existierender Dateien wird der alte Dateiinhalt vollständig ersetzt. In allen Fällen bleibt die Arbeitsdatei erhalten.

Wenn in diesem Abschnitt von Datei die Rede ist, dann kann dies eine SAM-Datei, eine ISAM-Datei, ein Bibliothekselement oder eine POSIX-Datei sein.

Operation	Operanden	F-Modus, L-Modus
@WRITE		

LIBRARY=... Ein Bibliothekselement soll geschrieben werden. Dieses wird durch explizite Angabe des Bibliotheksnamens und der Elementbezeichnung bestimmt.

path1 Name der Bibliothek.

elname Name des Elements.

vers Version des gewünschten Elements (siehe Handbuch LMS [14]). Wird `vers` nicht angegeben oder wird `*STD` angegeben, wird die höchste vorhandene Version des Elementes gewählt.

eltype Typ des Elements. Zulässige Typangaben sind `S`, `M`, `P`, `J`, `D`, `X`, `*STD` und freie Typnamen mit entsprechendem Basistyp. Wird `eltype` nicht angegeben, wird der mit @PAR ELEMENT-TYPE voreingestellte Wert verwendet. Die zulässigen Elementtypen und deren Bedeutung sind im Kapitel „Dateibearbeitung“ auf Seite 137 beschrieben.

ELEMENT=... Ein Bibliothekselement soll geschrieben werden. Dieses wird durch die Elementbezeichnung ohne Angabe des Bibliotheknamens bestimmt. Es wird implizit die mit @PAR LIBRARY voreingestellte Bibliothek verwendet (sofern @PAR LIBRARY spezifiziert wurde, andernfalls wird die Fehlermeldung EDT5181 ausgegeben).

Die Operanden elname, vers und eltype haben die gleiche Bedeutung wie bei expliziter Angabe der Bibliothek (siehe oben).

FILE= Eine BS2000-Datei soll geschrieben werden.

path2 Name der BS2000-Datei (voll qualifizierter Dateiname), die geschrieben werden soll.

*linkname Dateikettungsname der BS2000-Datei, die geschrieben werden soll. Der Dateiname und die Dateiattribute sind in der Task File Table abgelegt. Dateien mit vom Standard abweichenden Attributen können so neu angelegt werden. Der Dateikettungsname darf nicht mit den Spezial-Dateinamen *BY-PROGRAM vereinbart worden sein. Dies führt zum Fehler EDT4923. Ist der Dateikettungsname nicht definiert, wird die Anweisung mit dem Fehler EDT5480 abgewiesen.

Ist der Dateikettungsname mit dem Spezial-Dateinamen *DUMMY vereinbart worden, wird dies wie eine nicht existierende Datei behandelt. Es entsteht aber keine Datei.

TYPE= Legt die Zugriffsmethode beim Neuanlegen einer Datei fest. Für existierende Dateien wird der Operand ignoriert.

SAM Eine SAM-Datei wird angelegt und geschrieben. Dies ist der Standardwert.

ISAM Eine ISAM-Datei wird angelegt und geschrieben.

KEY= Legt für ISAM-Dateien fest, wie der ISAM-Schlüssel gebildet werden soll. Für andere Dateitypen wird der Operand ignoriert.

Ist der Operand nicht angegeben, wird beim Schreiben einer neuen Datei oder beim Überschreiben einer existierenden Datei der ISAM-Schlüssel aus der Zeilennummer gebildet. Beim Zurückschreiben in eine geöffnete Datei wird der ISAM-Schlüssel aus der Zeilennummer gebildet, wenn beim Öffnen der Datei KEY=LINENUMBER oder KEY=IGNORE angegeben wurde. Wurde beim Öffnen KEY=DATA angegeben, wird der ISAM-Schlüssel aus dem Datenbereich übernommen.

LINENUMBER
: Der ISAM-Schlüssel wird aus der Zeilennummer gebildet. Weicht die Schlüsselposition vom Standard ab oder ist die Schlüssellänge zu groß, wird die Meldung EDT5465 ausgegeben und die Datei nicht geschrieben. Ist die Schlüssellänge zu klein, wird die Zeilennummer von links her verkürzt.

DATA
: Der ISAM-Schlüssel ist Bestandteil des Datenbereichs der Arbeitsdatei. In diesem Fall muss der Benutzer sicherstellen, dass die Reihenfolge der Arbeitsdateisätze der Reihenfolge der ISAM-Schlüssel entspricht, da sonst das Schreiben mit der Meldung EDT4208 (DVS-Fehlercode 0AAB) abgewiesen wird.

POSIX-FILE=
: Es soll eine POSIX-Datei geschrieben werden.

xpath
: Pfadname der POSIX-Datei, die geschrieben werden soll.

  Der Operand xpath kann auch als Zeichenfolgevariable angegeben werden. Er muss als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

MODE=
: Legt fest, ob die Datei schon vorhanden sein soll. Beim Zurückschreiben geöffneter Dateien wird der Operand ignoriert.

ANY
: Falls die Datei schon existiert, wird sie überschrieben, andernfalls wird sie neu angelegt und geschrieben. Dies ist der Standardwert.

UPDATE
: Die Datei, die geschrieben werden soll, muss bereits existieren, andernfalls wird je nach Dateityp die Meldung EDT5281, EDT5270 oder EDT5310 ausgegeben. Der alte Inhalt wird vollständig überschrieben.

NEW
: Die Datei wird neu angelegt und geschrieben. Sie darf noch nicht vorhanden sein, andernfalls wird je nach Dateityp die Meldung EDT5258, EDT5273 oder EDT5311 ausgegeben.

REPLACE
: Hat die gleiche Bedeutung wie ANY. Falls die Datei schon existiert, wird sie überschrieben, andernfalls wird sie neu angelegt und geschrieben.

CODE=
: Der Operand steuert, in welchem Zeichensatz die Arbeitsdatei geschrieben werden soll.

  Ist der Operand nicht angegeben, wird für POSIX-Dateien der mit @PAR CODE eingestellte Zeichensatz und für andere Dateien der Zeichensatz der Arbeitsdatei verwendet. Unterscheidet sich für SAM-Dateien, ISAM-Dateien oder Bibliothekselemente der Zeichensatz einer existierenden Datei von dem der Arbeitsdatei, dann wird im Stapelbetrieb die Meldung EDT5457 ausgegeben und es wird nicht geschrieben. Im Dialogbetrieb wird folgende Abfrage ausgegeben:

```
%  EDT0915 CONVERT TO FILE CCS (&00)? REPLY (Y=YES; N=NO)?
```

Bei der Antwort Y wird vor dem Schreiben in den Zeichensatz der Datei umcodiert. Bei der Antwort N wird der Zeichensatz der Arbeitsdatei verwendet.

name — Zeichensatz, der beim Schreiben verwendet werden soll. Als name muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).

*FILE — Die Arbeitsdatei wird vor dem Schreiben in den Zeichensatz der existierenden SAM-Datei oder ISAM-Datei, des Bibliothekselements oder den beim Öffnen der POSIX-Datei verwendeten Zeichensatz umcodiert. War dieser Zeichensatz *NONE, wird EDF03IRV verwendet. Existiert die Datei noch nicht oder soll eine existierende POSIX-Datei überschrieben werden, wird die Meldung EDT1181 ausgegeben und der CODE-Operand ignoriert. Das Verhalten entspricht dann dem bei weggelassenem CODE-Operanden.

*EDT — Beim Schreiben wird der Zeichensatz der Arbeitsdatei verwendet, unabhängig davon, ob eine eventuell existierende Datei einen anderen Zeichensatz hat.

Beim Schreiben neuer Dateien oder beim Überschreiben existierender Dateien muss immer ein Dateinamensoperand angegeben werden. Beim Zurückschreiben mit @OPEN geöffneter Dateien kann der Dateinamensoperand weggelassen werden. Wird der Dateinamensoperand weggelassen und ist keine Datei geöffnet, wird die Anweisung mit der Meldung EDT5122 abgewiesen. Werden sämtliche Operanden der Anweisung weggelassen und ist keine Datei geöffnet, so wird die Anweisung als @WRITE (Format 2) interpretiert und nach einem @FILE-Eintrag gesucht (siehe @WRITE Format 2).

Ist die angegebene Datei nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Nach dem Schreiben wird für SAM-Dateien, ISAM-Dateien und Bibliothekselemente der verwendete Zeichensatz in den Katalog eingetragen. Ist dieser Zeichensatz EDF03IRV und existiert die zu schreibende Datei bereits mit dem Zeichensatz *NONE im Katalog, so behält sie diesen.

Wird die Arbeitsdatei vor dem Schreiben umcodiert und enthält sie Zeichen, die im Zeichensatz der zu schreibenden Datei ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht geschrieben und die Fehlermeldung EDT5453 ausgegeben. Der Benutzer kann dann ein Ersatzzeichen definieren oder den Zeichensatz für das Schreiben verändern und @WRITE erneut ausführen.

Enthält die Arbeitsdatei Zeilen, die für die zu schreibende Datei zu lang sind (z.B. bei fester Satzlänge der Datei) oder entstehen durch die Konvertierung solche Sätze (bei Unicode-Zeichensätzen möglich), dann wird der Schreibvorgang mit der Meldung EDT5444 abgebrochen.

Wird während der Verarbeitung einer geöffneten ISAM-Datei der Zeichensatz von dem oder in den Zeichensatz UTF16 geändert, bzw. geschieht dies implizit durch entsprechende Angabe des CODE-Operanden, kann eine durch @OPEN geöffnete Datei nicht zurückgeschrieben werden, da sich die Länge des Schlüsselfeldes dadurch ändern würde. In diesem Fall wird die @WRITE-Anweisung mit der Fehlermeldung EDT5468 abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Achtung

Da MODE=ANY voreingestellt ist, werden bereits existierende Dateien ohne Warnung überschrieben.

Beispiel

```
@WRITE LIBRARY=PROGLIB(ELEMENT=SYNT)
```

Die aktuelle Arbeitsdatei wird in das Element SYNT der Bibliothek PROGLIB geschrieben. Dabei wird die höchstmögliche Version und der mit @PAR ELEMENT-TYPE voreingestellte Typ verwendet.

```
@PAR LIBRARY=BIB1
@SET #SO2='PROC.PR'
@WRITE ELEMENT=.#SO2 (V01),J
```

Die aktuelle Arbeitsdatei wird in das Element mit dem Namen PROC.PR, der Version V01 und dem Elementtyp J in der Bibliothek BIB1 geschrieben.

```
@WRITE FILE=DATEI2,MODE=NEW,CODE=*EDT
```

Die SAM-Datei DATEI2 wird neu angelegt und die aktuelle Arbeitsdatei wird in die neue Datei geschrieben. Dabei wird der Zeichensatz der Arbeitsdatei verwendet.

```
@OPEN POSIX-FILE=/home/user1/test/data,CODE=UTF8
@WRITE ,MODE=ANY
```

Die aktuelle Arbeitsdatei wird in die geöffnete POSIX-Datei data in dem Verzeichnis /home/user1/test zurück geschrieben.

9.141 @WRITE (Format 2) – Schreiben einer SAM-Datei

Mit der Anweisung @WRITE (Format 2) wird der Inhalt der aktuellen Arbeitsdatei ganz oder teilweise als SAM-Datei auf Platte oder Band geschrieben.

Operation	Operanden	F-Modus, L-Modus
@WRITE	[file] [(ver)] [,] [lines[,...]] [:cols[,...]:] [**K**EY] [<**U**PDATE | **O**VERWRITE>]	

file — Name der SAM-Datei, die geschrieben werden soll. Der Name muss dem SDF-Datentyp <filename 1..54> entsprechen oder die spezielle Angabe '/' sein.

Existiert die Datei file noch nicht, wird sie vor dem Schreiben neu angelegt. Fehlt der Operand file, so wird, falls vorhanden, der explizite lokale @FILE-Eintrag, dann der globale @FILE-Eintrag (aus der @FILE-Anweisung) und zuletzt der implizite lokale @FILE-Eintrag (z.B. aus der @READ-Anweisung) als Dateiname herangezogen. Ist kein @FILE-Eintrag vorhanden, wird @WRITE mit der Meldung EDT5484 abgewiesen.

Ist die angegebene Datei nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

Wenn der Dateikettungsname EDTSAM einer Datei zugeordnet ist, genügt die Angabe '/', um diese Datei zu schreiben (siehe Kapitel „Dateibearbeitung“ auf Seite 137).

ver — Versionsnummer der zu überschreibenden Datei. Wird für eine existierende Datei die falsche Versionsnummer angegeben, wird die Anweisung mit der Meldung EDT4985 abgewiesen. Bei einer noch nicht existierenden Datei wird die Angabe ignoriert und grundsätzlich die Version 001 geschrieben.

lines — Einer oder mehrere Zeilenbereiche, die in die SAM-Datei geschrieben werden sollen. Werden dabei Zeilen mehrfach angegeben, so werden sie auch mehrfach geschrieben.

Wird lines nicht angegeben, wird die gesamte Datei geschrieben.

cols
: Einer oder mehrere Spaltenbereiche, durch die der zu schreibende Bereich jedes Satzes festgelegt wird. Wiederholungen und Überlappungen der Bereiche sind erlaubt. Die Spaltenangaben beziehen sich auf die *Zeichen* in der aktuellen Arbeitsdatei. Werden Spaltenwerte angegeben, die die Länge eines Arbeitsdateisatzes überschreiten, so werden dafür Leerzeichen in die Datei geschrieben.

  Wird kein Spaltenbereich angegeben, werden die Zeilen in voller Länge geschrieben.

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
: Die Angabe von UPDATE bewirkt, dass die abzuspeichernden Zeilen an das Ende der existierenden SAM-Datei angehängt werden.

  Dieser Operand wird ignoriert, wenn keine SAM-Datei mit dem angegebenen Namen existiert.

OVERWRITE
: Eine vorhandene Datei gleichen Namens wird ohne Rückfrage überschrieben. Existiert die angegebene Datei noch nicht, ist `OVERWRITE` wirkungslos.

Wird weder `UPDATE` noch `OVERWRITE` angegeben und existiert bereits eine Datei mit gleichem Namen, reagiert der EDT im Dialogbetrieb mit der Frage:

```
%  EDT0903 FILE 'file' IS IN THE CATALOG, FCBTYPE = fcbtyp
%  EDT0296 OVERWRITE FILE? REPLY (Y=YES; N=NO)?
```

Wird die Meldung mit `Y` beantwortet, wird die bestehende Datei als SAM-Datei mit dem Inhalt der aktuellen Arbeitsdatei überschrieben. Wird dagegen die Meldung mit `N` beantwortet, wird die Datei nicht geschrieben und die Meldung `EDT0293` ausgegeben.
Im Stapelbetrieb wird die Datei immer überschrieben.

Beim Überschreiben einer existierenden Datei durch @WRITE ohne den Operanden `UPDATE` ändern sich möglicherweise Datei-Typ und/oder Datei-Attribute. Die Datei wird als SAM-Datei mit Standardattributen (z.B. variable Satzlänge) geschrieben, sofern nicht vorher ein entsprechendes `/SET-FILE-LINK`-Kommando mit dem Dateikettungsnamen `EDTSAM` und den abweichenden Attributen gegeben worden ist (siehe Kapitel „Dateibearbeitung“ auf Seite 137). Dateien vom Typ `PAM` oder `BTAM` können nicht überschrieben werden.

Die Datei wird nur während des Schreibvorgangs temporär geöffnet.

Der Zeichensatz, mit dem das Schreiben erfolgt, hängt davon ab, ob die Datei überschrieben, neu angelegt oder erweitert wird (siehe Operand `UPDATE`).

Wird die Datei überschrieben oder neu angelegt, werden die Daten im Zeichensatz der Arbeitsdatei geschrieben und für die Datei wird dieser Zeichensatz im Katalog eingetragen.

Wird die Datei erweitert, werden die Daten vom Zeichensatz der Arbeitsdatei in den Zeichensatz umcodiert, der im Katalogeintrag der Datei spezifiziert ist. Ist im Katalog der Datei `*NONE` eingetragen, wird `EDF03IRV` angenommen (siehe auch Abschnitt „Zeichensätze“ auf Seite 48).

Enthält die Arbeitsdatei Zeichen, die im Zeichensatz der zu schreibenden Datei ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht geschrieben und die Fehlermeldung `EDT5453` ausgegeben. Dies gilt nicht für ungültige Zeichen außerhalb des zu schreibenden Zeilen- oder Spaltenbereichs. Diese Zeichen werden ignoriert.

Enthält die Arbeitsdatei Zeilen, die für die zu schreibende Datei zu lang sind (z.B. bei fester Satzlänge der Datei) oder entstehen durch die Konvertierung solche Sätze (bei Unicode-Zeichensätzen möglich), dann wird der Schreibvorgang mit der Meldung `EDT5444` abgebrochen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Werden sämtliche Operanden der Anweisung weggelassen und ist eine Datei in der aktuellen Arbeitsdatei geöffnet, so wird die Anweisung als @WRITE (Format 1) interpretiert und nicht nach einem @FILE-Eintrag gesucht (siehe @WRITE Format 1).

Achtung

Wenn mit der @FILE-Anweisung ein Dateiname voreingestellt wurde, dann führt die Anweisungsfolge

```
@READ 'dateiname'
@WRITE
```

nicht zum Schreiben der eingelesenen Datei. Vielmehr wird die Datei aus dem @FILE-Eintrag geschrieben.

Beispiel

```
1.     EINE GANZ KURZE DATEI ------------------------------------------ (1)
2.     @WRITE 'TEST.@WRITE.1' ----------------------------------------- (2)
2.     @FILE 'TEST.@WRITE.1' ------------------------------------------ (3)
2.     @WRITE UPDATE -------------------------------------------------- (4)
2.     @DELETE -------------------------------------------------------- (5)
1.     @READ ---------------------------------------------------------- (6)
3.     @PRINT
1.0000 EINE GANZ KURZE DATEI
2.0000 EINE GANZ KURZE DATEI ------------------------------------------ (7)
3.
```

(1) Zeile wird in die Arbeitsdatei geschrieben.

(2) Diese Zeile wird als Datei `TEST.@WRITE.1` auf Platte geschrieben.

(3) Der Dateiname `TEST.@WRITE.1` wird über @FILE vereinbart.

(4) @WRITE bezieht sich auf den unter (3) vereinbarten Dateinamen. Mit `UPDATE` erreicht man, dass der Inhalt der Arbeitsdatei – immer noch die unter (1) angelegte Zeile – an das Ende der Datei `TEST.@WRITE.1` angehängt wird.

(5) Der Inhalt der Arbeitsdatei wird gelöscht.

(6) Die Datei `TEST.@WRITE.1` wird in die Arbeitsdatei gebracht (auch hier braucht kein Dateiname angegeben werden.)

(7) Man sieht, dass unter (4) die Zeile an das Ende der Datei angehängt wurde.

9.142 @XCOPY – Einlesen einer POSIX-Datei

Mit der Anweisung @XCOPY kann eine POSIX-Datei, die im POSIX-Dateisystem abgelegt ist, in die aktuelle Arbeitsdatei eingelesen werden. Diese Anweisung wird nur noch aus Kompatibilitätsgründen unterstützt. Es wird empfohlen, stattdessen die Anweisung @COPY (Format 1) mit dem Operanden `POSIX-FILE` zu benutzen.

Operation	**Operanden**	**F-Modus, L-Modus**
@XCOPY	**F**ILE=xpath [,**C**ODE= { name / **E**BCDIC / **I**SO }]	

xpath — Pfadname der POSIX-Datei, die in die aktuelle Arbeitsdatei eingelesen werden soll.

Der Operand `xpath` kann auch als Zeichenfolgevariable angegeben werden. Er *muss* als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

CODE= — Es wird festgelegt, welcher Zeichensatz für die POSIX-Datei angenommen wird. Da POSIX-Dateien innerhalb des POSIX-Dateisystems kein Zeichensatz zugeordnet werden kann, ist hier eine Angabe des Anwenders erforderlich.

Wird `CODE` nicht angegeben, wird für die Datei der bei @PAR CODE eingestellte Zeichensatz angenommen. Nach dem Starten des EDT ist `EDF041` eingestellt.

- name — Zeichensatz der einzulesenden POSIX-Datei. Als `name` muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).
- EBCDIC — Das Schlüsselwort `EBCDIC` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `EDF041` unterstützt.
- ISO — Das Schlüsselwort `ISO` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `ISO88591` unterstützt.

Das Einfügen der aus der Datei gelesenen Sätze erfolgt in der aktuellen Arbeitsdatei hinter der letzten Zeile nach dem Verfahren Einfügen zwischen zwei Zeilen (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).

Ist die aktuelle Arbeitsdatei leer und hat den Zeichensatz *NONE, erhält die Arbeitsdatei den Zeichensatz, der durch CODE angegeben ist. Ist kein Zeichensatz angegeben, erhält die Arbeitsdatei den Zeichensatz, der mit @PAR CODE eingestellt ist.

Hat die Arbeitsdatei bereits einen Zeichensatz, dann werden die einzulesenden Sätze vom Zeichensatz der Datei in den Zeichensatz der Arbeitsdatei konvertiert.

Enthält die einzulesende Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht eingelesen und die Fehlermeldung EDT5453 ausgegeben.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird das Lesen mit der Meldung EDT5454 abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

9.143 @XOPEN – Öffnen und Einlesen einer POSIX-Datei

Mit der Anweisung @XOPEN kann eine POSIX-Datei, die im POSIX-Dateisystem abgelegt ist, geöffnet und in die aktuelle Arbeitsdatei eingelesen werden. Diese Anweisung wird nur noch aus Kompatibilitätsgründen unterstützt. Es wird empfohlen, stattdessen die Anweisung @OPEN (Format 1) mit dem Operanden `POSIX-FILE` zu benutzen.

Operation	Operanden	F-Modus, L-Modus
@XOPEN	**F**ILE=xpath [,**C**ODE={name / **E**BCDIC / **I**SO}] [,**M**ODE={**A**NY / **U**PDATE / **N**EW / **R**EPLACE}]	

xpath — Pfadname der POSIX-Datei, die geöffnet werden soll.

Der Operand `xpath` kann auch als Zeichenfolgevariable angegeben werden. Er *muss* als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

Existiert die angegebene Datei nicht oder ist sie nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

CODE= — Es wird festgelegt, welcher Zeichensatz für die POSIX-Datei angenommen wird. Da POSIX-Dateien innerhalb des POSIX-Dateisystems kein Zeichensatz zugeordnet werden kann, ist hier eine Angabe des Anwenders erforderlich.

Wird `CODE` nicht angegeben, wird für die Datei der mit @PAR CODE eingestellte Zeichensatz angenommen. Nach dem Starten des EDT ist `EDF041` eingestellt.

- name — Zeichensatz der zu öffnenden POSIX-Datei. Als `name` muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).
- EBCDIC — Das Schlüsselwort `EBCDIC` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `EDF041` unterstützt.
- ISO — Das Schlüsselwort `ISO` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `ISO88591` unterstützt.

MODE= Legt fest, ob die Datei schon vorhanden sein soll bzw. sein darf.

- ANY Falls die Datei schon existiert, wird sie zur Bearbeitung geöffnet und eingelesen, andernfalls wird sie neu angelegt und zur Bearbeitung geöffnet. Dies ist der Standardwert.
- UPDATE Die Datei, die zur Bearbeitung geöffnet und eingelesen werden soll, muss bereits existieren, andernfalls wird die Meldung EDT5310 ausgegeben.
- NEW Die Datei wird neu angelegt und zur Bearbeitung geöffnet. Sie darf noch nicht vorhanden sein, andernfalls wird die Meldung EDT5311 ausgegeben.
- REPLACE Falls die Datei schon existiert, wird sie zur Bearbeitung geöffnet, ihr alter Inhalt wird aber gelöscht und nicht in die Arbeitsdatei eingelesen. Falls die Datei nicht existiert, wird sie neu angelegt und zur Bearbeitung geöffnet.

Ist die aktuelle Arbeitsdatei nicht leer, wird die Anweisung @XOPEN mit der Meldung EDT5191 abgewiesen. Im Gegensatz zu BS2000-Dateien oder Bibliothekselementen können POSIX-Dateien mehrfach in verschiedenen Arbeitsdateien geöffnet sein (in POSIX gibt es keinen Schutz gegen das mehrfache Öffnen einer Datei durch unterschiedliche Prozesse).
Das Einfügen der aus der Datei gelesenen Sätze erfolgt in der aktuellen Arbeitsdatei hinter Position 0.0000 nach dem Verfahren Einfügen zwischen zwei Zeilen (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37).
Hat die leere Arbeitsdatei den Zeichensatz *NONE, erhält die Arbeitsdatei den Zeichensatz, der durch CODE angegeben ist. Ist kein Zeichensatz angegeben, erhält die Arbeitsdatei den Zeichensatz, der mit @PAR CODE eingestellt ist.

Hat die leere Arbeitsdatei bereits einen Zeichensatz (z.B. durch vorhergehendes @CODE-NAME), dann werden die einzulesenden Sätze vom Zeichensatz der Datei in den Zeichensatz der Arbeitsdatei konvertiert. Enthält die einzulesende Datei Zeichen, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht eingelesen und die Fehlermeldung EDT5453 ausgegeben.

Liegt die Datei in einem Unicode-Zeichensatz vor und enthält sie ungültige Bytefolgen, z.B. Surrogate-Zeichen, kann sie auch bei Angabe eines SUBSTITUTION-CHARACTERS nicht eingelesen werden. In diesem Fall wird das Lesen mit der Meldung EDT5454 abgewiesen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit /INFORM-PROGRAM fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung EDT5501 ausgegeben.

Hinweis

Ist bei Beenden des EDT (@HALT, @END, @RETURN) eine Datei mit @XOPEN geöffnet und wird die Sicherungsabfrage EDT0900 ausgegeben, so wird der POSIX-Dateiname in der Form X=xpath angezeigt.

9.144 @XWRITE – Inhalt der aktuellen Arbeitsdatei in eine POSIX-Datei speichern

Mit der Anweisung @XWRITE kann der Inhalt der aktuellen Arbeitsdatei in eine POSIX-Datei geschrieben werden. Die Arbeitsdatei bleibt dabei erhalten. Diese Anweisung wird nur noch aus Kompatibilitätsgründen unterstützt. Es wird empfohlen, stattdessen die Anweisung @WRITE (Format 1) mit dem Operanden `POSIX-FILE` zu benutzen.

Operation	Operanden	F-Modus, L-Modus
@XWRITE	[**F**ILE=xpath] [,**C**ODE={name / **E**BCDIC / **I**SO}] [,**M**ODE={**A**NY / **U**PDATE / **N**EW / **R**EPLACE}]	

xpath — Pfadname der POSIX-Datei, die geschrieben werden soll

Der Operand `xpath` kann auch als Zeichenfolgevariable angegeben werden. Er muss als Zeichenfolgevariable angegeben werden, wenn der Pfadname Zeichen enthält, die in der EDT-Syntax eine Sonderbedeutung haben (z.B. Leerzeichen, Semikolon im F-Modus oder Komma).

Wird `xpath` nicht angegeben, wird eine mit @XOPEN oder @OPEN (Format 1) geöffnete POSIX-Datei zurück geschrieben. Gibt es keine geöffnete POSIX-Datei, wird die Anweisung mit der Meldung `EDT5122` abgewiesen.

Ist die angegebene Datei nicht wie erforderlich zugreifbar, wird die Anweisung mit einer entsprechenden Fehlermeldung abgewiesen.

CODE= — Es wird festgelegt, in welchem Zeichensatz die POSIX-Datei geschrieben wird.

Wird `CODE` nicht angegeben, wird die Datei in dem mit @PAR CODE eingestellten Zeichensatz geschrieben (auch beim Zurückschreiben von Dateien, die mit einem anderen Zeichensatz geöffnet wurden). Nach dem Starten des EDT ist `EDF041` eingestellt.

name — Zeichensatz der zu schreibenden POSIX-Datei. Als `name` muss der Name eines gültigen Zeichensatzes angegeben werden (siehe Abschnitt „Zeichensätze“ auf Seite 48).

EBCDIC — Das Schlüsselwort `EBCDIC` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `EDF041` unterstützt.

ISO Das Schlüsselwort `ISO` wird nur noch aus Kompatibilitätsgründen als Synonym für den Zeichensatz `ISO88591` unterstützt.

MODE= Legt fest, ob die Datei schon vorhanden sein soll bzw. sein darf. Beim Zurückschreiben geöffneter Dateien wird der Operand ignoriert.

ANY Falls die Datei schon existiert, wird sie überschrieben, andernfalls wird sie neu angelegt und geschrieben. Dies ist der Standardwert.

UPDATE Die Datei, die geschrieben werden soll, muss bereits existieren, andernfalls wird die Meldung `EDT5310` ausgegeben.

NEW Die Datei wird neu angelegt und geschrieben. Sie darf noch nicht vorhanden sein, andernfalls wird die Meldung `EDT5311` ausgegeben.

REPLACE Hat die gleiche Bedeutung wie `ANY`. Falls die Datei schon existiert, wird sie überschrieben, andernfalls wird sie neu angelegt und geschrieben.

Wurde eine POSIX-Datei mit @XOPEN oder @OPEN (Format 1) geöffnet, so kann die Angabe des Dateinamens bei @XWRITE entfallen. Der Inhalt der geöffneten Datei wird durch den Inhalt der Arbeitsdatei ersetzt. Die Datei bleibt geöffnet, bis @CLOSE abgesetzt wird.

Wird die Arbeitsdatei vor dem Schreiben umcodiert und enthält sie Zeichen, die im Zeichensatz der zu schreibenden Datei ungültig sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER), andernfalls wird die Datei nicht geschrieben und die Fehlermeldung `EDT5453` ausgegeben. Der Benutzer kann dann ein Ersatzzeichen definieren oder den Zeichensatz für das Schreiben verändern und @XWRITE erneut ausführen.

Wird die Anweisung mit K2 unterbrochen und der EDT-Lauf mit `/INFORM-PROGRAM` fortgesetzt, so wird die Bearbeitung der Anweisung abgebrochen und die Meldung `EDT5501` ausgegeben.

Achtung

Da `MODE=ANY` voreingestellt ist, werden bei fehlender `MODE`-Angabe bereits existierende Dateien ohne Warnung überschrieben.

9.145 0..22 – Wechseln der Arbeitsdatei

Mit dieser Anweisung wechselt der EDT in eine andere Arbeitsdatei.

Operation	Operanden	F-Modus
0..22		

Der EDT zeigt die mit der Anweisung `0..22` ausgewählte Arbeitsdatei in dem Arbeitsfenster an, in dem die Anweisung eingegeben wurde.

Die Zeilen- und Spaltenposition wird auf die Werte eingestellt, die in der neu eingestellten Arbeitsdatei vorher gültig waren. Wurde die Arbeitsdatei vorher noch nicht benutzt, werden die Standardwerte eingestellt.

Mit der Anweisung @SETF kann man ebenfalls die Arbeitsdatei wechseln und gleichzeitig auf eine beliebige Zeile und Spalte positionieren.

10 Kurzanweisungen des F-Modus (alphabetisch)

Dieses Kapitel enthält die ausführliche Beschreibung aller Kurzanweisungen des F-Modus des EDT in alphabetischer Reihenfolge.

Für Kurzanweisungen, die aus einem Sonderzeichen bestehen, bedeutet *alphabetisch* die durch den Zeichensatz EBCDIC.DF.04 vorgegebene Ordnung.

10.1 + – Vorwärts positionieren des Arbeitsfensters

Die Kurzanweisung + positioniert das Arbeitsfenster so, dass die Zeile, in der + eingegeben wurde, zur ersten Zeile des Datenfensters wird.

Kurzanweisung	Taste
+	DUE oder F2

Die Spaltenposition wird durch diese Kurzanweisung nicht verändert.

10.2 + – Vorwärts positionieren des Arbeitsfensters nach der Strukturtiefe

Wird die Kurzanweisung + mit der Funktionstaste F1 übertragen, wird zum nächsten Satz positioniert, der die gleiche Strukturtiefe hat wie der angegebene.

Strukturtiefe ist der Abstand des ersten Zeichens ungleich eines Leerzeichens vom Satzbeginn. Dabei spielt es keine Rolle, ob der Datensatz ab Spalte 1 angezeigt wird oder der Bildschirmausschnitt nach rechts verschoben wurde.

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

Kurzanweisung	Taste
+	F1

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

Enthält der angegebene Satz kein Struktursymbol, wird die Kurzanweisung mit der Meldung EDT5354 abgewiesen.

Beispiel

Es wird vorausgesetzt, dass mit @PAR STRUCTURE=' ' das Leerzeichen als Struktursymbol definiert ist.

```
   1.00 if (b != 0)<·····························································
   2.00 {<·····································································
   3.00     if (a == 1)<·····························································
+  4.00     {<·····································································
   5.00          b = 0;<·····························································
   6.00          c = 3;<·····························································
   7.00     }<·····································································
   8.00     else<·····································································
   9.00     {<·····································································
```

```
·····························································0001.00:00001(00)
```

In Zeile 4.00 wird die Kurzanweisung + eingegeben und mit F1 abgeschickt.

```
  7.00     }<·························································
  8.00     else<·····················································
  9.00     {<·························································



··························································0007.00:00001(00)
```

Es wurde auf den nächsten Datensatz mit gleicher Strukturtiefe positioniert.

10.3 * – Löschen des Kopierpuffers

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

Kurzanweisung	Taste
*	[DUE] oder [F2]

Die Kurzanweisung * 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 in einer Zeile * eingegeben wird.

Das Löschen des Kopierpuffers wird mit der Meldung EDT0292 quittiert. Die Meldung unterbleibt, wenn durch gleichzeitige Eingabe der Kurzanweisungen C, M oder R neue Zeilennummern in den Kopierpuffer aufgenommen wurden.

10.4 – – Rückwärts positionieren des Arbeitsfensters

Die Kurzanweisung – positioniert das Arbeitsfenster so, dass die Datenzeile, in der – eingegeben wurde, zur letzten Zeile des Datenfensters wird.

Kurzanweisung	Taste
–	DUE oder F2

Die Spaltenposition wird durch diese Kurzanweisung nicht verändert.

Die Kurzanweisung – ist nicht wirksam, wenn nicht genug Datensätze existieren, um das Arbeitsfenster vollständig zu füllen.

10.5 – – Rückwärts positionieren des Arbeitsfensters nach der Strukturtiefe

Wird die Kurzanweisung – mit der Funktionstaste F1 übertragen, wird zum vorigen Satz positioniert, der die gleiche Strukturtiefe hat wie der angegebene.

Strukturtiefe ist der Abstand des ersten Zeichens ungleich eines Leerzeichens vom Satzbeginn. Dabei spielt es keine Rolle, ob der Datensatz ab Spalte 1 angezeigt wird oder der Bildschirmausschnitt nach rechts verschoben wurde.

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

Kurzanweisung	Taste
–	F1

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

Enthält der angegebene Satz kein Struktursymbol, wird die Kurzanweisung mit der Meldung EDT5354 abgewiesen.

Beispiel

Es wird vorausgesetzt, dass mit @PAR STRUCTURE=' ' das Leerzeichen als Struktursymbol definiert ist.

```
   1.00 if (b != 0)<·····················································
   2.00 {<·······························································
   3.00     if (a == 1)<·················································
   4.00     {····························································
   5.00          b = 0;<·················································
   6.00          c = 3;<·················································
-  7.00     }<···························································
   8.00     else<························································
   9.00     {<···························································



·······························································0001.00:00001(00)
```

In Zeile `7.00` wird die Kurzanweisung – eingegeben und mit [F1] abgeschickt.

```
   4.00     {<···························································
   5.00          b = 0;<·················································
   6.00          c = 3;<·················································
   7.00     }<···························································
   8.00     else<························································
   9.00     {<···························································



·······························································0004.00:00001(00)
```

Es wurde auf den vorigen Datensatz mit gleicher Strukturtiefe positioniert.

10.6 A – Kopieren oder Verschieben hinter eine Zeile

Die Kurzanweisung A kopiert oder verschiebt Datensätze, deren Zeilennummern mit C, M oder R im Kopierpuffer aufgesammelt wurden, *hinter* (after) die angegebene Zeile. Wurde der Kopierpuffer mit den Kurzanweisungen C oder M gefüllt, wird er anschließend gelöscht.

Kurzanweisung	Taste
A	DUE oder F2

Im Folgenden wird der Einfachheit halber immer von *Kopieren* gesprochen, auch wenn die Zeilen nach dem Kopieren gelöscht, d.h. verschoben werden.

Ist der Kopierpuffer leer, wird die Anweisung mit der Meldung EDT5376 abgewiesen.

Beim Ein- oder Anfügen der kopierten Zeilen vergibt der EDT die Zeilennummern nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Ist ein Einfügen nicht möglich, unterbleibt der Kopiervorgang, der Kopierpuffer wird nicht gelöscht und die Meldung EDT5365 wird ausgegeben.

Ist die aktuelle Arbeitsdatei leer und hat sie den Zeichensatz *NONE, dann erhält sie beim Kopieren den Zeichensatz der Quell-Arbeitsdatei der ersten zu kopierenden Zeile.

Hat die aktuelle Arbeitsdatei einen Zeichensatz, dann werden die zu kopierenden Zeilen in den Zeichensatz der aktuellen Arbeitsdatei konvertiert.

Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER). Andernfalls wird die Kurzanweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Hinweis

Die Kurzanweisung A wird erst nach Abarbeitung von C-, M- und R-Anweisungen ausgeführt, so dass in einem Arbeitsfenster in einem Dialogschritt der Zielort auch vor den zu kopierenden Zeilen angegeben werden kann.

Beispiel

```
   1.00 BERGER   ADALBERT  HOCHWEG 10     81234 MUENCHEN<·····················
c  2.00 HOFER    LUDWIG    GANGGASSE 3A   80123 MUENCHEN<·····················
a  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 hinter die Zeile 3.00 kopiert werden.

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

Die Zeile `2.00` wurde als neue Zeile `3.10` hinter Zeile `3.00` kopiert.

10.7 B – Kopieren oder Verschieben vor eine Zeile

Die Kurzanweisung B kopiert oder verschiebt Datensätze, deren Zeilennummern mit C, M oder R im Kopierpuffer aufgesammelt wurden, *vor* (before) die angegebene Zeile. Wurde der Kopierpuffer mit den Kurzanweisungen C oder M gefüllt, wird er anschließend gelöscht.

Kurzanweisung	Taste
B	DUE oder F2

Im Folgenden wird der Einfachheit halber immer von *Kopieren* gesprochen, auch wenn die Zeilen nach dem Kopieren gelöscht, d.h. verschoben werden.

Ist der Kopierpuffer leer, wird die Anweisung mit der Meldung EDT5376 abgewiesen.

Beim Ein- oder Anfügen der kopierten Zeilen vergibt der EDT die Zeilennummern nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Ist ein Einfügen nicht möglich, unterbleibt der Kopiervorgang, der Kopierpuffer wird nicht gelöscht und die Meldung EDT5365 wird ausgegeben.

Wurde die Kurzanweisung B in der ersten Bildschirmzeile eingegeben, wird der Bildschirm anschließend so repositioniert, dass die eingefügten Zeilen sichtbar sind.

Ist die aktuelle Arbeitsdatei leer und hat sie den Zeichensatz *NONE, dann erhält sie beim Kopieren den Zeichensatz der Quell-Arbeitsdatei der ersten zu kopierenden Zeile.

Hat die aktuelle Arbeitsdatei einen Zeichensatz, dann werden die zu kopierenden Zeilen in den Zeichensatz der aktuellen Arbeitsdatei konvertiert. Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER). Andernfalls wird die Kurzanweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Hinweis

Die Kurzanweisung B wird erst nach Abarbeitung von C-, M- und R-Anweisungen ausgeführt, so dass in einem Arbeitsfenster in einem Dialogschritt der Zielort auch vor den zu kopierenden Zeilen angegeben werden kann.

10.8 C – Aufsammeln von Zeilen zum Kopieren

Die Kurzanweisung C überträgt Zeilen- und Arbeitsdateinummer des angegebenen Satzes in den Kopierpuffer des EDT, um den Satz später mit einer der Kurzanweisungen A, B oder O zu kopieren.

Kurzanweisung	Taste
C	DUE oder F2

Sobald eine der Kurzanweisungen A, B oder O eingegeben wurde, wird der Kopiervorgang ausgeführt, d.h. die Sätze werden an der entsprechenden Stelle eingefügt. Anschließend wird der Inhalt des Kopierpuffers gelöscht.

Enthält bei Angabe von C der Kopierpuffer schon Einträge, die durch die Kurzanweisungen M oder R erzeugt wurden, so wird der Kopierpuffer gelöscht, bevor die angegebene Zeile eingetragen wird; die Meldung EDT0295 wird ausgegeben.

Der Kopierpuffer kann auch durch Aufsammeln von Zeilen aus verschiedenen Arbeitsdateien aufgebaut werden. Die durch den Kopierpuffer bestimmten Sätze können dann in jede Arbeitsdatei kopiert werden.

Der Kopierpuffer enthält Arbeitsdatei- und Zeilennummer der mit C aufgesammelten Sätze. Die Zeilennummern und der Inhalt der zu kopierenden Sätze sollten daher zwischen dem Aufsammeln mit C und der Ausführung des Kopiervorgangs durch Eingabe der Kurzanweisungen A, B oder O nicht verändert werden. Geschieht dies doch, werden die neuen Inhalte kopiert. Wurden zwischenzeitlich Sätze gelöscht, werden diese ohne Warnung beim Kopieren übergangen.

Da der Kopierpuffer nur Arbeitsdatei- und Zeilennummer enthält, findet zum Zeitpunkt des Aufsammelns auch noch keine Umcodierung der ausgewählten Zeilen statt. Da der Ziel-Zeichensatz erst durch die Festlegung der Ziel-Arbeitsdatei bestimmt wird, findet eine eventuell nötige Umcodierung erst bei Auswertung der Kurzanweisungen A, B bzw. O statt (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Hinweis

Bei einem geteilten 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.

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin<··········································
c  3.00 <···············································································
   4.00 Und leider auch Theologie<·····································
   5.00 Durchaus studiert, mit heißem Bemühn.<···························
   6.00 Da steh ich nun ich armer Tor!<·····································
b  7.00 Und bin so klug als wie zuvor;<·····································
a  8.00 Heiße Magister, heiße Doktor gar<·····························
   9.00 Und ziehe schon an die zehen Jahr<·····························
  10.00 ···············································································
```

Die Zeile `3.00` soll vor die Zeile `7.00` und hinter die Zeile `8.00` kopiert werden. Dazu wird in Zeile `3.00` die Kurzanweisung `C`, in Zeile `7.00` die Kurzanweisung `B` und in Zeile `8.00` die Kurzanweisung `A` eingegeben.

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin<··········································
   3.00 <···············································································
   4.00 Und leider auch Theologie<·····································
   5.00 Durchaus studiert, mit heißem Bemühn.<···························
   6.00 Da steh ich nun ich armer Tor!<·····································
   6.10 <···············································································
   7.00 Und bin so klug als wie zuvor;<·····································
   8.00 Heiße Magister, heiße Doktor gar<·····························
   9.00 Und ziehe schon an die zehen Jahr<·····························
  10.00 ···············································································
  21.00 ···············································································
%  EDT5360 NO COPY. BUFFER EMPTY
·····························································0001.00:00001(00)
```

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` aufgesammelte 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` aufzusammeln.

10.9 D – Löschen von Sätzen

Die Kurzanweisung `D` löscht den angegebenen Satz aus der Arbeitsdatei.

Kurzanweisung	Taste
D	DUE 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 in der Kurzanweisungsspalte `D` eingegeben.

```
   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 ···············································································
```

Die Zeile `2.00` wurde gelöscht.

10.10 D – Löschen einer Satzmarkierung

Wird die Kurzanweisung `D` mit der Funktionstaste F3 übertragen, löscht sie eine eventuell vorhandene Satzmarkierung (siehe Abschnitt „Satzmarkierungen“ auf Seite 46).

Kurzanweisung	Taste
D	F3

Die Kurzanweisung `D` löscht nur die Satzmarkierungen `1..9`; Sondermarkierungen bleiben erhalten.

10.11 E – Einfügen von Zeichen

Mit der Kurzanweisung E wird die angegebene Zeile für das anschließende Einfügen von Zeichen auf überschreibbar gestellt. Wenn nötig wird Platz geschaffen, um die Zeichen einfügen zu können.

Kurzanweisung	Taste
E	DUE oder F2

Enthält die mit E gekennzeichnete Zeile nicht mindestens 20 NIL-Zeichen am Zeilenende, stellt der EDT 20 NIL-Zeichen am Zeilenende zur Verfügung. Die Zeichen der Zeile (incl. des abschließenden LZE), 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 nach der Datenübertragung ins Datenfenster nachgerückt.

Im EDIT-LONG-Modus bewirkt die Kurzanweisung E, dass zusätzlich zur üblichen Darstellung des Datensatzes eine Zeile mit NIL-Zeichen angeboten wird, sofern eine weitere Zeile im Datenfenster darstellbar ist.

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
   3.00 <·····························································
   4.00 Und leider auch Theologie<····································
   5.00 Durchaus studiert, mit heißem Bemühn.<·····························
e  6.00 Da steh ich nun ich armer! Und bin so klug als wie zuvor;<·····
   7.00 Heiße Magister, heiße Doktor gar<·····························
   8.00 Und ziehe schon an die zehen Jahr<····························
  10.00 ·····························································
```

In Zeile 6.00 wird die Kurzanweisung E zum Einfügen eingegeben.

```
  1.00 Habe nun, ach! Philosophie,<·····································
  2.00 Juristerei und Medizin,<·····································
  3.00 <·····························································
  4.00 Und leider auch Theologie<····································
  5.00 Durchaus studiert, mit heißem Bemühn.<·····························
  6.00 Da steh ich nun ich armer! Und bin so klug als wie z·····················
  7.00 Heiße Magister, heiße Doktor gar<·····································
  8.00 Und ziehe schon an die zehen Jahr<····································
 10.00 ·····························································
```

Da am Ende der Zeile `6.00` weniger als 20 Zeichen zur Verfügung stehen, schiebt der EDT den Zeilenrest (einschließlich des LZE) aus dem Datenfenster und stellt 20 `NIL`-Zeichen zur Verfügung.

```
  1.00 Habe nun, ach! Philosophie,<·····································
  2.00 Juristerei und Medizin,<·····································
  3.00 <·····························································
  4.00 Und leider auch Theologie<····································
  5.00 Durchaus studiert, mit heißem Bemühn.<·····························
  6.00 Da steh ich nun ich armer Tor! Und bin so klug als wie z·················
  7.00 Heiße Magister, heiße Doktor gar<·····································
  8.00 Und ziehe schon an die zehen Jahr<····································
 10.00 ·····························································
```

In Zeile `6.00` wurde das Wort *Tor* eingefügt.

```
  1.00 Habe nun, ach! Philosophie,<·····································
  2.00 Juristerei und Medizin,<·····································
  3.00 <·····························································
  4.00 Und leider auch Theologie<····································
  5.00 Durchaus studiert, mit heißem Bemühn.<·····························
  6.00 Da steh ich nun ich armer Tor! Und bin so klug als wie zuvor;<···········
  7.00 Heiße Magister, heiße Doktor gar<·····································
  8.00 Und ziehe schon an die zehen Jahr<····································
 10.00 ·····························································
```

Da in Zeile `6.00` weniger als 20 Zeichen eingegeben wurden, rückt der EDT den verschobenen Zeilenrest ins Datenfenster nach.

10.12 H – Hexadezimalmodus für einen Satz einschalten

Die Kurzanweisung H schaltet nur für den ausgewählten Satz den Hexadezimalmodus ein (siehe Abschnitt „Hexadezimalmodus“ auf Seite 125) und stellt alle zu diesem Satz gehörenden Bildschirmzeilen auf überschreibbar.

Kurzanweisung	Taste
H	DUE oder F2

Ist das Datenfenster (bei geteiltem Bildschirm) zu klein, um die Datenzeile mit allen Hexzeilen anzuzeigen, wird die Kurzanweisung mit der Meldung EDT2404 abgewiesen.

Wenn erforderlich und möglich, wird das Datenfenster so repositioniert, dass die Anzeige der Hexzeilen möglich wird.

Die Hexzeilen werden beim neuen Bildschirmaufbau nach dem Abschicken nicht mehr angezeigt, es sei denn, es wurde in der Kurzanweisungsspalte erneut die Kurzanweisung H eingegeben. Damit das möglich ist, muss aber vorher mit @PAR EDIT-FULL=ON das Datenfenster und die Kurzanweisungsspalte auf überschreibbar gestellt werden.

Beispiel

Die Daten im Beispiel sind in UTF8 codiert.

```
   1.00 Preisänderung:<·····························································
   2.00 <···········································································
H  3.00       200,00 €<·····················································
```

Nur die Zeile 3.00 soll im Hexadezimalmodus angezeigt werden. Dazu wird in der Kurzanweisungsspalte H eingegeben.

```
   1.00 Preisänderung:<·····························································
   2.00 <···········································································
   3.00       200,00 €<·····················································
        2222223332332E···················································
        000000200C0002···················································
        ·············8···················································
        ·············2···················································
        ·············A···················································
        ·············C···················································
        ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
```

10.13 I – Aktivieren der Dauereinfügefunktion

Mit der Kurzanweisung I wird die Dauereinfügefunktion aktiviert, d.h. es werden am Bildschirm *vor* der angegebenen Zeile neue Zeilen bereitgestellt. Sie können mit Text gefüllt und nach der Datenübertragung in die Arbeitsdatei eingefügt werden. Solange die Dauereinfügefunktion nicht beendet wird (siehe unten) wiederholt sich dieser Vorgang, d.h. nach der Datenübertragung wird im Anschluss an die gerade eingefügten Zeilen erneut ein Bereich von neuen Zeilen bereitgestellt.

Zur Unterscheidung zwischen neuen Zeilen und Leerzeilen (die Sätzen der Länge 0 entsprechen) siehe Abschnitt „F-Modus“ auf Seite 105.

Kurzanweisung	Taste
I	DUE oder F2

Die Dauereinfügefunktion wird beendet, wenn

- in der letzten ausgegebenen neuen Zeile keine Eingabe erfolgt oder
- die Kurzanweisung S eingegeben wird oder
- durch andere Anweisungen der Einfügebereich aus dem Datenfenster herauspositioniert wird.

Sofern das Arbeitsfenster groß genug ist, wird ein 9 Zeilen umfassender Einfügebereich von neuen Zeilen bereitgestellt. Evtl. wird der dargestellte Bereich im Datenfenster verschoben, um die neuen Zeilen bereitstellen zu können.

Die Dauereinfügefunktion wird nur dann aktiviert, wenn im Datenfenster mindestens 10 Bildschirmzeilen sichtbar sind (Spaltenzähler nicht mitgerechnet). Für kürzere Datenfenster wird mit I nur einmal ein Einfügebereich in der Datenfensterlänge – 1 bereitgestellt.

Beim Einfügen der neuen Zeilen vergibt der EDT die Zeilennummern nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Ist ein Einfügen nicht möglich, wird die Kurzanweisung I mit der Meldung EDT5365 zurückgewiesen.

Wird in einer angebotenen neuen Zeile kein Text eingetragen, wird auch kein Satz in der Arbeitsdatei angelegt. Dies beendet aber nicht die Dauereinfügefunktion, sofern es sich dabei nicht um die letzte Zeile handelt.

Beispiel

```
   1.00 a<·····································································
   2.00  b<····································································
   3.00    c<··································································
   4.00     d<·································································
   5.00      e<································································
   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 in Zeile `8.00` die Kurzanweisung `I` (Dauereinfügefunktion) eingegeben.

```
   1.00 a<·····································································
   2.00  b<····································································
   3.00    c<··································································
   4.00     d<·································································
   5.00      e<································································
   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 ·····································································
·····························································0001.00:00001(00)
```

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.

10.14 J – Zusammenketten zweier Sätze

Mit der Kurzanweisung J wird der angegebene Satz an seinen Vorgängersatz angefügt. Der angefügte Satz wird anschließend gelöscht.

Kurzanweisung	Taste
J	DUE oder F2

Übersteigt die Summe der Satzlängen der verketteten Sätze die maximale Satzlänge von 32768 Zeichen, wird auf die maximale Satzlänge abgeschnitten. Der mit J (teilweise) angefügte Satz wird dann nicht gelöscht und es wird die Meldung EDT2400 ausgegeben.

Wird die Kurzanweisung J für den ersten Satz in der Arbeitsdatei gegeben, wird sie ignoriert.

Die inverse Operation, das Auftrennen eines Datensatzes, ist im Abschnitt „Anweisung im Datenfenster - Auftrennen eines Datensatzes“ auf Seite 117 beschrieben (siehe auch die Anweisung @SEP).

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
   3.00 <·····························································
j  4.00 Und leider auch Theologie<·····································
   5.00 Durchaus studiert, mit heißem Bemühn.<·····························
   6.00 Da steh ich nun ich armer Tor!<·····································
j  7.00  Und bin so klug als wie zuvor;<·····································
   8.00 Heiße Magister, heiße Doktor gar<·····································
   9.00 Und ziehe schon an die zehen Jahr<·····································
  10.00 ·····························································
```

Die Sätze in den Zeilen 4.00 und 7.00 soll an ihre jeweiligen Vorgängersätze angefügt werden.

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
   3.00 Und leider auch Theologie<·····································
   5.00 Durchaus studiert, mit heißem Bemühn.<·····························
   6.00 Da steh ich nun ich armer Tor! Und bin so klug als wie zuvor;<·····
   8.00 Heiße Magister, heiße Doktor gar<·····································
   9.00 Und ziehe schon an die zehen Jahr<·····································
  10.00 ·····························································
```

Die Sätze wurden angefügt. Anfügen an einen Satz der Länge 0 ist gleichbedeutend mit Verschieben des angefügten Satzes.

10.15 K – Kopieren einer Zeile in die Anweisungszeile

Die Kurzanweisung K kopiert die angegebene Bildschirmzeile in die Anweisungszeile. Das Kopieren beginnt an der eingestellten Spaltenposition. Von dort an werden maximal so viele Zeichen kopiert, wie in der Anweisungszeile Platz haben. Das letzte Zeichen der Anweisungszeile wird auf binär Null gesetzt. Ein etwa vorhandener alter Inhalt der Anweisungszeile wird vorher gelöscht.

Kurzanweisung	Taste
K	DUE oder F2

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

Die Anweisungszeile wird in dem Zeichensatz interpretiert, der für die Kommunikation mit der Datensichtstation eingestellt ist (siehe Anweisung @CODENAME name,TERMINAL). Die Zeile muss daher evtl. vom Zeichensatz der Arbeitsdatei in diesen Ziel-Zeichensatz umcodiert werden. Sind einzelne Zeichen dabei nicht konvertierbar, wird die Meldung EDT5453 ausgegeben, die Anweisung aber ausgegeben, wobei die nicht konvertierbaren Zeichen durch Fragezeichen '?' ersetzt werden.

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
@SHIH·····························································0001.00:00001(00)
```

Mit der Anweisung @SHIH wird der Anweisungspuffer des EDT in Arbeitsdatei 9 ausgegeben.

```
   1.00 @par lower=on<·····························································
   2.00 @split 12(7)<·····························································
k  3.00 @on & find 'Tor'<·····························································
   4.00 @scale on<·····························································
·····························································0001.00:00001(09)
```

Mit der Kurzanweisung K wird Bildschirmzeile 3.00 in die Anweisungszeile kopiert.

```
   1.00 @par lower=on<·····················································
   2.00 @split 12(7)<·····················································
   3.00 @on & find 'Tor'<·····················································
   4.00 @scale on<·····················································
                                                                        
                                                                        
                                                                        
                                                                        
0; @on & find 'Tor'·····································0001.00:00001(09)
```

Vor der kopierten Anweisung wird 0; eingefügt. Dadurch wird in die Arbeitsdatei 0 gewechselt und die Suche gestartet.

10.16 L – Umsetzen von Zeilen in Kleinbuchstaben

Die Kurzanweisung L bewirkt, dass der angegebene Satz in Kleinbuchstaben umgesetzt wird. Die Umsetzung wird analog zu @CONVERT TO=LOWER mit der entsprechenden XHCS-Funktion durchgeführt.

Kurzanweisung	Taste
L	DUE oder F2

Beispiel

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

Zeile 5.00 soll in Kleinbuchstaben umgewandelt werden.

```
   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<·····················
```

Zeile 5.00 wurde in Kleinbuchstaben umgewandelt.

10.17 M – Aufsammeln von Zeilen zum Verschieben

Die Kurzanweisung M überträgt Zeilen- und Arbeitsdateinummer des angegebenen Satzes in den Kopierpuffer des EDT, um den Satz später mit einer der Kurzanweisungen A, B oder O zu verschieben.

Kurzanweisung	Taste
M	DUE oder F2

Sobald eine der Kurzanweisungen A, B oder O eingegeben wurde, wird die Verschiebung ausgeführt, d.h. die Sätze werden an der entsprechenden Stelle eingefügt und an ihrer ursprünglichen Position gelöscht. Anschließend wird der Inhalt des Kopierpuffers gelöscht.

Enthält bei Angabe von M der Kopierpuffer schon Einträge, die durch die Kurzanweisungen C oder R erzeugt wurden, so wird der Kopierpuffer gelöscht, bevor die angegebene Zeile eingetragen wird; die Meldung EDT0295 wird ausgegeben.

Der Kopierpuffer kann auch durch Aufsammeln von Zeilen aus verschiedenen Arbeitsdateien aufgebaut werden. Die durch den Kopierpuffer bestimmten Sätze können dann in jede Arbeitsdatei verschoben werden.

Der Kopierpuffer enthält Arbeitsdatei- und Zeilennummer der mit M aufgesammelten Sätze. Die Zeilennummern und Inhalt der zu kopierenden Sätze sollten daher zwischen dem Aufsammeln mit M und der Ausführung des Kopiervorgangs durch Eingabe der Kurzanweisungen A, B oder O nicht verändert werden. Geschieht dies doch, werden die neuen Inhalte verschoben. Wurden zwischenzeitlich Sätze gelöscht, werden diese ohne Warnung beim Verschieben übergangen.

Da der Kopierpuffer nur Arbeitsdatei- und Zeilennummer enthält, findet zum Zeitpunkt des Aufsammelns auch noch keine Umcodierung der ausgewählten Zeilen statt. Da der Ziel-Zeichensatz erst durch die Festlegung der Ziel-Arbeitsdatei bestimmt wird, findet eine eventuell nötige Umcodierung erst bei Auswertung der Kurzanweisungen A, B bzw. O statt (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Hinweis

Bei einem geteilten Bildschirm können in einem Dialogschritt Zeilen vom ersten Arbeitsfenster ins zweite Arbeitsfenster verschoben werden. Beim Verschieben vom zweiten Arbeitsfenster ins erste Arbeitsfenster sind, bedingt durch die Abarbeitungsreihenfolge, zwei Dialogschritte notwendig.

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
m  3.00 <·····························································
   4.00 Und leider auch Theologie<····································
a  5.00 Durchaus studiert, mit heißem Bemühn.<·····························
   6.00 Da steh ich nun ich armer Tor!<·····································
   7.00 Und bin so klug als wie zuvor;<·····································
   8.00 Heiße Magister, heiße Doktor gar<····································
   9.00 Und ziehe schon an die zehen Jahr<····································
  10.00 ·····························································
```

Die Zeile `3.00` soll hinter die Zeile `5.00` verschoben werden. Dazu wurde in Zeile `3.00` die Kurzanweisung `M` und in Zeile `5.00` die Kurzanweisung `A` eingegeben.

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
   4.00 Und leider auch Theologie<····································
   5.00 Durchaus studiert, mit heißem Bemühn.<·····························
   5.10 <·····························································
   6.00 Da steh ich nun ich armer Tor!<·····································
   7.00 Und bin so klug als wie zuvor;<·····································
   8.00 Heiße Magister, heiße Doktor gar<····································
   9.00 Und ziehe schon an die zehen Jahr<····································
  10.00 ·····························································
```

Die Zeile `3.00` wurde hinter die Zeile `5.00` verschoben.

10.18 O – Kopieren oder Verschieben über einen Zeilenbereich

Die Kurzanweisung O kopiert oder verschiebt Datensätze, deren Arbeitsdatei- und Zeilennummern mit C, M oder R im Kopierpuffer aufgesammelt wurden, *über* (on) die angegebene Zeile und den nachfolgenden Zeilenbereich.

Kurzanweisung	Taste
O	DUE oder F2

Im Folgenden wird der Einfachheit halber immer von Kopieren gesprochen, auch wenn die Zeilen nach dem Kopieren gelöscht, d.h. verschoben werden.

Ist der Kopierpuffer leer, wird die Anweisung mit der Meldung EDT5376 abgewiesen.

Es sind zwei Fälle zu unterscheiden:

1. Wenn der angegebene Satz ab Spalte 1 im Datenfenster angezeigt wird, wird der gesamte Inhalt des angegebenen Satzes mit dem zu kopierenden Satz überschrieben.
2. Wenn der angegebene Satz ab einer Spalteposition größer als 1 im Datenfenster angezeigt wird, wird der Bereich ab der angezeigten Spaltenposition in der Länge des zu kopierenden Satzes mit dessen Inhalt überschrieben. Ist die Länge des zu kopierenden Satzes kleiner als die Länge des bei O angegebenen Satzes, bleiben die restlichen Teile des zu überschreibenden Satzes erhalten. Auf diese Weise können Sätze in andere Sätze eingefügt oder an andere Sätze angefügt werden.

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

Beim Anfügen der kopierten Zeilen nach dem Ende der Arbeitsdatei vergibt der EDT die Zeilennummern nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Ist ein Anfügen nicht möglich, unterbleibt der Vorgang, der Kopierpuffer wird nicht gelöscht und die Meldung EDT5365 wird ausgegeben. Bereits überschriebene Zeilen bleiben aber überschrieben.

Ist die aktuelle Arbeitsdatei leer und hat sie den Zeichensatz *NONE, dann erhält sie beim Kopieren den Zeichensatz der Quell-Arbeitsdatei der ersten zu kopierenden Zeile.

Hat die aktuelle Arbeitsdatei einen Zeichensatz, dann werden die zu kopierenden Zeilen in den Zeichensatz der aktuellen Arbeitsdatei konvertiert. Werden dabei Zeichen gefunden, die im Zeichensatz der Arbeitsdatei nicht darstellbar sind, werden diese durch ein Ersatzzeichen ersetzt, sofern ein solches spezifiziert ist (siehe @PAR SUBSTITUTION-CHARACTER). Andernfalls wird die Kurzanweisung abgewiesen und die Fehlermeldung EDT5453 ausgegeben.

Hinweis

Die Kurzanweisung `O` wird erst ausgeführt, nachdem `C`-, `M`- und `R`-Anweisungen abgearbeitet wurden. Deshalb kann in einem Arbeitsfenster in einem Dialogschritt der Zielort auch vor den zu kopierenden Zeilen angegeben werden.

Wenn mit `O` über eine Zeile kopiert wird, die selbst im Bereich der zu kopierenden oder zu verschiebenden Zeilen liegt, kann es zu unerwarteten Effekten kommen, insbesondere beim Verschieben von Zeilen mit `M`. Siehe dazu die folgenden Beispiele.

Beispiel 1

```
   1.00 1<·····························································
c  2.00  2<·····························································
o  3.00   3<·····························································
```

Die Zeile `2.00` soll in die Zeile `3.00` kopiert werden. Dazu wird in Zeile `2.00` die Kurzanweisung `C` und in Zeile `3.00` die Kurzanweisung `O` eingegeben.

```
   1.00 1<·····························································
   2.00  2<·····························································
   3.00  2<·····························································
```

Die Zeile `3.00` wurde mit dem Inhalt der Zeile `2.00` überschrieben.

Beispiel 2

```
   1.00 1<·····························································
c  2.00  2<·····························································
c  3.00   3<·····························································
c  4.00    4<·····························································
c  5.00     5<·····························································
   6.00      6<·····························································
```

Die Zeilen `2.00` bis `5.00` werden zum Kopieren aufgesammelt.

```
   1.00 1<·····························································
   2.00  2<·····························································
   3.00   3<·····························································
o  4.00    4<·····························································
   5.00     5<·····························································
   6.00      6<·····························································
```

Die Zeilen aus dem Kopierpuffer sollen über Zeile `4.00` und folgende kopiert werden.

```
   1.00 1<·····························································
   2.00  2<·····························································
   3.00   3<·····························································
   4.00  2<·····························································
   5.00   3<·····························································
   6.00  2<·····························································
   7.00   3<·····························································
```

Da durch das Kopieren der Inhalt der Zeilen aus dem Kopierpuffer selbst verändert wurde, kommt es zu dem obigen Ergebnis.

Beispiel 3

```
   1.00 1<.....................................................................
m  2.00  2<.....................................................................
   3.00   3<.....................................................................
```

Die Zeile `2.00` wird zum Verschieben ausgewählt.

```
   1.00 1<.....................................................................
o  2.00  2<.....................................................................
   3.00   3<.....................................................................
```

Sie soll Zeile `2.00` überschreiben.

```
   1.00 1<.....................................................................
   3.00   3<.....................................................................
```

Zeile `2.00` ist gelöscht worden, da beim Verschieben die zu verschiebende Zeile nach dem Übertragen gelöscht wird.

Beispiel 4

```
c  1.00 -111-222<·····························································
c  2.00 -333-444<·····························································
c  3.00 -555-666<·····························································
   4.00 ·····························································


>8·····························································0001.00:00001(00)
```

Die Zeilen `1.00` bis `3.00` werden zum Kopieren ausgewählt. Gleichzeitig wird das Datenfenster mit der Anweisung `>8` um 8 Zeichen nach rechts verschoben.

```
o  1.00 <·····························································
   2.00 <·····························································
   3.00 <·····························································
   4.00 <·····························································



<<·····························································0001.00:00009(00)
```

Die Zeilen aus dem Kopierpuffer werden mit `0` über die nach rechts verschobene Zeile 1 und die Folgezeile kopiert. Gleichzeitig wird das Datenfenster mit der Anweisung << wieder auf den Datensatzanfang geschoben.

```
   1.00 -111-222-111-222<·····················································
   2.00 -333-444-333-444<·····················································
   3.00 -555-666-555-666<·····················································
   4.00 ·····························································



·····························································0001.00:00001(00)
```

Als Ergebnis wurde jede Zeile an sich selbst angefügt.

10.19 R – Aufsammeln von Zeilen zum mehrfachen Kopieren

Die Kurzanweisung R überträgt Zeilen- und Arbeitsdateinummer des angegebenen Satzes in den Kopierpuffer des EDT, um den Satz später mit einer der Kurzanweisungen A, B oder O mehrfach zu kopieren.

Kurzanweisung	Taste
R	DUE oder F2

Sobald eine der Kurzanweisungen A, B oder O eingegeben wurde, wird der Kopiervorgang ausgeführt, d.h. die Sätze werden an der entsprechenden Stelle eingefügt. Im Gegensatz zu den Kurzanweisungen C und M wird bei R der Inhalt des Kopierpuffers anschließend nicht gelöscht, sondern steht für weitere Kopiervorgänge zur Verfügung.

Enthält bei Angabe von R der Kopierpuffer schon Einträge, die durch die Kurzanweisungen C oder M erzeugt wurden, so wird der Kopierpuffer gelöscht, bevor die angegebene Zeile eingetragen wird; die Meldung EDT0295 wird ausgegeben.

Der Kopierpuffer kann auch durch Aufsammeln von Zeilen aus verschiedenen Arbeitsdateien aufgebaut werden. Die durch den Kopierpuffer bestimmten Sätze können dann in jede Arbeitsdatei kopiert werden.

Der Kopierpuffer enthält Arbeitsdatei- und Zeilennummer der mit R aufgesammelten Sätze. Die Zeilennummern und Inhalt der zu kopierenden Sätze sollten daher zwischen dem Aufsammeln mit R und der Ausführung des Kopiervorgangs durch Eingabe der Kurzanweisungen A, B oder O nicht verändert werden. Geschieht dies doch, werden die neuen Inhalte kopiert. Wurden zwischenzeitlich Sätze gelöscht, werden diese ohne Warnung beim Kopieren übergangen.

Da der Kopierpuffer nur Arbeitsdatei- und Zeilennummer enthält, findet zum Zeitpunkt des Aufsammelns auch noch keine Umcodierung der ausgewählten Zeilen statt. Da der Ziel-Zeichensatz erst durch die Festlegung der Ziel-Arbeitsdatei bestimmt wird, findet eine eventuell nötige Umcodierung erst bei Auswertung der Kurzanweisungen A, B bzw. O statt (siehe Abschnitt „Zeichensätze“ auf Seite 48).

Hinweis

Bei einem geteilten 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.

Beispiel

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
r  3.00 <·····································
   4.00 Und leider auch Theologie<·····································
   5.00 Durchaus studiert, mit heißem Bemühn.<·····································
   6.00 Da steh ich nun ich armer Tor!<·····································
b  7.00 Und bin so klug als wie zuvor;<·····································
a  8.00 Heiße Magister, heiße Doktor gar<·····································
   9.00 Und ziehe schon an die zehen Jahr<·····································
  10.00 ·····································
```

Die Zeile `3.00` soll vor die Zeile `7.00` und hinter die Zeile `8.00` kopiert werden. Dazu wird in Zeile `3.00` die Kurzanweisung `R`, in Zeile `7.00` die Kurzanweisung `B` und in Zeile `8.00` die Kurzanweisung `A` eingegeben.

```
   1.00 Habe nun, ach! Philosophie,<·····································
   2.00 Juristerei und Medizin,<·····································
   3.00 <·····································
   4.00 Und leider auch Theologie·····································
   5.00 Durchaus studiert, mit heißem Bemühn.<·····································
   6.00 Da steh ich nun ich armer Tor!<·····································
   6.10 <·····································
   7.00 Und bin so klug als wie zuvor;<·····································
   8.00 Heiße Magister, heiße Doktor gar<·····································
   8.10 <·····································
   9.00 Und ziehe schon an die zehen Jahr<·····································
  10.00 ·····································



  21.00 ·····································
  22.00 ·····································
·····0001.00:00001(00)
```

Die Zeile `3.00` wurde vor die Zeile `7.00` und hinter die Zeile `8.00` kopiert.

10.20 S – Positionieren des Arbeitsfensters (horizontal und vertikal)

Mit S wird das Arbeitsfenster in zwei Schritten zu der gewünschten Zeilen- und Spaltenposition verschoben.

Im *ersten* Schritt, d.h. nach Übertragung der Kurzanweisung S mit DUE (F2 wirkt bei S wie DUE), wird die angegebene Zeile auf überschreibbar gestellt und in die zweite Zeile des Arbeitsfensters positioniert. In der ersten Zeile des Arbeitsfensters wird ein Spaltenzähler ausgegeben.

Im *zweiten* Schritt wird die Zeile unterhalb des Spaltenzählers benutzt, um zur gewünschten Spalte zu positionieren. Dazu sind bis vor die gewünschte Spaltenposition Leerzeichen einzugeben und die Eingabe zu übertragen. Dann positioniert der EDT

– die vorher bei S angegebene Zeile in die erste Zeile des Arbeitsfensters und

– das Arbeitsfenster auf die erste Spalte ungleich Leerzeichen in dieser Zeile.

Kurzanweisung	Taste
S	DUE oder F2

Die Kurzanweisung S wird ignoriert, wenn sie in einer nicht (mehr) vorhandenen Zeile eingegeben wird. Die Kurzanweisung S ist im EDIT-LONG-Modus nicht zulässig.

Wird die Zeile im zweiten Schritt 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.

Überschreibt der Benutzer den gesamten Text einer mit S gekennzeichneten Zeile mit Leerzeichen, dann positioniert der EDT das Arbeitsfenster auf die erste nicht mehr am Bildschirm sichtbare Spalte.

Im zweiten Schritt der Bearbeitung (Positionieren auf die gewünschte Spalte) können zusätzlich beliebige weitere (miteinander kombinierbare) Kurzanweisungen eingegeben werden. Dabei wird die Positionierung vor der Bearbeitung der anderen Kurzanweisungen ausgeführt. Das hat insbesondere Auswirkungen auf die Kurzanweisung O, die nach den Regeln ausgeführt wird, die für eine nach rechts verschobene Spaltenposition gelten.

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` und die Spalte mit der Warenbezeichnung positioniert werden. Im ersten Schritt wird daher `S` in Zeile `4.00` angegeben.

```
        ----+----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 ·····················································································
```

Zeile `4.00` wurde überschreibbar gestellt. In dieser Zeile werden bis zur gewünschten Spaltenposition Leerzeichen eingegeben.

```
   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 ·····················································································
```

Nach Übertragung hat der EDT das Arbeitsfenster auf Zeile `4.00` und Spalte 18 positioniert.

Hinweis

Das Positionieren auf Zeile und Spalte ist mit der Anweisung @SETF auch in einem Dialogschritt möglich. Allerdings muss dann die Spaltenposition als Zahl eingegeben werden.

10.21 T – Syntaxtest durch SDF

Die Kurzanweisung T übergibt die angegebene Zeile evtl. gemeinsam mit Fortsetzungszeilen (im Sinne von SDF) zur Kontrolle der Kommando- oder Anweisungssyntax an die Systemkomponente SDF.

Ist die SDF-Option GUIDANCE=MIN|MED|MAX eingestellt, 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, in dem die Anweisung übernommen wird, wird durch die SDF-Option 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 Satzinhalten:

1. Sätze, 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 ist durch den aktuellen Benutzer und die aktuelle Umgebung bestimmt.

2. Sätze, die mit '//' beginnen:

   Diese werden an SDF zur Anweisungsüberprüfung übergeben. Der externe oder interne Programmname muss durch die Anweisung @PAR SDF-PROGRAM voreingestellt sein, andernfalls wird die Kurzanweisung mit der Meldung EDT5320 zurückgewiesen. Der Programmname muss zudem in einer aktuellen SDF-Syntaxdatei bekannt sein. Andernfalls wird die Kurzanweisung mit der Meldung EDT5321 zurückgewiesen.

3. sonstige Datenzeilen:

   Die Angabe von T wird für sonstige Datenzeilen ignoriert.

Zeilen, die mit '/' oder '//' beginnen und als letztes Zeichen ein Fortsetzungszeichen ('-') enthalten, werden mit der Folgezeile, sofern diese ebenfalls mit '/' bzw. '//' beginnt, verkettet und bei der Bearbeitung der Kurzanweisung T gemeinsam an SDF übergeben. In Fortsetzungszeilen müssen keine T-Kurzanweisungen angegeben werden; die Angabe in der ersten Zeile genügt.

Das geprüfte Kommando bzw. die Anweisung überschreibt das alte Kommando bzw. die Anweisung inklusive aller Folgezeilen in der Arbeitsdatei. Wenn das Kommando bzw. die Anweisung bei der Prüfung durch SDF verändert wurde (z.B. weil in einem vorangegangenen /MODIFY-SDF-OPTIONS-Kommando LOGGING=INVARIANT eingestellt wurde), werden die betroffenen Zeilen neu formatiert und evtl. in mehrere Fortsetzungszeilen aufgeteilt. Das Fortsetzungszeichen wird in der 72. Spalte gesetzt.

Wenn nötig, werden die nachfolgenden Zeilen umnummeriert. Die Vergabe der Zeilennummern erfolgt dabei nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37). Können die von SDF erzeugten Zeilen nicht eingefügt werden, wird die Bearbeitung der Kurzanweisung mit der Meldung EDT5365 abgebrochen.

Enthält ein geprüftes Kommando bzw. eine Anweisung Fehler und wird kein erfolgreicher Korrekturdialog geführt, so gibt der EDT die Meldung EDT4310 aus und bricht die Überprüfung ab, d.h. nachfolgende markierte Kommandos und Anweisungen werden nicht mehr geprüft.

Hat die Arbeitsdatei mit den zu prüfenden Zeilen einen anderen Zeichensatz als EDF03IRV muss man Besonderheiten beim Umgang mit SDF beachten, insbesondere sind Zeichen die nicht zum EBCDIC-Kern gehören natürlich nur in Literalen oder Kommentaren zulässig. Darüber hinaus führt SDF einen Korrekturdialog immer in dem mit /MODIFY-TERMINAL-OPTIONS eingestellten Zeichensatz und interpretiert die ihm übergebenen Byte-Sequenzen auch immer in diesem Zeichensatz.

Daher konvertiert der EDT die Anweisungen oder Kommandos vor Übergabe an SDF in den mit /MODIFY-TERMINAL-OPTIONS eingestellten Zeichensatz, falls der aktuell eingestellte GUIDANCE-MODE einen Korrekturdialog ermöglicht. Gelingt das nicht, wird die T-Kurzanweisung mit der Meldung EDT5327 abgebrochen.

Ist kein Korrekturdialog möglich, konvertiert der EDT nach anderen (großzügigeren) Regeln. Hat die Arbeitsdatei einen EBCDIC-Zeichensatz, wird dieser ohne Konvertierung benutzt. Hat die Arbeitsdatei einen ISO-Zeichensatz, wird der entsprechende EBCDIC-Referenzzeichensatz verwendet. In allen anderen Fällen verwendet der EDT den Zeichensatz UTFE. Ist die Konvertierung nicht möglich, wird die T-Kurzanweisung mit der Meldung EDT5453 abgebrochen.

Wenn SDF Daten zurückliefert, werden diese wieder in den Zeichensatz der Arbeitsdatei umgewandelt. Ist das nicht möglich, wird die T-Kurzanweisung mit der Meldung EDT5453 abgebrochen.

Hinweis

Bei GUIDANCE=EXPERT werden von SDF gemeldete Fehler von EDT nur in Form der Meldung EDT4310 angezeigt. Zur genaueren Fehleranalyse empfiehlt es sich, GUIDANCE=MIN, MED oder MAX einzustellen.

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.

Ist mit /MODIFY-TERMINAL-OPTIONS der Zeichensatz UTFE eingestellt, ist das Bildschirmlayout beim Korrekturdialog von SDF verschoben, wenn Zeichen vorkommen, die außerhalb von EDF03IRV liegen. Das liegt daran, dass SDF derzeit Unicode nicht unterstützt, stellt aber normalerweise keine funktionelle Einschränkung dar.

Ein Kommando oder eine Anweisung darf sowohl bei der Ein- als auch bei der Ausgabe eine maximale Länge von 16379 Byte nicht übersteigen. Andernfalls wird die Kurzanweisung T mit der Meldung EDT5325 bzw. EDT5326 abgebrochen.

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, -<·····························
   3.00 /                       NEW-NAME=FILE3, -<·····························
   4.00 /                       PROT=*PARAMETERS(UCCESS=*READ)<·····················
t  5.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE1, -<·····························
   6.00 /                       NEW-NAME=FILE2, -<·····························
   7.00 /                       PROT=*PARAMETERS(ACCESS=*READ)<·····················
   8.00 ···········································································
```

Die Zeilen 1.00 bis 8.00 sollen durch SDF auf richtige SDF-Syntax geprüft werden.

```
   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 0002.0000
···········································0002.00:00001(00)
```

Es wird auf die erste Zeile des fehlerhaften Kommandos positioniert und die Zeilen des Kommandos werden überschreibbar 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, -<·····························
   3.00 /                       NEW-NAME=FILE3, -<·····························
   4.00 /                       PROT=*PARAMETERS(UCCESS=*READ)<·····················
t  5.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE1, -<·····························
   6.00 /                       NEW-NAME=FILE2, -<·····························
   7.00 /                       PROT=*PARAMETERS(ACCESS=*READ)<·····················
   8.00 ·····························································
```

Die Zeilen `1.00` bis `8.00` sollen durch SDF auf richtige SDF-Syntax geprüft werden.

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


------------------------------------------------------------------------------
FILE-NAME             = FILE2
NEW-NAME              = FILE3
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             = FILE2
NEW-NAME              = FILE3
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 Benutzer korrigiert den Fehler.

```
   1.00 /SET-JOB-STEP<·····························································
   2.00 /MODIFY-FILE-ATTRIBUTES FILE-NAME=FILE2,NEW-NAME=FILE3,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 `2.00` bis `4.00` werden durch die Zeilen `2.00` bis `3.00` ersetzt und neu formatiert.

10.22 U – Umsetzen von Zeilen in Großbuchstaben

Die Kurzanweisung U bewirkt, dass der angegebene Satz in Großbuchstaben umgesetzt wird. Die Umsetzung wird analog zu @CONVERT TO=UPPER mit der entsprechenden XHCS-Funktion durchgeführt.

Kurzanweisung	Taste
U	DUE oder F2

Beispiel

```
   1.00 aaa bbb ccc ddd<·····················································
   2.00 eee fff ggg hhh<·····················································
   4.00 <····················································································
u  5.00 iii jjj kkk lll<·····················································
```

Zeile 5.00 soll in Großbuchstaben umgewandelt werden.

```
   1.00 aaa bbb ccc ddd<·····················································
   2.00 eee fff ggg hhh<·····················································
   4.00 <····················································································
   5.00 III JJJ KKK LLL<·····················································
```

Zeile 5.00 wurde in Großbuchstaben umgewandelt.

10.23 X – Ändern von Zeilen

Mit der Kurzanweisung X werden Zeilen zur Änderung auf überschreibbar gestellt.

Kurzanweisung	Taste
X	DUE oder F2

Änderungen sind in dem im Datenfenster dargestellten Ausschnitt der Sätze wirksam. Die Bereiche links und rechts vom dargestellten Ausschnitt bleiben unverändert. Beim Einfügen mit EFG ist jedoch zu berücksichtigen, dass Teile der Sätze, die über das rechte Ende des Datenfensters hinausgeschoben werden, verloren gehen. Dies kann vermieden werden, wenn die Kurzanweisung E zum Einfügen verwendet wird.

Das gesamte Datenfenster kann durch F2 auf überschreibbar gestellt werden (siehe Abschnitt „Die F-Tasten“ auf Seite 128). Mit @PAR EDIT-FULL=ON kann der EDT veranlasst werden, alle Zeilen dauernd überschreibbar zu stellen.

Hinweis

Die Satzmarkierungen mit Sonderfunktion 14 und 15 (siehe Abschnitt „Satzmarkierungen“ auf Seite 46) werden beim Ändern eines Satzes gelöscht. Ein Satz mit Satzmarkierung 15 kann nur dann geändert (überschrieben) werden, wenn der durch Markierung 15 gesetzte Schreibschutz mit @PAR PROTECTION=OFF ignoriert wird. Das Konvertieren in Groß- oder Kleinbuchstaben mit den Kurzanweisungen U bzw. L oder das Aneinanderfügen von Zeilen mit der Kurzanweisung J gilt in diesem Sinne nicht als Änderung. Die Sondermarkierungen bleiben dabei erhalten.

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 2.00 wird zum Ändern ausgewählt.

```
   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 überschreibbar gesetzt. Vor DISNEYLAND wird die Postleitzahl eingefügt.

10.24 1..9 – Einfügen von Zeilen

Die Kurzanweisungen `1..9` dienen dazu, am Bildschirm vor der angegebenen Zeile neue Zeilen bereitzustellen. Sie können mit Text gefüllt und nach der Datenübertragung in die Arbeitsdatei eingefügt werden. Zur Unterscheidung zwischen neuen Zeilen und Leerzeilen (die Sätzen der Länge `0` entsprechen) siehe den Abschnitt „F-Modus“ auf Seite 105.

Kurzanweisung	Taste
1..9	DUE oder F2

Abhängig von der gewählten Kurzanweisung (`1..9`) werden vor der angegebenen Zeile 1 bis 9 neue Zeilen angeboten. Evtl. wird der dargestellte Bereich im Datenfenster verschoben, um die neuen Zeilen bereitstellen zu können.
Für diese neuen Zeilen werden bereits Zeilennummern nach dem Verfahren „Einfügen zwischen zwei Zeilen“ (siehe Abschnitt „Zeilennummernvergabe“ auf Seite 37) gebildet. Ist ein Einfügen nicht möglich, wird die Kurzanweisung mit der Meldung `EDT5365` abgewiesen. Wird in einer angebotenen neuen Zeile kein Text eingetragen, wird auch kein Satz in der Arbeitsdatei angelegt.

Beispiel

```
   1.00 1··9    EINFUEGEN VON ZEILEN<·····································
   2.00 <·····································································
   3.00 <·····································································
   4.00 Mit 1··9 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 in Zeile `7.00` die Kurzanweisung 3 eingegeben.

```
   1.00 1··9    EINFUEGEN VON ZEILEN<·····································
   2.00 <·····································································
   3.00 <·····································································
   4.00 Mit 1··9 koennen Saetze in eine Arbeitsdatei eingefuegt werden.<·····
   5.00 <·····································································
   6.00 111<··································································
   6.10 ·····································································
   6.20 ·····································································
   6.30 ·····································································
   7.00    222<·····························································
   8.00       333<··························································
   9.00 ·····································································
```

Die neuen Zeilen `6.10` bis `6.30` werden bereitgestellt.

10.25 1..9 – Setzen einer Satzmarkierung

Wird eine der Kurzanweisungen `1..9` mit der Funktionstaste F3 übertragen, wird im angegebenen Satz die entsprechende Satzmarkierung gesetzt (siehe Abschnitt „Satzmarkierungen“ auf Seite 46). Diese Satzmarkierungen können z.B. zum Positionieren des Datenfensters oder zum Kopieren der markierten Sätze (siehe @ON-Anweisung) benutzt werden.

Kurzanweisung	Taste
1..9	F3

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

11 Kompatibilitäts-Modus

Dieser Abschnitt beschreibt den Kompatibilitäts-Modus und sein Zusammenspiel mit dem Unicode-Modus.

Der Kompatibilitäts-Modus bietet die volle Funktionalität des EDT V16.6B inklusive der alten L-Modus-Unterprogramm-Schnittstelle. Die erweiterten Funktionen des Unicode-Modus sind im Kompatibilitäts-Modus nicht verfügbar.

Darüber hinaus wird die Funktionalität des EDT V16.6B im Kompatibilitäts-Modus in den folgenden drei geänderten bzw. neuen Anweisungen erweitert.

11.1 @CODENAME – Zeichensatz einstellen

Mit der Anweisung @CODENAME wird im Kompatibilitäts-Modus der global verwendete Zeichensatz festgelegt.

Operation	Operanden	F-Modus, L-Modus
@CODENAME	[name] [,**F**ORCE=⟨**Y**ES / **N**O⟩]	

name Name des einzustellenden Zeichensatzes. Er muss in XHCS bekannt sein.

Handelt es sich um einen 8-Bit-Zeichensatz oder den 7-Bit-Zeichensatz `EDF03IRV`, wird dieser Zeichensatz eingestellt. Im Dialogbetrieb wird zusätzlich verlangt, dass der Zeichensatz von der verwendeten Datensichtstation unterstützt wird. Andernfalls wird die Anweisung mit der Meldung `EDT5259` abgewiesen.

Handelt es sich um einen ISO-Zeichensatz, einen Unicode-Zeichensatz oder um einen anderen 7-Bit-Zeichensatz als `EDF03IRV`, und sind alle Arbeitsdateien leer und keine Dateien geöffnet, dann wird wie bei der @MODE-Anweisung automatisch in den Unicode-Modus gewechselt und der Zeichensatz wird dort für alle Arbeitsdateien eingestellt.

Beim Umschalten in den Unicode-Modus mit @CODENAME wird zudem der Mechanismus der automatischen Wahl des Kommunikations-Zeichensatzes aktiviert (siehe Abschnitt „Kommunikationszeichensatz“ auf Seite 54).

Falls der Zeichensatz angegeben wird, der schon eingestellt ist, wird die Anweisung ohne Meldung ignoriert.

Wird `name` nicht angegeben, wird – falls zulässig – auf den Default-Zeichensatz des EDT umgeschaltet (siehe Abschnitt „Zeichensätze“ auf Seite 48).

FORCE=NO Bei Angabe von `FORCE=NO` bzw. wenn der Operand `FORCE` nicht angegeben wird, ist das Einstellen des Zeichensatzes nur dann zulässig, wenn alle Arbeitsdateien leer und keine Dateien real geöffnet sind.

FORCE=YES Die Angabe von `FORCE=YES` bewirkt, dass der Zeichensatz auch dann umgeschaltet wird, wenn eine oder mehrere Arbeitsdateien nicht leer oder Dateien geöffnet sind.

Die Daten werden nicht umcodiert sondern nur im neuen Zeichensatz interpretiert. Entsprechend ändert sich die Darstellung der Zeichen an der Datensichtstation. Der neue Zeichensatz wird auch im Katalog der Datei bzw. des Bibliothekselements eingetragen, wenn die Arbeitsdatei zurück geschrieben bzw. eine geöffnete Datei geschlossen wird.

Die Angabe von `FORCE=YES` hat keine Wirkung für ISO-Zeichensätze, Unicode-Zeichensätze oder andere 7-Bit-Zeichensätze als `EDF03IRV`.

11.2 @IF (Format 5) – Abfrage von Einstellungen des EDT

Mit diesem Format der @IF-Anweisung kann man in EDT-Prozeduren oder im L-Modus den aktuell eingestellten Betriebsmodus abfragen (siehe Abschnitt „Einführung zu den Betriebsmodi des EDT“ auf Seite 21). Abhängig vom Ergebnis wird eine angegebene Zeichenfolge als Eingabe abgearbeitet oder nicht.

Operation	Operanden	L-Modus
@IF	**OP**ERATING-MODE= { **U**NICODE / **C**OMPATIB- } : [text]	

OPERATING-MODE=

Der Betriebsmodus des EDT wird überprüft.

UNICODE Die Bedingung ist erfüllt, wenn sich der EDT im Unicode-Modus befindet.

COMPATIBLE

Die Bedingung ist erfüllt, wenn sich der EDT im Kompatibilitäts-Modus befindet.

text EDT-Anweisung oder Datenzeile. Falls die Bedingung erfüllt ist, wird die Zeichenfolge so behandelt, als ob man sie im L-Modus an der Eingabeaufforderung eingegeben hätte. Insbesondere die Interpretation als Dateneingabe bzw. als Anweisung erfolgt nach den gleichen Regeln (siehe dazu den Abschnitt „L-Modus“ auf Seite 131).

Der Operand `text` beginnt unmittelbar hinter dem Zeichen `':'`, d.h. evtl. angegebene Leerzeichen gehören schon dazu und werden bei Dateneingaben auch in die Zeile übernommen.

Wird `text` nicht angegeben, wird die Anweisung ignoriert.

11.3 @MODE – Betriebsmodus umschalten

Mit der Anweisung @MODE kann zwischen den Betriebsmodi (Kompatibilitäts-Modus und Unicode-Modus) umgeschaltet werden.

Operation	Operanden	F-Modus, L-Modus
@MODE	**OP**ERATING-MODE= { **U**NICODE / **C**OMPATIB- }	

OPERATING-MODE=

Der Betriebsmodus des EDT wird umgeschaltet.

UNICODE Es wird vom Kompatibilitäts-Modus in den Unicode-Modus umgeschaltet. Wenn sich der EDT bereits im Unicode-Modus befindet, wird die Anweisung ignoriert.

COMPATIBLE

Es wird vom Unicode-Modus in den Kompatibilitäts-Modus umgeschaltet. Wenn sich der EDT bereits im Kompatibilitäts-Modus befindet, wird die Anweisung ignoriert.

Der Betriebsmodus kann nur dann umgeschaltet werden, wenn alle Arbeitsdateien des EDT leer und keine Dateien geöffnet sind. Andernfalls wird die Anweisung mit der Meldung `EDT4983` abgewiesen.

Der Wechsel zwischen den Betriebsmodi entspricht einem Beenden des EDT im einen Modus mit anschließendem Neustart des EDT im anderen Modus. Dabei gehen alle Einstellungen verloren und alle Variablen werden neu initialisiert. Einzelheiten dazu siehe Abschnitt „Aktivieren von Kompatibilitäts- und Unicode-Modus“ auf Seite 637.

Neben dem expliziten Umschalten mit der Anweisung @MODE ist auch ein implizites Umschalten vom Kompatibilitäts-Modus in den Unicode-Modus möglich, wenn alle Arbeitsdateien leer und keine Dateien geöffnet sind und eine @CODENAME-Anweisung mit einem ISO-Zeichensatz, einem Unicode-Zeichensatz oder einem anderen 7-Bit-Zeichensatz als `EDF03IRV` angegeben wird. Ein implizites Umschalten vom Unicode-Modus in den Kompatibilitäts-Modus findet hingegen nie statt.

11.4 Aktivieren von Kompatibilitäts- und Unicode-Modus

Wenn der EDT über /START-EDT im Dialog oder Stapelbetrieb gestartet wird, wird zunächst der Kompatibilitäts-Modus aktiviert. Ein direktes Starten des EDT im Unicode-Modus ist über das Kommando /START-EDTU möglich (siehe auch Abschnitt „Starten des EDT“ auf Seite 91). Wenn der EDT über eine der Unterprogramm-Schnittstellen gestartet wird, gelten die Regeln, die weiter unten in einer Tabelle zusammengefasst sind.

Ein automatischer Wechsel vom Kompatibilitäts-Modus in den Unicode-Modus findet schon während der Initialisierung des EDT statt, wenn

– der EDT im Dialogbetrieb aufgerufen wurde und für die DSS mit /MODIFY-TERMINAL-OPTIONS ein Unicode-Zeichensatz eingestellt wurde,

– der EDT im Prozedur- oder Stapelbetrieb aufgerufen wurde und der Zeichensatz der SYSDTA zugeordneten Datei (des Bibliothekselements) ein Unicode- bzw. ein ISO-Zeichensatz ist,

– der EDT eine EDT-Startprozedur findet, die in einem Unicode- bzw. ISO-Zeichensatz codiert ist.

Ein Wechsel vom Kompatibilitäts-Modus in den Unicode-Modus kann zu einem späteren Zeitpunkt stattfinden, wenn

– bei leeren Arbeitsdateien die @MODE- oder @CODENAME-Anweisung mit entsprechendem Operanden verwendet wird,

Ein Wechsel vom Unicode-Modus in den Kompatibilitäts-Modus findet statt, wenn

– bei leeren Arbeitsdateien eine @MODE-Anweisung mit entsprechendem Operanden verwendet wird.

Weitere Einzelheiten, insbesondere die Beschreibung der Operanden und ihrer Standardwerte findet man bei der Beschreibung der @MODE-Anweisung.

Da ein Wechsel zwischen Unicode- und Kompatibilitäts-Modus nur möglich ist, wenn alle Arbeitsdateien leer sind, können beim Umschalten keine Datenverluste oder Datenverfälschungen in Dateien auftreten. Das Umschalten erfolgt daher generell ohne Sicherheitsabfrage.

Der aktuell eingestellte Betriebsmodus ist im F-Modus am Layout der Zustandsanzeige in der Anweisungszeile zu erkennen (siehe Abschnitt „F-Modus“ auf Seite 105). Ferner wird sowohl im Unicode-Modus als auch im Kompatibilitäts-Modus bei der Ausgabe von @STATUS=ALL der aktuelle Betriebsmodus mit ausgegeben (diese Information ist auch im L-Modus verfügbar).
Schließlich kann mit der @IF-Anweisung in Prozeduren abgefragt werden, in welchem Betriebsmodus man sich befindet.

Wegen des Umschaltens nur bei leeren Arbeitsdateien ist eine Datenübertragung zwischen den Betriebsmodi auf diesem Wege nicht möglich. Darüber hinaus werden auch alle anderen globalen Daten, z.B. der Inhalt von Variablen oder Einstellungen, die mit @PAR festgelegt wurden, nicht in den jeweils anderen Modus übernommen, gehen also verloren. Vielmehr verhält sich der EDT so, als ob zwischen verschiedenen Instanzen des Programms, die nichts voneinander wissen, gewechselt würde.
Das Umschalten selbst ist dabei einem Beenden des EDT im einen Modus mit sofortigem Neustart im anderen Modus vergleichbar. Es laufen also folgende Schritte ab:

– Die Zeichenfolgevariablen `#S00` bis `#S20` werden im alten Modus in die korrespondierenden S-Variablen `SYSEDT-S00` bis `SYSEDT-S20` exportiert und im neuen Modus wieder initialisiert, sofern die S-Variablen existieren und vom Typ `STRING` sind.

– Die EDT-Startprozedur wird im neuen Modus neu ausgeführt (sofern die Voraussetzungen gemäß Abschnitt „EDT-Startprozedur“ auf Seite 74 dafür vorliegen).

– Im Übrigen werden alle Einstellungen des EDT auf ihre Anfangswerte zurückgesetzt, d.h. auch bei mehrmaligem Wechsel zwischen Kompatibilitäts- und Unicode-Modus merkt sich der EDT nicht die evtl. vorher getroffenen Einstellungen.

11.5 Unterprogrammschnittstellen und Betriebsmodi

Über die Unterprogramm-Schnittstellen kann zum einen der EDT geladen und initialisiert werden, zum anderen können die Unterprogramm-Schnittstellen in einer bereits initialisierten EDT-Umgebung aufgerufen werden (etwa nachdem mit @DIALOG in den F-Modus gewechselt wurde und der Anwender mit @HALT in das rufende Programm zurückgekehrt ist).
Wenn im zweiten Fall die Arbeitsdateien nicht leer sind, befindet sich der EDT entweder im Kompatibilitäts-Modus oder im Unicode-Modus und ein Umschalten ist nicht möglich. Unmittelbar nach der Initialisierung bzw. wenn die Arbeitsdateien leer sind, darf umgeschaltet werden.
Diese Fälle müssen bei Aufruf über die Unterprogramm-Schnittstelle unterschieden werden.
Die folgende Tabelle gibt eine Übersicht, welche Unterprogramm-Schnittstellen in welcher Umgebung erlaubt sind und wie die Aufrufe behandelt werden.
Dabei wird auch der Fall berücksichtigt, dass in der aktuellen Systemumgebung nur der EDT V16 verfügbar ist.

	EDT V17 nicht vorhanden	**EDT V17 vorhanden-Umschalten erlaubt und Aufuf über IEDTCMD**	**EDT V17 vorhanden-Kompatibilitätsmodus aktiv, Umschalten verboten**	**EDT V17 vorhanden-Unicode-Modus aktiv, Umschalten verboten**
Aufruf über IEDTGLE V16 Schnittstelle	Wie bisher	Kompatibilitäts-Modus einstellen	Keine Sonderbehandlung nötig	Umsetzen auf V17 Schnittstelle-wenn nicht möglich: Fehler
Aufruf über IEDTGLE V17 Schnittstelle (kompatibles Format)	Umsetzen auf V16 Schnittstelle	Unicode-Modus einstellen	Umsetzen auf V16 Schnittstelle	Keine Sonderbehandlung nötig
Aufruf über IEDTGLE V17 Schnittstelle (erweitertes Format)	Fehler	Unicode-Modus einstellen	Fehler	Keine Sonderbehandlung nötig
Aufruf über L-Modus Schnittstelle	Wie bisher	Kompatibilitäts-Modus einstellen	Keine Sonderbehandlung nötig	Fehler

Abhängig vom eingestellten Modus des EDT V17.0A und der verwendeten Unterprogramm-Schnittstelle stehen folgende Funktionen zur Verfügung:

L-Modus-Schnittstelle:

Alle Funktionen des Kompatibilitäts-Modus stehen zur Verfügung. Wenn der Unicode-Modus aktiviert ist, werden alle Aufrufe über die L-Modus-Schnittstelle mit dem RC `X'0C'` abgewiesen. Da ein Wechsel in den Unicode-Modus und zurück generell nur bei leeren Arbeitsdateien erlaubt ist, kann es nicht vorkommen, dass über die L-Modus-Schnittstelle auf die neue Satzstruktur des Unicode-Modus zugegriffen wird.

IEDTGLE-Schnittstelle (V16-Format):

Alle Funktionen des Kompatibilitäts-Modus stehen zur Verfügung. Wenn der Unicode-Modus aktiviert ist, kann `IEDTCMD`, `IEDTEXE`, `IEDTDEL`, `IEDTREN` uneingeschränkt genutzt werden. `IEDTPUT`, `IEDTGET` sind im Move-Mode erlaubt - es können Konflikte bei unzureichender Puffergröße auftreten. Die Länge des bereitgestellten Puffers konnte auch bisher schon zu kurz sein, um den Satz aufzunehmen. Es wurde und wird dann ein entsprechender Returncode geliefert.

IEDTGLE-Schnittstelle (kompatibles V17-Format):

Nur die Funktionen des Unicode-Modus werden angeboten, die sich in äquivalente Funktionen des V16-Formats umsetzen lassen. Dieses Format ist für Anwender gedacht, die zwar die neue Schnittstelle benutzen wollen, aber sichergehen müssen, dass ihre Programme auch mit EDT V16.6B noch laufen. Der Locate-Mode wird nicht mehr angeboten.

IEDTGLE-Schnittstelle (erweitertes V17-Format):

Nur die Funktionen des Unicode-Modus stehen zur Verfügung. Der Locate-Mode wird nicht mehr angeboten.

Einzelheiten zu den Unterprogramm-Schnittstellen werden im Handbuch „Unterprogramm-Schnittstellen“[1] beschrieben.

11.6 Zeichensätze

Mit dem EDT können Texte bearbeitet werden, die in unterschiedlichen Zeichensätzen vorliegen. Die Arbeitsweise im Kompatibilitäts-Modus unterscheidet sich dabei deutlich von der im Unicode-Modus. Da der EDT im Kompatibilitäts-Modus niemals Daten von einem Zeichensatz in einen anderen konvertiert, ist die Unterstützung in diesem Modus nur eingeschränkt möglich.

Im Kompatibilitäts-Modus hat der EDT immer genau einen Zeichensatz eingestellt.
Es können also gleichzeitig nur Texte bearbeitet werden, die im gleichen Zeichensatz vorliegen. Dieser Zeichensatz kann dann und nur dann geändert werden, wenn keine Daten in den Arbeitsdateien vorhanden sind. Er wird also festgelegt, wenn Daten in eine Arbeitsdatei gelangen. In diesem Zeichensatz kommuniziert der EDT auch mit der Datensichtstation bzw. liest und schreibt seine Ein- und Ausgaben bei anderen Quellen. Der EDT kann also insbesondere nur Dateien bearbeiten, deren vollständige Darstellbarkeit am Bildschirm garantiert ist. Bestimmte Zeichensätze sind ganz ausgeschlossen z.B. `EDF03DRV`, `EDF046` und ISO-Zeichensätze.

11.6.1 Unterstützte Zeichensätze

Der EDT lässt grundsätzlich nur Zeichensätze zu, die von der aktuellen XHCS-Installation unterstützt werden. Die notwendigen Konvertierungen werden über XHCS abgewickelt und auch die Eigenschaften der Zeichen (Groß-/Kleinbuchstaben, Sonderzeichen) werden von XHCS bereitgestellt.

Im Kompatibilitäts-Modus gibt es noch weitere Einschränkungen, die durch VTSU induziert sind. Der EDT lässt als einzigen 7-Bit-Zeichensatz `EDF03IRV` zu, auch wenn andere wie z.B. `EDF03DRV` in XHCS definiert sind, da `EDF03IRV` der einzige 7-Bit-Zeichensatz ist, der zur Kommunikation mit einer Datensichtstation verwendet werden kann.
Diese Einschränkung gilt auch im Stapelbetrieb. Weiterhin sind ausschließlich EBCDIC-Zeichensätze zugelassen, keine ISO-Zeichensätze. Kommuniziert der EDT mit einer Datensichtstation, muss der Zeichensatz mit dieser verträglich sein. Im Stapelbetrieb sind dagegen alle 8-Bit-EBCDIC-Zeichensätze zugelassen.

Der EDT akzeptiert alle Eingaben, auch wenn die Zeichen/Bytes nicht zum Code-Umfang gehören. Alle Bytes, deren Code nicht zum Code-Umfang des aktuellen Zeichensatzes gehört, werden an der Datensichtstation als Schmierzeichen dargestellt.
Sie werden bei der Umwandlung von Klein-/Großbuchstaben nicht berücksichtigt und sie sollten nicht als Sonderzeichen (z.B. Tabulatoren) verwendet werden (es gibt aber keine Vorkehrungen dagegen). Sie werden aber durch den EDT (normalerweise) nicht verändert.

Unicode-Zeichensätze und ISO-Zeichensätze sind im Kompatibilitäts-Modus nicht zulässig.

11.6.2 Zeichenfolgen

Die Interpretation und Verarbeitung aller Zeichenfolgen erfolgt immer in einem Zeichensatz.

Im Kompatibilitäts-Modus gibt es genau einen Zeichensatz, in dem Daten gelesen werden, in dem sie abgelegt werden, in dem sie angezeigt und bearbeitet werden und in dem sie geschrieben werden. Das ist immer der **aktuelle** Zeichensatz.

11.6.3 Kommunikationszeichensatz

Der Kommunikations-Zeichensatz ist der Zeichensatz, den der EDT zum Datenaustausch mit einer Datensichtstation verwendet.

Im Kompatibilitäts-Modus ist dies der im EDT aktuell eingestellte Zeichensatz. Damit gibt es keinen eigenständigen Kommunikations-Zeichensatz.

11.6.4 Zeichensätze in Arbeitsdateien

Die Behandlung von Zeichensätzen unterscheidet sich im Kompatibilitäts-Modus erheblich vom Verhalten im Unicode-Modus.

Im Kompatibilitäts-Modus hat der EDT immer genau einen Zeichensatz eingestellt. Er ermittelt beim Start einen Default-Wert für den Zeichensatz.
Liest der EDT seine Eingaben von einer Datensichtstation, nimmt er den Zeichensatz aus der Terminal-Option `CODED-CHARACTER-SET`. Ist dort `7-BIT` angegeben, nimmt er `EDF03IRV`, sonst den dort angegebenen Zeichensatz.
Liest der EDT seine Eingaben von `SYSDTA`, verwendet er den Zeichensatz der `SYSDTA` zugeordnet ist. Ist dort keiner angegeben `(*NONE)`, nimmt er `EDF03IRV`.

Der aktuelle Zeichensatz ist nach dem Start zunächst der Default-Zeichensatz. Solange sich in den Arbeitsdateien keine Daten befinden, kann er geändert werden. Sobald der erste Datensatz in eine Arbeitsdatei geschrieben wird, wird er festgelegt und kann erst dann wieder geändert werden, wenn alle Arbeitsdateien leer sind. Eine solche Änderung kann dann implizit beim Einlesen einer Datei geschehen oder explizit mit der @CODENAME-Anweisung.

Die @CODENAME-Anweisung wird nur ausgeführt, wenn der Zeichensatz zulässig ist und wenn sich keine Daten in den Arbeitsdateien befinden. Sonst wird die @CODENAME-Anweisung abgewiesen bzw. hat keine Wirkung.

Mit der @MODE-Anweisung kann das Umschalten in den Unicode-Modus erfolgen (siehe MODE-Anweisung).

Anmerkung

Mit einer @CODENAME-Anweisung kann der aktuelle Zeichensatz zwar geändert, aber nicht festgelegt werden. Beim Einlesen einer Datei wird daher evtl. der Zeichensatz wieder geändert..

11.6.5 Einlesen von Dateien

Der EDT wertet beim Einlesen einer Datei den Zeichensatz des Katalogeintrags aus.

Im Kompatibilitäts-Modus wird die Datei nur dann eingelesen, wenn entweder dieser Zeichensatz mit dem aktuellen Zeichensatz des EDT identisch ist, oder wenn keine Daten in den Arbeitsdateien vorhanden sind. Dann wird der entsprechende Zeichensatz eingestellt.
Sind in irgendeiner Arbeitsdatei schon Daten in einem anderen Zeichensatz vorhanden, wird die Datei nicht eingelesen. Die Datei wird ebenfalls nicht eingelesen, wenn der Zeichensatz nicht unterstützt wird, also wenn sie im Dialogbetrieb z.B. nicht an der Datensichtstation darstellbar ist.

11.6.6 Schreiben von Dateien

Beim Schreiben von neuen Dateien wird der Zeichensatz der Arbeitsdatei in den Katalog eingetragen.

Im Kompatibilitäts-Modus wird beim Schreiben von Dateien nach dem Schließen der Datei der Zeichensatz der Arbeitsdatei, also der aktuelle Zeichensatz des EDT, in den Katalog eingetragen.
Dabei gilt die Ausnahme, dass der EDT den Katalog-Eintrag nicht verändert, wenn die Datei den Zeichensatz `*NONE` hat und der aktuelle Zeichensatz `EDF03IRV` ist, d.h. `*NONE` bleibt erhalten.

11.6.7 Kopieren zwischen Arbeitsdateien

Mit Hilfe der @COPY-Anweisung, der @MOVE-Anweisung, der @ON-Anweisung bzw. mit Kurzkommandos können Daten von einer Arbeitsdatei in eine andere kopiert werden.

Im Kompatibilitäts-Modus ist dabei nur ein Zeichensatz beteiligt, so dass die Daten nicht konvertiert, sondern unverändert übertragen werden.

11.6.8 Zeichensatz einer Anweisung

Im Kompatibilitäts-Modus werden Anweisungen zwar im aktuellen Zeichensatz gelesen, die Anweisungs-Analyse erfolgt jedoch immer in EDF041. So wird etwa als EDT-Anweisungssymbol immer ein Byte mit X'7C' erwartet (falls nicht umdefiniert).
Wird die Anweisung in EDF04DRV gelesen, muss als EDT-Anweisungssymbol das Zeichen '§' verwendet werden, da dieses Zeichen die Codierung X'7C' hat und nicht das Zeichen '@'.
Analoges gilt auch, wenn Symbole umdefiniert wurden. Wurde z.B. in EDF041 als EDT-Anweisungssymbol das Währungszeichen '¤' vereinbart, so muss in EDF04F das Eurozeichen '€' verwendet werden.
Schließlich kann sogar ein Zeichen als Anweisungssymbol definiert werden, dass in einem anderen Zeichensatz gar kein Sonderzeichen ist. So ist z.B. nach @CODENAME EDF04DRV; §:|; @CODENAME EDF041 das Zeichen 'ä' das Anweisungssymbol. Da dies kein Sonderzeichen ist, werden im L-Modus überhaupt keine Anweisungen mehr erkannt.

Hinweis

Im Unicode-Modus wird diese Situation bereinigt, in dem die Anweisungs-Analyse immer in UTFE durchgeführt wird.

11.6.9 Zeichenfolgevariablen

Zeichenfolgevariablen können beliebige Texte aufnehmen, wie eine Zeile in einer Arbeitsdatei. Sie sind über Arbeitsdateien hinweg global zugreifbar. Jede Zeichenfolgevariable hat zu jedem Zeitpunkt einen Inhalt, da ihr bei der Initialisierung ein Zeichen '␣' (X'40') zugewiesen wird.

Im Kompatibilitäts-Modus haben Zeichenfolgevariablen keinen Zeichensatz. Ihre Inhalte werden immer im aktuellen Zeichensatz interpretiert.
Auch nach einem Löschen des Inhalts oder der Zuweisung eines Leerstrings wird die Zeichenfolgevariable wieder mit einem Zeichen '␣' (X'40') initialisiert.

Anmerkung

Über Umwege (Speichern von Daten in Zeichenfolgevariablen, Löschen aller Arbeitsdateien, Einfügen aus diesen Zeichenfolgevariablen) kann im Kompatibilitätsmodus erreicht werden, dass zwischen Speichern und Einfügen einer Zeichenfolge ein Wechsel des Zeichensatzes stattfindet. Dabei werden die zu kopierenden Daten natürlich unverändert übertragen. Das Ergebnis ist in der Regel nicht sinnvoll und wird in jedem Fall auch anders als im Unicode-Modus sein.
So enthält z.B. Zeile 1 einer Arbeitsdatei im Kompatibilitäts-Modus nach

```
@CODENAME EDF047
@CREATE 1 ' πλάτων '
@CREATE #s0:1
@DELETE
@CODENAME EDF041
@CREATE 1 #S0
```

den Inhalt **Ðëáôùí**.

Im Unicode-Modus wird versucht die Zeichenfolge ' **πλάτων** ' nach EDF041 zu konvertieren. Dies wird entweder abgelehnt oder es wird das Ersatzzeichen eingesetzt.

11.6.10 S-Variablen und Job-Variablen

Mit den Anweisungen @GETVAR, @GETLIST und @GETJV können die Inhalte von S-Variablen bzw. von Job-Variablen in Zeichenfolgevariablen oder Arbeitsdateien übernommen werden.
Im Kompatibilitäts-Modus wird der Inhalt unverändert d.h. ohne Berücksichtigung eines Zeichensatzes übernommen.
Mit den Anweisungen @SETVAR, @SETLIST und @SETJV können S-Variablen bzw. Job-Variablen erzeugt und ihnen ein Wert zugewiesen werden.
Im Kompatibilitäts-Modus wird der Inhalt unverändert d.h. ohne Berücksichtigung eines Zeichensatzes zugewiesen.

11.6.11 POSIX-Dateien

POSIX-Dateien führen ebenfalls keine Information über den Zeichensatz mit sich.

Im Kompatibilitäts-Modus kann bei den Anweisungen @XOPEN, @XCOPY und @XWRITE mit dem Operand CODE angegeben werden, ob es sich um eine Datei in EBCDIC (CODE=EBCDIC) oder in ASCII (CODE=ISO) handelt.

Im letzteren Fall wird beim Einlesen der Text mit einer festen Tabelle entsprechend ISO88591 -> EDF041, beim Schreiben mit einer festen Tabelle entsprechend EDF041-> ISO88591 konvertiert. Mit der Anweisung @PAR CODE=EBCDIC/ISO kann eine Voreinstellung gesetzt werden.

11.7 Starten des EDT

Der EDT wird im Kompatibilitäts-Modus mit dem Kommando /START−EDT geladen und gestartet. Das /START−EDT- Kommando entspricht dem des EDT V16.6B. Seine Beschreibung kann dem Handbuch „EDT V16.6B Anweisungen“ [2] entnommen werden.

Aus Kompatibilitätsgründen wird das Starten des EDT mittels /START−PROGRAM weiterhin unterstützt. Der EDT wird dann als Hauptprogramm mit einem der folgenden BS2000-Kommandos geladen und im Kompatibilitäts-Modus gestartet.

Kommando	**AMODE**
START-PROGRAM $.EDT	AMODE 31
START-PROGRAM *MODULE ($.SYSLNK.EDT.170, EDTC, RUN-MODE=*ADVANCED)	AMODE 24

12 Umstellhilfen

In diesem Kapitel werden als Umstellhilfe die Unterschiede zwischen Kompatibilitäts-Modus und Unicode-Modus aufgeführt.

12.1 Kompatibilitäts-Modus

Der Kompatibilitäts-Modus wurde um 2 Anweisungen erweitert.

Zum einen wurde die Anweisung @MODE eingeführt, mit der in den Unicode-Modus umgeschaltet werden kann.

Zum anderen wurde die Anweisung @IF(Format 5) eingeführt, die es erlaubt, den aktuellen Betriebs-Modus abzufragen und gegebenenfalls zu reagieren.

Des Weiteren wurden im Kompatibilitäts-Modus einige Fehler korrigiert. Diese Korrekturen können auch ein geändertes Verhalten hervorrufen.

12.2 Unicode-Modus

Der Unicode-Modus, in den allein die wesentlichen Erweiterungen des EDT V17.0A wie Unterstützung der Unicode-Zeichensätze und lange Sätze eingebaut wurden, verhält sich bei vielen Funktionen anders als der Kompatibilitäts-Modus. Generell gilt hier, dass Änderungen im L-Modus erheblich restriktiver vorgenommen wurden, als im F-Modus, da sie im Dialog weniger gravierende Auswirkungen haben.

In diesem Kapitel werden die Unterschiede zwischen Unicode-Modus und Kompatibilitäts-Modus zusammengefasst aufgeführt. Diese Hinweise sind vor allem zu beachten, wenn Prozeduren auf Unicode-Modus umgestellt werden sollen. Dies wird normalerweise nicht ohne, ggf. auch aufwändige, Anpassungen möglich sein.

Die Aufstellung ist in die folgenden Unterabschnitte gegliedert:

- Funktionen des EDT V16.6B, die im Unicode-Modus des EDT V17.0A nicht mehr unterstützt werden,
- Anweisungen deren Verhalten im Unicode-Modus des EDT V17.0A gegenüber dem EDT V16.6B geändert wurde,
- Änderungen bei der Bildschirm-Darstellung und bei der Ein-/Ausgabe im Unicode-Modus des EDT V17.0A gegenüber dem EDT V16.6B und
- Änderungen bei generellen oder arbeitsdateispezifischen Einstelungen im Unicode-Modus des EDT V17.0A gegenüber dem EDT V16.6B.
- Änderungen an der Unterprogramm-Schnittstelle, wenn das neue V17-Format der Schnittstelle benutzt wird oder wenn mit dem alten V16-Format der EDT V17.0A im Unicode-Modus bedient werden soll.

Im Zuge der Implementierung wurden zahlreiche Fehler des EDT V16.6B aufgedeckt, von denen viele nur im Unicode-Modus korrigiert wurden. Diese Korrekturen sind nicht alle in diesem Papier aufgeführt.

12.2.1 Nicht mehr unterstützte Funktionen

Die kompatible Syntaxkontrolle des L-Modus für die EDT V15-Anweisungen wird nicht mehr unterstützt. Der Operand `SECURITY` bei der Anweisung @SYNTAX entfällt. Diese Syntaxanalyse, die im Prozedurbetrieb voreingestellt war, war nicht zuverlässig und hat viele syntaktisch falsche Angaben akzeptiert. Das hat allerdings zur Folge, dass solche Angaben bei der Umstellung auf Unicode-Modus korrigiert werden müssen.

Die alte L-Modus-Unterprogrammschnittstelle wird nicht mehr unterstützt., da an ihr das interne Satzformat des EDT offen gelegt wurde. Als Unterprogrammschnittstelle steht nur noch die modernere `IEDTGLE`-Schnittstelle zur Verfügung.

Die alte Anweisung @RUN zum Aufruf eines Benutzerprogramms, wie sie im EDT V16.6B definiert war, gibt es im Unicode-Modus nicht mehr, da an dieser Schnittstelle ebenfalls das interne Satzformat des EDT offen gelegt wurde. Als Ersatz wurde eine neue Anweisung @RUN eingeführt bzw. kann auch die Anweisung @USE genutzt werden.

Der Locate-Modus bei der Datenübertragung über die `IEDTGLE`-Schnittstelle wird nicht mehr unterstützt. An dieser Schnittstelle wurde ebenfalls das interne Satzformat des EDT offen gelegt. Es kann nur noch der MOVE-Modus verwendet werden.

Datensichtstationen mit arabischem oder Farsi-Zeichensatz werden nicht mehr unterstützt. Diese speziellen Datensichtstationen eignen sich nicht für das Arbeiten mit Unicode-Zeichensätzen. Wenn erforderlich, können sie im Kompatibilitäts-Modus bedient werden.

Die Datensichtstation 3270 (IBM) wird nicht mehr unterstützt. Diese spezielle Datensichtstation eignet sich nicht für das Arbeiten mit Unicode-Zeichensätzen. Wenn erforderlich, kann sie im Kompatibilitäts-Modus bedient werden.

Die @CODE-Anweisung wird nicht mehr unterstützt. Sie ist bei korrekter Verwendung von Zeichensätzen für Dateien überflüssig.

Die @UPDATE-Anweisung wird nicht mehr unterstützt, da sie mit Einführung des F-Modus überflüssig geworden ist.

Die @ZERO-RECORDS-Anweisung wird nicht mehr unterstützt. Sie wird wegen der neuen Behandlung von leeren Sätzen nicht mehr benötigt.

Bei der Anweisung @LOG ist die direkte Angabe einer Listenvariablen als Ausgabemedium nicht mehr möglich. Man kann aber mit dem Kommando `/ASSIGN-SYSLST` der Systemdatei `SYSLST` eine Listenvariable zuweisen.

12.2.2 Geändertes Verhalten bei Anweisungen

Bei einer Reihe von Anweisungen wurde in speziellen Fällen das Verhalten geändert. Grund dafür war normalerweise, dass das alte Verhalten inkonsequent war und zu überraschenden Ergebnissen geführt hat. In vielen Fällen entspricht die Änderung eigentlich einer Fehlerkorrektur.

12.2.2.1 E/A-Anweisungen

Beim Lesen einer Datei (Anweisungen @OPEN, @COPY, @XOPEN, @XCOPY, @READ, @GET) werden Leerzeilen (Sätze der Länge 0) wie normale Zeilen behandelt. Bisher wurden Leerzeilen beim Lesen unterdrückt und gingen beim Zurückschreiben der Datei verloren.

Bei der Anweisung @OPEN (Format 1) ist für existierende DVS-Dateien die Einstellung `TYPE=CATALOG` Standardeinstellung. Bisher war dies die Einstellung `TYPE=SAM`.

Beim Schreiben oder Rückschreiben einer leeren Arbeitsdatei wird eine leere Datei erzeugt.

Beim Schreiben oder Rückschreiben in eine Datei, die das Dateiattribut `SECONDARY-ALLOCATION=0` hat, wird dieses Attribut berücksichtigt, d.h. die Datei wird im Bedarfsfall nicht erweitert und eine Fehlermeldung wird ausgegeben.

Ist in einer Arbeitsdatei eine Datei geöffnet, wird eine Anweisung @WRITE ohne Operanden nicht mehr abgewiesen, sondern als @WRITE (Format 1) erkannt und die Datei wird zurück geschrieben.

Das Schließen einer Datei (Anweisung @CLOSE) bzw. das Rückschreiben einer existierenden Datei (Anweisung @WRITE) kann eventuell abgelehnt werden, wenn die Datei einen anderen Zeichensatz hat als den, der für die Arbeitsdatei vereinbart ist und der Operand `CODE` nicht angegeben wurde. In EDT V16.6B konnte diese Situation nur herbeigeführt werden, wenn die Arbeitsdatei gelöscht und in einem anderen Zeichensatz neu aufgebaut wurde. In diesem Fall wurde der Zeichensatz der Datei ohne Rückfrage geändert.

Bei den Anweisung @XOPEN und @XWRITE wirkt `MODE=REPLACE` jetzt wie `MODE=NEW`, falls die Datei noch nicht existiert. Bisher wurden die Anweisungen in diesem Fall abgewiesen. Das Verhalten wurde damit dem von @OPEN und @WRITE angepasst.

Mit den Anweisungen @READ und @GET können auch Dateien mit vom Standard abweichenden Attributen gelesen werden.

12.2.2.2 Arbeitsdatei-Anweisungen

Beim vollständigen Löschen einer Arbeitsdatei mit der Anweisung @DELETE ohne Parameter wird eine in der Arbeitsdatei geöffnete Datei bzw. ein geöffnetes Bibliothekselement implizit geschlossen. Die Arbeitsdatei ist danach (wie bei @DROP) nicht mehr belegt. Bisher blieb die Datei geöffnet und war nach einem anschließenden @CLOSE leer. Wenn man dieses Verhalten wirklich wünscht, müssen alle Sätze mittels einer Bereichsangabe gelöscht werden.

Bei der Anweisung @DROP wird implizit ein @DELETE ausgeführt, d.h. eine geöffnete Datei wird ebenfalls geschlossen. Bisher blieb die Datei geöffnet, war aber nicht mehr zugreifbar.

Bei den Anweisungen @COPY (Format 2) und @MOVE wurde die Syntax in der Weise geändert, dass beim Kopieren bzw. Übertragen von Zeichenfolgevariablen die Angabe einer Quell-Arbeitsdatei nicht mehr akzeptiert wird. Da Zeichenfolgevariablen zu keiner Arbeitsdatei gehören, war diese Angabe sinnlos und verwirrend.

Bei den Anweisungen @COPY (Format 2) und @MOVE können mehrere Sendebereiche mit jeweils mehreren Empfangsbereichen durch Komma getrennt angegeben werden.

12.2.2.3 ON-Anweisungen

Die Unterscheidung nach Groß- und Kleinbuchstaben beim Suchen mit @ON (einstellbar durch die Anweisung @SEARCH-OPTION) wirkt jetzt unabhängig von der Einstellung von @PAR LOWER. Außerdem wirkt sie auch für die @ON-Formate 8 bis 10. Das bisherige Verhalten war unmotiviert und verwirrend, da über @PAR LOWER nur die Umwandlung von Zeichen bei der Eingabe von der Datensichtstation und die Darstellung von Kleinbuchstaben im Arbeitsfenster gesteuert wird.

Wird in der Anweisung @ON (Format 3) der Operand `MARK` nicht angegeben, wurden bisher die Trefferzeilen im L-Modus nicht markiert. Jetzt werden sie wie im F-Modus mit `1` markiert. Das bisherige Verhalten war inkonsequent, L-Modus und F-Modus sollten keine subtilen Unterschiede aufweisen.

Wird bei der Anweisung @ON rückwärts gesucht (Operand `R`), wird jetzt nach einem Treffer so weit nach links positioniert, dass ein weiterer Treffer nicht in einen vorhergehenden hineinragt. Das bisherige Verhalten hatte dazu geführt, dass beim Rückwärtssuchen u.U. andere Treffer angezeigt wurden als beim Vorwärtssuchen.

Bei der Anweisung @RANGE entfällt die Angabe eines Spaltenbereichs. Da dieser Spaltenbereich mit dem bei @RANGE definierte Bereichssymbol nichts zu tun hatte und ausschließlich bei der @ON-Anweisung berücksichtigt wurde, wird er zur Anweisung @SEARCH-OPTION verlagert.

12.2.2.4 Tabulatoren

Bei der Definition von Softwaretabulatoren mit der Anweisung @TABS (Format 2) muss jetzt zusätzlich zum Tabulatorzeichen immer mindestens eine Tabulatorposition angegeben werden. Die Tabulatorpositionen müssen aufsteigend sortiert sein. Bisher wurde eine @TABS-Anweisung ohne Tabulatorposition akzeptiert, hatte aber keinerlei Wirkung, nicht einmal das Tabulatorzeichen wurde geändert.

Hardware- und Software-Tabulatoren können unabhängig voneinander definiert werden. Jeweils einer davon kann aktiviert werden.

Bei der Anzeigefunktion der @TABS-Anweisung kann jetzt auch zwischen Hardware- und Software-Tabulatoren unterschieden werden. Die Anweisung @TABS VALUES gibt die Positionen der Hardware-Tabulatoren aus, die Anweisung @TABS ::VALUES die Positionen der Software-Tabulatoren sowie das definierte Tabulatorzeichen. Bisher wurden die Positionen des Hardware-Tabulators (wenn definiert) vorrangig ausgegeben. Die Anzeige der Positionen des Software-Tabulators war dann nicht möglich.

12.2.2.5 Sonstiges

Der Operand `ISO` bei der Anweisung @INPUT (Format 3) wird ignoriert. Bei Hexadezimaler Eingabe wird grundsätzlich die Codierung des Zeichensatzes der aktuellen Arbeitsdatei zugrunde gelegt.

Das Ausgabeformat der Anweisung @STATUS wird geändert, da einige Ausgaben nicht mehr relevant sind, einige globale Eigenschaften jetzt arbeitsdateispezifisch sind und zusätzliche Eigenschaften ausgegeben werden sollen.

Bei der Anweisung @VTCSET=OFF werden jetzt nur noch die Line-Modus-Steuerzeichen in Schmierzeichen umgesetzt, nicht mehr alle Zeichen, die nicht zum `EBCDIC-DF03`-Kern gehören. Dies entspricht dem Sinn der Anweisung, ein Zerreißen der Bildschirmausgabe durch Steuerzeichen zu verhindern.

Die Angabe einer Zeilenzahl für die Bildschirmausgabe in der Anweisung @VDT entfällt. Die Bildschirmausgabe der Anweisung @PRINT V bzw. @PRINT E erfolgt immer mit so vielen Zeilen, wie am Bildschirm darstellbar sind (das sind im L-Modus immer 23 Zeilen).

Bei der Anweisung @VDT F2 wird nicht mehr in Spalte 1 zurückpositioniert, sondern der Bildschirm bleibt bei der eingestellten Anfangsspalte.

Die Anweisung @VDT unterstützt jetzt auch die weiteren Formate F3 und F4.

Die Anweisungen @COMPARE, @FSTAT, @SHIH, @SHOW, @STAJV, @STATUS geben Informationen in Arbeitsdatei 9 aus. Wenn dort bereits eine Datei geöffnet ist, wird jetzt die Ausgabe abgelehnt. Bisher verhalten sich nur @SHIH und @SHOW LIBRARY so.

Die Zeichen, die mit der Anweisung @QUOTE als Begrenzersymbol definiert sind, dürfen nicht mit den Musterzeichen (siehe Anweisung @SYMBOLS) übereinstimmen. Bisher wurde zwar die Anweisung @SYMBOLS abgewiesen, wenn ein Begrenzersymbol als Musterzeichen vereinbart werden sollte, umgekehrt aber die Anweisung @QUOTE akzeptiert, wenn ein Musterzeichen als Begrenzersymbol vereinbart werden sollte, was evtl. zu Problemen bei der Syntaxanalyse von Anweisungen führen konnte.

Bei der Anweisung @DELETE MARK werden die Sondermarkierungen (Markierungen 13, 14 und 15) nicht mehr gelöscht. Bisher konnten die Sondermarkierungen zwar ausschließlich über die Unterprogramm-Schnittstelle gesetzt werden, waren jedoch gegen Löschen durch den normalen EDT-Anwender nicht geschützt. Dieses Verhalten entspricht auch der Beschreibung im V16.6B Handbuch.

Bei der Anweisung @SEPARATE wirkt die Angabe einer Spaltennummer jetzt rekursiv, d.h. entsteht durch das Auftrennen eines Satzes ein Folgesatz, der länger ist als die angegebene Spaltennummer, so wird der Folgesatz erneut umgebrochen. Das bisherige Verhalten, den Folgesatz nicht aufzutrennen, entsprach nicht dem Sinn der Angabe einer Spaltennummer als Auftrennpunkt.

Wird bei @SEPARATE ein Trennzeichen angegeben, können Sätze der Länge 0 entstehen, wenn mehrere Trennzeichen direkt hintereinander im Satz vorkommen. Diese Änderung ist eine Folge der konsistenten Behandlung von Leerzeilen.

Das Speichern der zuletzt eingestellten aktuellen Zeilennummer und aktuellen Schrittweite bei der Anweisung @SET (Format 6) sowie die Wiederherstellung der gespeicherten Einstellungen sind jetzt auch im F-Modus möglich. Bisher war diese Funktion auf den L-Modus beschränkt.

Es wird eine neue Anweisung @RUN mit geänderter Syntax und geänderter Schnittstelle eingeführt.

Bei der Anweisung @SHOW (Format 1) entfällt der Operand `ISO4`. Das Layout der Info-Zeile wurde angepasst.

Die Anweisung @DIALOG wird im Stapelbetrieb mit der Meldung EDT5400 abgewiesen. Im Kompatibilitäts-Modus und im EDT V16.6B wird sie bei gesetztem Schalter 5 ohne Meldung ignoriert. Ist der Schalter 5 nicht gesetzt wird sie im Kompatibilitäts-Modus mit der Meldung EDT5400, im EDT V16.6B mit der (unpassenden) Meldung EDT5409 abgewiesen.

Bei der Anweisung @SDFTEST wirkt die explizite Angabe des Programmnamens, dessen Anweisungen geprüft werden sollen, oder des Namenstyps nur noch für diesen Aufruf und verändert nicht implizit die für diese Werte durch @PAR SDF-PROGRAM bzw. durch @PAR SDF-NAME-TYPE vorgenommene Einstellung.

Bei der Anweisung @SDFTEST können jetzt Kommandos und Anweisungen mit mehr als 255 Folgezeilen geprüft werden (bis zur maximalen Pufferlänge von 16379 Bytes).

Ein mit der Kurzanweisung `S` markierter Satz wird immer auf Zeile 2 des Bildschirms positioniert. Ggf. werden dabei Info-Zeile und Tabulatorzeile zeitweise ausgeblendet.

Enthält eine Anweisung, die mit der Anweisung # in die Anweisungszeile geholt wird, Zeichen, die im aktuellen Kommunikationszeichensatz nicht dargestellt werden können, werden stattdessen `?` ausgegeben.

12.2.3 Änderungen bei der Bildschirm-Darstellung und bei der Ein-/Ausgabe

In der Zustandsanzeige im EDT-Arbeitsfensters (rechter Teil der Anweisungszeile) wird die Spaltennummer jetzt fünfstellig und die Nummer der Arbeitsdatei zweistellig angezeigt. Dadurch verkürzt sich die maximal mögliche Eingabelänge in der Anweisungszeile um drei Zeichen. Diese Änderung ist wegen der Unterstützung längerer Sätze und der Unterstützung aller 23 Arbeitsdateien im F-Modus erforderlich.

Das Ende eines Satzes im Datenfenster wird mit dem datenstationsspezifischen Zeichen [LZE] (*Logisches Zeilenende*) gekennzeichnet. Der Rest der Bildschirmzeile rechts vom [LZE] wird von der Datensichtstation mit geschützten `NIL`-Zeichen (`X'00'`) gefüllt. In diesem Bereich sind keine Eingaben mehr möglich. Eingaben müssen entweder vor dem [LZE] erfolgen oder das [LZE] muss überschrieben werden. Diese Änderung ist wegen der konsistenten Behandlung von Leerzeilen (Sätze der Länge 0) erforderlich.

Die Zeichen im Rest der Bildschirmzeile nach dem [LZE] können nicht mehr mit anderen Zeichen dargestellt werden, als mit dem von der Datensichtstation hardwaremäßig dafür vorgesehene Zeichen (normalerweise `NIL`). Die EDT-Anweisung @SYMBOLS FILLER='char' kann also im Unicode-Modus nicht mehr dazu verwendet werden, das im F-Modus zwischen Satzende und Bildschirmzeilenende dargestellte Füllzeichen festzulegen. Diese Änderung ist wegen der konsistenten Behandlung von Leerzeilen (Sätze der Länge 0) erforderlich.

Die Funktionstasten [K4] bis [K15] werden bei der Datenübertragung im F-Modus jetzt wie [K3] behandelt. Bisher war das Verhalten des EDT bei Eingabe dieser Tasten undefiniert.

Im L-Modus erzeugt eine leere Eingabe mit der Taste [F1] eine Leerzeile (Satz der Länge 0). Bisher wurde [F1] wie [DUE] behandelt. Diese Änderung ist wegen der konsistenten Behandlung von Leerzeilen (Sätze der Länge 0) erforderlich.

In einem Arbeitsfenster werden Spaltenzähler, Tabulatoranzeige oder Informationszeile nur noch dann angezeigt, wenn das Arbeitsfenster groß genug ist, um noch mindestens eine Datenzeile anzuzeigen. Andernfalls werden alle oder einige der zusätzlichen Anzeigen ausgeblendet. Bisher konnte man das Fenster soweit verkleinern, dass zwar die Zusatzinformationen sichtbar waren, aber keine Datenzeile.

Die Info-Zeile (@PAR INFO=ON) hat ein neues Layout.

Hat ein Arbeitsfenster so wenige Zeilen, dass im Hexadezimalmodus die Darstellung einer vollständigen Datenzeile nicht möglich ist, dann wird nicht in den Hexadezimalmodus umgeschaltet bzw. ein schon eingeschalteter Hexadezimalmodus ausgeschaltet. Da eine Datenzeile, die in Unicode codiert ist, u.U. die Anzeige von bis zu 6 Hexzeilen erfordert, ist ein genügend großes Arbeitsfenster Voraussetzung für sinnvolles Arbeiten im Hexadezimalmodus.

Im Hexadezimalmodus werden in den Hexzeilen nur noch Hexadezimalziffern (`0...9`, `A...F`) und `NIL`-Zeichen dargestellt und auch nur solche und das Leerzeichen als Eingabe zugelassen. Nullbytes am Satzende werden nicht entfernt und noch leere Sätze hinter dem Dateiende, die noch im Datenfester angezeigt werden, werden auch in den Hexzeilen durch `NIL`-Zeichen dargestellt. Das bisherige Verhalten war uneinheitlich und nicht dokumentiert.

Der Fehlerdialog im Hexadezimalmodus wurde geändert. Wird nicht korrigiert, wird der Dialog abgebrochen und die fehlerhaften Angaben werden verworfen.

Im Testmodus erfolgen alle Ausgaben jetzt einheitlich nach `SYSLST`. Bisher wurde zwar das normale Logging nach `SYSLST` geschrieben, als fehlerhaft markierte Ausgaben aber nach `SYSOUT`.

Bei der Ausgabe nach `SYSLST` wird die Ausgabe der Vorschubsteuerzeichens `X'41'` ersetzt durch `X'40'` und die Ausgabe einer zusätzlichen Leerzeile vor der jeweiligen Zeile, falls `SYSLST` den Zeichensatz `UTFE` hat. Diese Änderung war notwendig, um auch in `SYSLST`-Dateien, die in `UTFE` codiert sind, interpretierbare 1-Byte Vorschubsteuerzeichen zu erzeugen.

Gehen Zeilenvorschübe über eine Seitengrenze, werden auf der neuen Seite keine weiteren Leerzeilen erzeugt. Die übrigen Zeilenvorschübe werden verworfen.

Bei der Ausgabe eines Inhaltsverzeichnisses einer Bibliothek mit der Anweisung @SHOW (Format 1) wird die Überschriftzeile nicht mehr in die erste Zeile der Arbeitsdatei ausgegeben.

12.2.4 Änderungen bei generellen oder arbeitsdateispezifischen Einstellungen

12.2.4.1 Zeichensätze

Jede Arbeitsdatei kann einen eigenen Zeichensatz haben. Dies ist eines der wesentlichen neuen Features des EDT V17.0A im Unicode-Modus.

Beim Start des EDT und nach dem vollständigen Löschen einer Arbeitsdatei ist kein Zeichensatz mehr für die Arbeitsdatei eingestellt (`*NONE`). Diese Änderung ist eine Folge der Unterstützung eines eigenen Zeichensatzes für jede Arbeitsdatei.

Beim Einlesen von Daten in eine Arbeitsdatei werden diese in den Zeichensatz der Arbeitsdatei konvertiert, falls er ungleich `*NONE` ist. Eine implizite Umschaltung des Zeichensatzes findet nur mehr dann statt, wenn der Zeichensatz der Arbeitsdatei `*NONE` ist. Diese Änderung ist eine Folge der Unterstützung eines eigenen Zeichensatzes für jede Arbeitsdatei.

Beim Start des EDT ist @PAR LOWER=ON für alle Arbeitsdateien eingestellt. Für einen Editor, der Unicode-Zeichensätze unterstützt, ist das Starten im Großbuchstaben-Modus nicht mehr vermittelbar.

12.2.4.2 Zeilennummern

Es gibt nur noch eine aktuelle Schrittweite. Diese kann sowohl mit @SET (Format 6) als auch mit @PAR INCREMENT verändert werden. Bisher gab es zwei verschiedene Schrittweiten, die bei unterschiedlichen Anweisungen Verwendung fanden, ohne dass dies klar dokumentiert war.

Bei verschiedenen Anweisungen werden Zeilen eingefügt. Wenn bei einer Schrittweite von `0.01` eine Menge von Zeilen nicht eingefügt werden kann, wird der Einfügevorgang nicht mehr abgebrochen, sondern es wird versucht, bis zur kleinsten Schrittweite von `0.0001` einzufügen. Der bisherige Abbruch des Einfügevorgangs war unmotiviert.

Bei Anweisungen, die ihre Ausgabe in eine leere (oder implizit gelöschte) Arbeitsdatei schreiben, z.B. @COMPARE oder @SHOW, erfolgt die Vergabe der Zeilennummern einheitlich nach dem Verfahren, wie es bei der @COPY-Anweisung praktiziert wird, d.h. es wird ggf. die Schrittweite verkleinert und umnummeriert, um alle Zeilen ausgeben zu können. Das bisherige Verhalten war uneinheitlich und daher verwirrend.

12.2.4.3 Arbeitsdateispezifisch

Die symbolische Zeilennummer `?` existiert jetzt arbeitsdateispezifisch.

Die mit der Anweisung @IF (Format 3) abzufragende Bedingung, ob bei der letzten Ausführung die @ON-Anweisung einen Treffer festgestellt und die dazugehörige Spaltennummer werden ebenfalls arbeitsdateispezifisch geführt.

12.2.4.4 Sonstiges

Für den Operandentyp Sonderzeichen (`spec`) sind jetzt nicht mehr alle Sonderzeichen zugelassen, sondern nur die in der Operandenbeschreibung explizit aufgeführten Sonderzeichen. Diese Änderung ist erforderlich, da die Sonderzeichen eine Spezialbedeutung bei der Syntaxanalyse haben und exotische Mehrbyte-Sonderzeichen aus Unicode-Zeichensätzen hier Probleme machen würden.

Der Programmname für die SDF-Syntax-Prüfung, der Typ des Programmnamens für die SDF-Syntax-Prüfung und das Zeichen für das Auftrennen von Datensätzen werden jetzt arbeitsdateispezifisch, nicht mehr global definiert. Diese Änderung dient der Vereinheitlichung des Verhaltens bei der @PAR-Anweisung. Es gibt jetzt nur noch zwei Ausnahmen von der Regel, dass mit @PAR Einstellungen für jede Arbeitsdatei separat getroffen werden können.

Der Initialwert der symbolischen Zeilennummer für Trefferzeilen `?` ist jetzt `0.0000` statt `0.0001`. Durch die Voreinstellung eines ungültigen Wertes kann erkannt werden, ob überhaupt schon eine Suchanweisung gegeben wurde.

Die Sondermarkierung 13 (Satz soll beim Schreiben der Datei ignoriert werden), wird jetzt auch für POSIX-Dateien berücksichtigt. Die bisherige Ausnahmeregelung für POSIX-Dateien war unmotiviert.

Die Angaben in Anweisungen von Dateinamen, Bibliothekselementen und Jobvarablennamen werden syntaktisch geprüft und ggf. durch eine Syntaxfehlermeldung zurückgewiesen.

Alle unterbrechbaren Anweisungen können jetzt mit K2 und `/INFORM-PROGRAM` abgebrochen werden. Bisher war das Verhalten uneinheitlich.

Die bisherige Behandlung europäischer 7-Bit-Terminals wird nicht mehr unterstützt, da sie nicht zur allgemeinen Unterstützung der 7-Bit-Zeichensätze passt. Zur Unter-stützung des Zeichensatzes `EDF03DRV`, des einzigen in XHCS bekannten nationalen 7-Bit-Codes, wurde eine spezielle Behandlung implementiert.

12.2.5 Änderungen an der Unterprogramm-Schnittstelle

In diesem Abschnitt werden die Änderungen beschrieben, die abgesehen vom Wegfall der L-Modus-Schnittstelle und Ablösung der @RUN-Schnittstelle (siehe Abschnitt „Nicht mehr unterstützte Funktionen“ auf Seite 648) von Anwendern zu beachten sind, die in ihren Programmen die IEDTGLE-Schnittstelle benutzen.

Dabei muss unterschieden werden, ob das V16-Format der IEDTGLE-Schnittstelle (alte Makros bzw. neue Makros mit entsprechendem VERSION-Parameter), das V17-Format der IEDTGLE-Schnittstelle (neue Makros mit entsprechendem VERSION-Parameter) oder das kompatible V17-Format (steuerbar über das Flag EGLCOMP) benutzt wird. Das kompatible V17-Format ist dann zu benutzen, wenn die Anwendung auch in Umgebungen laufen soll, in denen nur ein EDT mit einer Version klener als V17.0A installiert ist.

Werden im V16-Format Lese- oder Schreib-Operationen im Locate-Modus aufgerufen, wenn sich der EDT im Unicode-Modus befindet und ein Umschalten in den Kompatibilitäts-Modus nicht möglich ist (z.B. weil Arbeitsdateien nicht leer sind), wird der Aufruf mit Returncode abgewiesen.

Beim Lesen des globalen Status (Funktion IEDTGET mit Pseudo-Arbeitsdatei 'G') im V16-Format, wird das im V17-Format entfallene Feld für den global eingestellten Zeichensatz nur dann mit einem Wert ungleich Leerzeichen versorgt, wenn in allen nicht leeren Arbeitsdateien der gleiche Zeichensatz eingestellt ist.

Im V16-Format werden Zeichenfolgen, die im Puffer EDTREC übertragen werden, in dem Zeichensatz interpretiert, der für die betroffene Arbeitsdatei eingestellt ist. Funktionen des V16-Formats, die mit den Puffern COMMAND, MESSAGE1 und MESSAGE2 arbeiten, können im Unicode-Modus nur dann korrekt bearbeitet werden, wenn entweder in den Kompatibilitäts-Modus umgeschaltet werden kann oder wenn für alle Arbeitsdateien der gleiche Zeichensatz eingestellt ist (siehe Anweisung @CODENAME GLOBAL,...), andernfalls wird die jeweilige Funktion mit Returncode abgewiesen.

In beiden Formaten muss das aufrufende Programm damit rechnen, dass im Unicode-Modus Sätze mit einer Länge größer als 256 Byte geliefert werden bzw. dass längere Sätze gekürzt werden. Wenn kein ausreichend großer Puffer bereitgestellt wird, muss zumindest der Returncode (EAMAC04) ausgewertet und angemessen behandelt werden.

In beiden Formaten muss das aufrufende Programm damit rechnen, dass im Unicode-Modus Sätze mit der Länge 0 geliefert werden.

In beiden Formaten liefert die Funktion IEDTINF im Feld EGLINFM (Anzahl der für den statischen Datenbereich benötigten Speicherseiten) des Kontrollblocks EDTGLCB immer den Wert 0 zurück.

Im V17-Format des Kontrollblocks EDTGLCB ist das Flag EGLREOR (Speicher-Reorganisation unterbinden) entfallen, die Felder EGLCCSN (Name des Zeichensatzes, in dem die Puffer COMMAND, MESSAGE1, MESSAGE2 bzw. EDTREC codiert sind) und EGLCOMP (Flag für das kompatible V17-Format), das Feld EGLIND2 (Anzeige, dass der EDT im Kompatibilitäts-Modus läuft) sowie der Returncode EUPCMPER (Benutzung von inkompatiblen Funktionen im kompatiblen Format) sind neu. Im kompatiblen V17-Format darf das Feld EGLCCSN nur Leerzeichen enthalten.

Im V17-Format des Kontrollblocks EDTUPCB ist das Flag EUPNUNI (Sperren des Umschaltens vom Unicode- zum Kompatibilitäts-Modus) neu. Im kompatiblen V17-Format darf EUPNUNI nicht gesetzt sein.

Im V17-Format des Kontrollblocks EDTAMCB sind das Feld EAMMMODB und die beiden Flags EAMMOVM und EAMLOCM (alle im Zusammenhang mit dem MOVE-Mode) entfallen. Ferner sind die Equates für die nicht verwendeten Markierungen (EAMMK10, EAMMK11, EAMMK12 und EAMMK0) entfallen.

Im V17-Format des Kontrollblocks EDTPARG ist das Feld EPGCCSN (Name des im EDT global eingestellten Zeichensatzes) entfallen. Es ist jetzt mit dem Namen EPLCCSN im Kontrollblock EDTPARL lokal für jede Arbeitsdatei vorhanden.

Im V17-Format des Kontrollblocks EDTPARL ist das Feld EPLCCSN neu (früher mit dem Namen EPGCCSN im Kontrollblock EDTPARG). Ferner ist das Feld EPLCCSNG (Zeichensatz gilt global für alle Arbeitsdateien) neu und die Felder EPLSTCOD (Code-Voreinstellung EBCDIC/ISO) und EPLOPNXC (Code der POSIX-Datei EBCDIC/ISO) sind entfallen.

Sollen externe Anweisungsroutinen (mittels @USE-Anweisung definierte Benutzeranweisungen) benutzt werden, die das V17-Format der Unterprogramm-Schnittstelle in ihrer Implementierung verwenden, muss der Ersteller der Anweisungsroutine zusätzlich eine Initialisierungsroutine bereitstellen. Am Vorhandensein der Initialisierungsroutine erkennt der EDT, dass das V17-Format verstanden wird.

13 Meldungen

Die Meldungen des EDT werden von der BS2000-Komponente MIP (*Message Improvement Processing*) bereitgestellt. Bei der Ausgabe einer Meldung können aktuelle Einfügungen (*Inserts*) mitgegeben werden.
Der Anwender kann den Umfang (`/MODIFY-JOB-OPTIONS INFORMATION-LEVEL`) und die Sprache (`/MODIFY-MSG-ATTRIBUTES TASK-LANGUAGE`) der Meldungsausgabe festlegen.

13.1 Meldungsgewichte

Die Meldungen des EDT werden durch einen 7-stelligen Meldungsschlüssel identifiziert. Dieser ist nach folgendem Schema aufgebaut:

`EDTwnnn`

`w` Meldungsgewicht

`nnn` laufende Nummer

Die Meldungsgewichte haben folgende Bedeutung:

Meldungs-gewicht	**Bedeutung**
`0`	Information
`1`	Warnung
`2`	leichte Fehler (z.B. Arbeitsdatei leer, Zeilenlänge überschritten)
`3`	Syntaxfehler
`4` und `5`	Funktionsfehler (auch Systemfehler, DVS-Probleme)
`8`	Beendigung und Abbruchfehler

Die aufgetretenen Meldungsgewichte beeinflussen den Kommando-Returncode des EDT (siehe Kapitel 4 „Ablauf des EDT“ auf Seite 91). Innerhalb einer Meldungsgewichtsklasse sind die Meldungen zum Teil nach Anwendungsgebieten gruppiert, zum Teil aber einfach fortlaufend durchnummeriert.

13.2 Fehlerschalter

Im L-Modus gibt es zwei Fehlerschalter, den EDT-Fehlerschalter und den DVS-Fehlerschalter. Der EDT-Fehlerschalter wird für den Großteil der Meldungen gesetzt. Er zeigt nicht korrekte EDT-Anweisungen an. Der DVS-Schalter wird für Meldungen gesetzt, die bei Datei- oder bei Systemzugriffsfehlern ausgegeben werden. Bei manchen Meldungen werden sowohl der EDT- als auch der DVS-Fehlerschalter gesetzt. Daneben gibt es Meldungen, für die kein Fehlerschalter gesetzt wird (z.B. Informationen, Abfragemeldungen, Meldungen, die nur im F-Modus ausgegeben werden). Die Fehlerschalter können in EDT-Prozeduren abgefragt werden (siehe Anweisungen @IF (Format 1) und @RESET in Kapitel 9). Bei welchen Meldungen ein Fehlerschalter gesetzt wird, ist den Hilfetexten der Meldungen zu entnehmen.

13.3 Meldungen, die eine Antwort erfordern

Im Dialogbetrieb fordert eine Reihe von Meldungen den Benutzer zu einer Antwort auf. Das Format dieser Meldungen ist

```
EDTOnnn <Question>? REPLY (Y=YES; N=NO)?
EDTOnnn <Frage>? ANTWORT (J=JA; N=NEIN)?
```

Die zur Verfügung stehenden Alternativen sind dabei *Ja* oder *Nein*. Die Antworten `Y`, `y`, `J` und `j` werden als *Ja* interpretiert, die Antworten `N` und `n` sowie eine leere Eingabe als *Nein*. Besteht die Antwort aus mehr als einem Zeichen, werden die restlichen Zeichen ignoriert. Die Reaktion auf andere Zeichen als die genannten ist vom Arbeitsmodus abhängig. Im L-Modus führen alle anderen Zeichen zur Wiederholung der Frage. Erfolgt auf die zehnte Wiederholung immer noch keine korrekte Antwort, so wird *Nein* angenommen. Im F-Modus wird die Eingabe anderer Zeichen als *Nein* interpretiert.

13.4 Meldungsausgabe

Die Meldungen werden im L-Modus-Dialog nach SYSOUT, im Stapelbetrieb nach SYSLST ausgegeben.

Im F-Modus werden die Meldungen in der letzten Zeile des Datenfensters, der so genannten Meldungszeile ausgegeben. Die Datenfenster-Anzeige ist dann um eine Zeile verkürzt. Ist die Meldung länger als die Meldungszeile, etwa weil ein *Insert* sehr lang ist, wird sie in den letzten beiden Zeilen des Datenfensters ausgegeben und die Datenfenster-Anzeige ist um eine weitere Zeile verkürzt. Steht bei einem geteilten Bildschirm nur eine Datenzeile zur Verfügung, wird die Meldung abgekürzt in dieser Zeile ausgegeben. Einige Meldungen fordern den Benutzer zu einer Antwort auf. Sie werden normalerweise auch in der Meldungszeile ausgegeben. In manchen Fällen wird die Fragemeldung jedoch gemeinsam mit einer anderen Meldung ausgegeben. Dann wird diese Meldung in der Meldungszeile und die Fragemeldung in der Anweisungszeile des Bildschirms ausgegeben. In allen Fällen muss die Antwort in der ersten Spalte der Anweisungszeile eingegeben werden. Führt ansonsten eine EDT-Anweisung zu mehr als einer Meldung, so wird nur eine davon in der Meldungszeile ausgegeben, und zwar die mit dem höchsten Meldungsgewicht. Sind die Meldungsgewichte gleich, wird die zuletzt erzeugte Meldung ausgegeben.

13.5 Meldungstexte

```
EDTCOPY    Copyright (C) (&00) (&01) All Rights Reserved
EDTCOPY    Copyright (C) (&00) (&01) Alle Rechte vorbehalten

EDTLOAD    Program '(&00)', Version '(&01)' of '(&02)' loaded from file '(&03)'
EDTLOAD    Programm '(&00)', Version '(&01)' vom '(&02)' aus Datei '(&03)' geladen

EDTSTRT    Procedure '(&00)', Version '(&01)' of '(&02)' started from file '(&03)'
EDTSTRT    Prozedur '(&00)', Version '(&01)' vom '(&02)' aus Datei '(&03)' gestartet

EDT0001    (&00) STARTED
EDT0001    (&00) GESTARTET
```

Bedeutung
Der EDT wurde gestartet.

```
EDT0002    (&00) RESTARTED IN COMPATIBILITY MODE
EDT0002    (&00) IM KOMPATIBILITAETSMODUS NEU GESTARTET
```

Bedeutung
Es wurde eine Anweisung zum Wechsel des Betriebsmodus eingegeben. Der EDT wurde im Kompatibilitaetsmodus neu gestartet.

```
EDT0003    (&00) RESTARTED IN UNICODE MODE
EDT0003    (&00) IM UNICODE-MODUS NEU GESTARTET
```

Bedeutung
Es wurde eine Anweisung zum Wechsel des Betriebsmodus eingegeben. Der EDT wurde im Unicode-Modus neu 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: wird nicht gesetzt.

```
EDT0110    TESTMODE: SYNTAX CANNOT BE TESTED
EDT0110    TESTMODE: SYNTAX KANN NICHT GEPRUEFT WERDEN
```

Bedeutung
Der Testmodus ist eingeschaltet. Die Anweisung wird nicht ausgefuehrt. Sie kann aber nur bei der Ausfuehrung syntaktisch geprueft werden, da sie z.B. indirekte Operanden oder Operanden in Variablen enthaelt oder eine Benutzeranweisung ist.
Fehlerschalter: wird nicht gesetzt.

```
EDT0120    TESTMODE: CHARACTER(S) SKIPPED
EDT0120    TESTMODUS: ZEICHEN WURDEN UEBERLESEN
```

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
```

Bedeutung
Der alte Inhalt des Bibliothekselements wurde ersetzt.

```
EDT0171    FILE '(&00)' REPLACED AND WRITTEN
EDT0171    DATEI '(&00)' ERSETZT UND GESCHRIEBEN
```

Bedeutung
Der alte Inhalt der Datei wurde ersetzt.

```
EDT0172    MEMBER '(&00)' IN LIBRARY '(&01)' CREATED AND WRITTEN
EDT0172    ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' ERZEUGT UND GESCHRIEBEN
```

Bedeutung
Das Bibliothekselement wurde erzeugt und geschrieben.

```
EDT0173    FILE '(&00)' CREATED AND WRITTEN
EDT0173    DATEI '(&00)' ERZEUGT UND GESCHRIEBEN
```

Bedeutung
Die Datei wurde erzeugt und geschrieben.

```
EDT0178    FILE '(&00)' CLOSED
EDT0178    DATEI '(&00)' GESCHLOSSEN

EDT0190    WORK FILE (&00) EMPTY
EDT0190    ARBEITSDATEI (&00) LEER

EDT0193    WORK FILE (&00) CLEARED
EDT0193    ARBEITSDATEI (&00) GELOESCHT
```

```
EDT0196   UFS FILE '(&00)' REPLACED AND WRITTEN
EDT0196   UFS DATEI '(&00)' ERSETZT UND GESCHRIEBEN
```

Bedeutung
Der alte Inhalt der POSIX-Datei wurde ersetzt.

```
EDT0197   UFS FILE '(&00)' CREATED AND WRITTEN
EDT0197   UFS DATEI '(&00)' ERZEUGT UND GESCHRIEBEN
```

Bedeutung
Die POSIX-Datei wurde erzeugt und geschrieben.

```
EDT0200   CCS CHANGED TO '(&00)'
EDT0200   CCS GEAENDERT AUF '(&00)'
```

Bedeutung
Da eine Datei oder ein Bibliothekselement mit der Code-Eigenschaft (&00) gelesen oder geoeffnet 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
```

Bedeutung
Die SDF-P-Listenvariable (&00) wurde erweitert, indem am Ende oder am Anfang ein oder mehrere Elemente hinzugefuegt wurden.

```
EDT0211   /FREE-VARIABLE COMMAND PROCESSED FOR S-VARIABLE '(&00)'
EDT0211   /FREE-VARIABLE KOMMANDO FUER S-VARIABLE '(&00)' AUSGEFUEHRT
```

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.

```
EDT0227   ISAM FILE '(&00)' CREATED AND OPENED IN WORK FILE (&01)
EDT0227   ISAM-DATEI '(&00)' ERZEUGT UND GEOEFFNET IN ARBEITSDATEI (&01)
```

Bedeutung
Die ISAM-Datei wurde erzeugt und in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0228   ISAM FILE '(&00)' REPLACED AND OPENED IN WORK FILE (&01)
EDT0228   ISAM-DATEI '(&00)' ERSETZT UND GEOEFFNET IN ARBEITSDATEI (&01)
```

Bedeutung
Der alte Inhalt der ISAM-Datei wurde geloescht, dann wurde sie in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0229   ISAM FILE '(&00)' OPENED IN WORK FILE (&01)
EDT0229   ISAM-DATEI '(&00)' GEOEFFNET IN ARBEITSDATEI (&01)
```

Bedeutung
Die ISAM-Datei wurde in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0230   FILE '(&00)' OPENED IN CURRENT WORK FILE (&01)
EDT0230   DATEI '(&00)' IN AKTUELLER ARBEITSDATEI (&01) GEOEFFNET
```

Bedeutung
Die Datei wurde in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0231   FILE '(&00)' CREATED AND OPENED IN CURRENT WORK FILE (&01)
EDT0231   DATEI '(&00)' IN AKTUELLER ARBEITSDATEI (&01) ANGELEGT UND GEOEFFNET
```

Bedeutung
Die Datei wurde erzeugt und in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0232   FILE '(&00)' REPLACED AND OPENED IN WORK FILE (&01)
EDT0232   DATEI '(&00)' ERSETZT UND GEOEFFNET IN ARBEITSDATEI (&01)
```

Bedeutung
Der alte Inhalt der Datei wurde geloescht, dann wurde sie in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0235   FILE '(&00)' WRITTEN AND CLOSED
EDT0235   DATEI '(&00)' GESCHRIEBEN UND GESCHLOSSEN
```

Bedeutung
Die Datei wurde geschrieben und geschlossen.

```
EDT0236   FILE '(&00)' CLOSED UNCHANGED
EDT0236   DATEI '(&00)' UNVERAENDERT GESCHLOSSEN
```

Bedeutung
Die Datei wurde unveraendert geschlossen.

```
EDT0237   UFS FILE '(&00)' OPENED
EDT0237   UFS DATEI '(&00)' GEOEFFNET
```

Bedeutung
Die POSIX-Datei wurde in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0238   UFS FILE '(&00)' CREATED AND OPENED
EDT0238   UFS DATEI '(&00)' ANGELEGT UND GEOEFFNET
```

Bedeutung
Die POSIX-Datei wurde erzeugt und in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0239   UFS FILE '(&00)' REPLACED AND OPENED
EDT0239   UFS DATEI '(&00)' ERSETZT UND GEOEFFNET
```

Bedeutung
Der alte Inhalt der POSIX-Datei wurde geloescht, dann wurde sie in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0240   UFS FILE '(&00)' CLOSED
EDT0240   UFS DATEI '(&00)' GESCHLOSSEN
```

Bedeutung
Die POSIX-Datei wurde geschrieben und geschlossen.

```
EDT0241   UFS FILE '(&00)' CLOSED UNCHANGED
EDT0241   UFS DATEI '(&00)' UNVERAENDERT GESCHLOSSEN
```

Bedeutung
Die POSIX-Datei wurde unveraendert geschlossen.

```
EDT0242   FILE '(&00)' COPIED
EDT0242   DATEI '(&00)' KOPIERT
```

Bedeutung
Die Datei wurde in die aktuelle Arbeitsdatei kopiert.

```
EDT0243   UFS FILE '(&00)' COPIED
EDT0243   UFS DATEI '(&00)' KOPIERT
```

Bedeutung
Die POSIX-Datei wurde in die aktuelle Arbeitsdatei kopiert.

```
EDT0244   ALLOW WRITE ACCESS FOR READ ONLY FILE? REPLY (Y=YES; N=NO)
EDT0244   SCHREIBZUGRIFF AUF SCHREIBGESCHUETZTE DATEI ERLAUBEN? ANTWORT (J=JA; N=NEIN)
```

Bedeutung
Abfrage nach den Anweisungen @OPEN, @WRITE, @XOPEN oder @XWRITE, wenn der Zugriff aus der TSOS-Benutzerkennung erfolgt und die POSIX-Datei schreibgeschuetzt ist.

Maßnahme
J: Die Datei wird ueberschrieben bzw. zum Schreiben geoeffnet
N: Die Datei wird nicht ueberschrieben bzw. nicht zum Schreiben geoeffnet.

```
EDT0258   MEMBER '(&00)' IN LIBRARY '(&01)' OPENED
EDT0258   ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' GEOEFFNET
```

Bedeutung
Das Bibliothekselement wurde in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0259   MEMBER '(&00)' IN LIBRARY '(&01)' CREATED AND OPENED
EDT0259   ELEMENT '(&00)' IN DER BIBLIOTHEK '(&01)' ANGELEGT UND GEOEFFNET
```

Bedeutung
Das Bibliothekselement wurde erzeugt und in der aktuellen Arbeitsdatei geoeffnet.

```
EDT0264   MEMBER '(&00)' IN LIBRARY '(&01)' WRITTEN AND CLOSED
EDT0264   ELEMENT '(&00)' IN BIBLIOTHEK '(&01)' GESCHRIEBEN UND GESCHLOSSEN
```

Bedeutung
Das Bibliothekselement wurde geschrieben und geschlossen.

```
EDT0265   MEMBER '(&00)' IN LIBRARY '(&01)' CLOSED UNCHANGED
EDT0265   ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' UNVERAENDERT GESCHLOSSEN
```

Bedeutung
Das Bibliothekselement wurde 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
```

Bedeutung
Das Bibliothekselement wurde in der aktuellen Arbeitsdatei geoeffnet, der alte Inhalt wurde nicht eingelesen.

```
EDT0274   MEMBER '(&00)' IN LIBRARY '(&01)' COPIED
EDT0274   ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' KOPIERT
```

Bedeutung
Das Bibliothekselement wurde in die aktuelle Arbeitsdatei 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.

```
EDT0283   UFS FILE '(&00)' DELETED
EDT0283   UFS-DATEI '(&00)' GELOESCHT
```

Bedeutung
Die POSIX-Datei wurde aus ihrem Verzeichnis entfernt.

```
EDT0285   SDF: SYNTAX TESTED. (&00) ERROR(S) IN RANGE
EDT0285   SDF: SYNTAX GEPRUEFT. (&00) FEHLER IM ZEILENBEREICH
```

Bedeutung
Bei der Bearbeitung der Anweisung @SDFTEST wurden (&00) fehlerhafte Zeilen entdeckt.
Fehlerschalter: wird nicht gesetzt.

```
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 vergleichenden Zeilen sind gleich.
Fehlerschalter: wird nicht gesetzt.

```
EDT0292   COPY BUFFER CLEARED
EDT0292   KOPIERPUFFER GELOESCHT
```

Bedeutung
Quittung nach '*' in der Markierungsspalte.

```
EDT0293   FILE NOT WRITTEN
EDT0293   DATEI NICHT GESCHRIEBEN
```

Bedeutung
Nach OVERWRITE-Abfrage wurde N eingegeben.

```
EDT0294   MAXIMUM LINE NUMBER
EDT0294   MAXIMALE ZEILENNUMMER
```

Bedeutung
Hinter der letzten belegten Zeile kann der Bildschirm nicht vollstaendig mit leeren Zeilen aufgefuellt werden, weil nicht mehr genuegend Zeilennummern zur Verfuegung stehen.

```
EDT0295   OLD COPY BUFFER CLEARED, NEW COPY BUFFER FILLED
EDT0295   ALTER KOPIERPUFFER GELOESCHT, NEUER KOPIERPUFFER GEFUELLT
```

Bedeutung
Auf eine Kurzanweisung R folgte eine Kurzanweisung C oder M.
Der durch die Kurzanweisung(en) R angelegte Kopierpuffer wurde geloescht.

```
EDT0296   OVERWRITE FILE? REPLY (Y=YES; N=NO)
EDT0296   DATEI UEBERSCHREIBEN? ANTWORT (J=JA; N=NEIN)
```

Bedeutung
Abfrage nach den Anweisungen @WRITE bzw. @SAVE, wenn diese Datei schon existiert.

Maßnahme
J: 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 (J=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.

Maßnahme
J: Die betroffenen Jobvariablen werden aus dem Katalog entfernt.
N: Die Anweisung wird abgebrochen und keine Jobvariable wird entfernt.

```
EDT0299   JOB VARIABLES NOT ERASED
EDT0299   JOBVARIABLEN NICHT ENTFERNT
```

Bedeutung
Meldung EDT0298 (ALLE JOBVARIABLEN ENTFERNEN?) wurde mit N beantwortet.

EDT0300 (&00)
EDT0300 (&00)

Bedeutung
Nach der @TMODE-Anweisung werden die Eigenschaften des Prozesses in dieser Reihenfolge von links nach rechts ausgegeben:

TSN - Prozessfolgenummer
USER-ID - Benutzerkennung des /LOGON-Kommandos
ACCOUNT- Abrechnungsnummer des Prozesses
CPU-TIME-verbrauchte CPU-Zeit des Prozesses
DATE - Datum (JJJJ-MM-TT)
TIME - Uhrzeit (HH:MM:SS)
STATEMENT-SYMBOL -aktuelles Anweisungssymbol
TERMINAL-Typ der Datensichtstation.

EDT0610 BUFFER SIZE UNCHANGED
EDT0610 PUFFERGROESSE NICHT VERAENDERT

Bedeutung
Der Puffer 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

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)' CANNOT BE SET. STANDARD WILL BE USED
EDT0651 CCS '(&00)' KANN NICHT EINGESTELLT WERDEN. STANDARD-EINSTELLUNG WIRD VERWENDET

Bedeutung
Bei der EDT-Initialisierung soll der Zeichensatz (&00) zum aktuellen Zeichensatz werden. Da dies im aktuellen Betriebsmodus nicht moeglich ist, wird stattdessen der 7-Bit-Standard EDF03IRV verwendet.

```
EDT0900   EDITED FILE(S) NOT SAVED!
EDT0900   EDITIERTE DATEI(EN) NICHT GESICHERT!
```

Bedeutung
Die EDT-Sitzung sollte mit @HALT oder einer anderen Anweisung beendet werden, doch es gibt noch ungesicherte Daten. Der EDT gibt eine Liste der Arbeitsdateien aus, in denen sich ungesicherte Daten befinden.

```
EDT0901   NO MATCH IN RANGE
EDT0901   KEIN TREFFER
```

Bedeutung
Bei der Ausfuehrung der @ON-Anweisung wurde der Suchbegriff im angegebenen Bereich nicht gefunden.
Fehlerschalter: EDT.
Tritt dieser Fehler bei der Ausfuehrung einer EDT-Prozedur (@DO oder @INPUT) auf, und ist die Protokollierung ueber den PRINT-Operanden nicht eingeschaltet, so wird diese Meldung nicht ausgegeben und der EDT-Fehlerschalter nicht gesetzt.

```
EDT0902   FILE (&00) VERSION (&01)
EDT0902   DATEI (&00) VERSION (&01)
```

Bedeutung
Moegliche Ursachen:
- Die Datei (&00) wurde bei einem Schreib-Zugriff mit der Versionsnummer '*' oder der aktuellen Versionsnummer angegeben:
  Die neue aktuelle Versionsnummer nach dem Schreiben ist (&01).
- Die Datei (&00) wurde bei einem Lese-Zugriff mit der Versionsnummer '*' oder einer falschen Versionsnummer angegeben:
  Die korrekte Versionsnummer ist (&01).

```
EDT0903   FILE '(&00)' IS IN THE CATALOG, FCBTYPE = (&01)
EDT0903   DATEI '(&00)' IST IM KATALOG, FCBTYPE = (&01)
```

Bedeutung
Die Datei (&00) ist bereits katalogisiert, ihre Zugriffsmethode ist (&01).

```
EDT0904   TERMINATE EDT? REPLY (Y=YES; N=NO)
EDT0904   EDT BEENDEN? ANTWORT (J=JA; N=NEIN)
```

Bedeutung
Dialogabfrage, ob der EDT beendet werden soll.

Maßnahme
J: Der EDT wird beendet
N: Der EDT wird nicht beendet.

```
EDT0905   EDITED MEMBER TO BE ADDED? REPLY (Y=YES; N=NO)
EDT0905   SOLL ELEMENT IN BIBLIOTHEK AUFGENOMMEN WERDEN? ANTWORT (J=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 (J=JA; N=NEIN)
```

Bedeutung
Ist fuer eine Anweisung zuwenig virtueller Speicher verfuegbar, besteht die Moeglichkeit, die Anweisung nach geeigneten Massnahmen zu wiederholen.

Maßnahme
J: Versuch wird wiederholt.
N: Anweisung wird abgebrochen.

```
EDT0907   NO WORK FILES USED
EDT0907   KEINE ARBEITSDATEIEN BELEGT
```

Bedeutung
Eine '@DROP ALL'-Anweisung wurde gegeben, aber es sind keine Arbeitsdateien belegt.

```
EDT0909   AUTOSAVE ABORTED. ERASE SAVING FILES? REPLY (Y=YES; N=NO)
EDT0909   AUTOSAVE WURDE ABGEBROCHEN. SICHERUNGSDATEIEN LOESCHEN? ANTWORT (J=JA; N=NEIN)
```

Bedeutung
Das Schreiben der Sicherungsdateien konnte nicht ausgefuehrt werden. Moegliche Ursachen dafuer sind ein Speicherengpass oder unvorhergesehene DVS-Fehler. Die automatische Sicherung wird ausgeschaltet.

Maßnahme
J: Die vorhandenen Sicherungsdateien werden geloescht.
N: Die vorhandenen Sicherungsdateien bleiben erhalten.

```
EDT0910   '@RENUMBER': LINES WILL BE LOST
EDT0910   '@RENUMBER': ZEILEN WERDEN VERLOREN GEHEN
```

Bedeutung
Eine @RENUMBER-Anweisung wurde eingegeben, um die Zeilen umzunummerieren. Wenn der EDT in der angegebenen Weise umnummeriert, wird die groesst-moegliche Zeilennummer (9999.9999) erreicht und der Rest der Datei ginge verloren.

Maßnahme
Bevor neu nummeriert wird, kann mit 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 (J=JA; N=NEIN)
```

Bedeutung
Bei der Abarbeitung einer Anweisung wurde ein Fehler entdeckt.
Der EDT fragt nach, ob die Anweisung noch weiter bearbeitet
werden soll.

Maßnahme
J: Die Bearbeitung wird fortgesetzt.
N: Die Anweisung wird abgebrochen.

```
EDT0912   INTERRUPTION NOT POSSIBLE
EDT0912   UNTERBRECHUNG NICHT MOEGLICH
```

Bedeutung
Waehrend einer nicht-unterbrechbaren Prozedur wurde die K2-Taste
gedrueckt. Der Wechsel in den Systemmodus ist momentan nicht moeglich.

```
EDT0913   /INFORM-PROG TO BE SIMULATED? REPLY (Y=YES; N=NO)
EDT0913   /INFORM-PROG SIMULIEREN? ANTWORT (J=JA; N=NEIN)
```

Bedeutung
In einer nicht-unterbrechbaren Prozedur kann das Kommando
/INFORM-PROGRAM nicht eingegeben werden.
Der EDT fragt nach, ob Aktionen durchgefuehrt werden sollen.

Maßnahme
J: Der EDT verhaelt sich, als ob /INFORM-PROGRAM eingegeben worden
waere.
N: Der Programmablauf wird fortgesetzt.

```
EDT0914   RECORD SIZE > 256. ONLY 256 CHARACTERS WILL BE WRITTEN
EDT0914   SATZLAENGE > 256. ES WERDEN NUR 256 ZEICHEN GESCHRIEBEN
```

Bedeutung
In jedem Satz werden nur 256 Zeichen geschrieben, der Rest der Saetze
wird mit undefiniertem Inhalt ueberschrieben.

```
EDT0915   CONVERT TO FILE CCS (&OO)? REPLY (Y=YES; N=NO)
EDT0915   IN DATEI-CCS (&OO) UMWANDELN? ANTWORT (J=JA; N=NEIN)
```

Bedeutung
Der Zeichensatz der Arbeitsdatei ist verschieden von dem der Datei,
in die geschrieben werden soll.

Maßnahme
J: Vor dem Schreiben wird in den Zeichensatz der Datei konvertiert.
N: Beim Schreiben wird der Zeichensatz der Arbeitsdatei verwendet.

```
EDT0990   (&00)
EDT0990   (&00)
```

Bedeutung
(&00): Meldung von Test-Routine.

```
EDT0999   (&00)
EDT0999   (&00)
```

Bedeutung
(&00): Meldung von externer Routine.

```
EDT1137   SPECIFIED WORK FILE IGNORED IN CONJUNCTION WITH 'SPLIT'
EDT1137   ARBEITSDATEI-ANGABE IN VERBINDUNG MIT 'SPLIT' IGNORIERT
```

Bedeutung
In der @PAR-Anweisung wurde eine Arbeitsdatei 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
```

Bedeutung
Mit der XWRITE-Anweisung wird die aktuelle Arbeitsdatei in die vorher mit XOPEN geoeffnete UFS-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.

```
EDT1181   CODE OPERAND IGNORED
EDT1181   CODE-OPERAND IGNORIERT
```

Bedeutung
Eine POSIX-Datei oder eine nicht existierende DVS-Datei bzw. ein nicht existierendes Bibliothekselement soll mit @WRITE CODE=*FILE geschrieben werden. Fuer POSIX-Dateien wird der mit @PAR CODE eingestellte Zeichensatz verwendet, ansonsten der Zeichensatz der Arbeitsdatei.

```
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
```

Bedeutung
Die Datei wurde erzeugt oder zurueckgeschrieben, aber das CCS-Attribut kann nicht vergeben werden, da der Zugriff zum Datei- bzw. Bibliothekskatalog einen Fehler liefert.
Fehlerschalter: EDT, DVS.

```
EDT1243   FILE '(&00)' TO BE COPIED IS EMPTY
EDT1243   KOPIERDATEI '(&00)' LEER
```

Bedeutung
Die DVS-Datei, die in die aktuelle Arbeitsdatei kopiert werden soll, ist leer.
Fehlerschalter: EDT.

```
EDT1244   FILE '(&00)' EMPTY
EDT1244   DATEI '(&00)' LEER
```

```
EDT1245   JOB VARIABLE IS EMPTY
EDT1245   JOBVARIABLE LEER
```

Bedeutung
Es wurde versucht, mit @GETJV den Wert einer Jobvariablen zu lesen, aber die Jobvariable ist leer.
Fehlerschalter: EDT.

```
EDT1246   UFS FILE '(&00)' TO BE COPIED IS EMPTY
EDT1246   ZU KOPIERENDE UFS-DATEI '(&00)' IST LEER
```

Bedeutung
Die POSIX-Datei, die in die aktuelle Arbeitsdatei kopiert werden soll, ist leer.
Fehlerschalter: EDT.

```
EDT1247   MEMBER '(&00)' IN LIBRARY '(&01)' TO BE COPIED IS EMPTY
EDT1247   ZU KOPIERENDES ELEMENT '(&00)' DER BIBLIOTHEK '(&01)' IST LEER
```

Bedeutung
Das Bibliothekselement, das in die aktuelle Arbeitsdatei kopiert werden soll, ist leer.
Fehlerschalter: EDT.

```
EDT1248   EMPHASIS INCOMPLETE IN SOME LINES
EDT1248   OPTISCHE HERVORHEBUNG IN EINIGEN ZEILEN UNVOLLSTAENDIG
```

Bedeutung
Nach einer @ON-Anweisung werden bei der Anzeige am Bildschirm nicht alle Treffer optisch hervorgehoben, da durch das Einfuegen der Bildschirmsteuerzeichen eine oder mehrere Zeilen zu lang werden.
Fehlerschalter: EDT.

```
EDT1253   (SOME) RECORD(S) TRUNCATED
EDT1253   (EINIGE) SAETZE ABGESCHNITTEN
```

Bedeutung
Beim Einlesen in die interne Arbeitsdatei werden zu lange Saetze abgeschnitten (im Unicode-Modus nach 32768, sonst nach 256 Zeichen).
Fehlerschalter: EDT.

```
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 (Anweisung @OPEN), 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 wurde fuer eine SAM-Datei eingegeben. Der EDT fuehrt automatisch eine @READ-Anweisung aus.
Fehlerschalter: EDT.

```
EDT1903    INPUT TRUNCATED
EDT1903    EINGABE ABGESCHNITTEN
```

Bedeutung
Im Unicode-Modus wird bei der Tabulator-Expansion die maximale Zeilenlaenge (32768 Zeichen) ueberschritten. Die Zeile wird abgeschnitten.
Im Kompatibilitaetsmodus werden bei der Eingabe von Datenzeilen im Line-Modus oder beim Einlesen des Elements einer Listenvariablen (Anweisung @GETLIST) mehr als 256 Zeichen gelesen. Die Eingabe wird abgeschnitten.
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 nicht geladen.
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.

```
EDT1937    MODIFIED LINE > 32768 CHARACTERS
EDT1937    MODIFIZIERTE ZEILE > 32768 ZEICHEN
```

Bedeutung
Eine Arbeitsdateizeile wuerde bei der Bearbeitung mit der Anweisung @ON zu lang werden. Die Zeile wird nicht veraendert.
Fehlerschalter: EDT.

```
EDT1938    MODIFIED LINE > 32768 CHARACTERS
EDT1938    MODIFIZIERTE ZEILE > 32768 ZEICHEN
```

Bedeutung
Eine Zeile in einer @DO-Prozedur wuerde bei der Parameter-Ersetzung zu lang werden. Die Zeile wird abgeschnitten.
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 im F-Modus-Datenfenster eingegebene Satz ueberschreitet die in der @PAR-Anweisung als LIMIT angegebene Laenge und wird abgeschnitten.
(&00): max. zulaessige Satzlaenge.

```
EDT2301   COPY BUFFER OVERFLOW
EDT2301   KOPIERPUFFER VOLL
```

Bedeutung
Der Kopierpuffer kann nur 256 Zeilennummern aufnehmen.

```
EDT2400   LINE TRUNCATED AFTER 32768 CHARACTERS
EDT2400   ZEILE NACH 32768 ZEICHEN ABGESCHNITTEN
```

Bedeutung
Bei der Bearbeitung mit den Anweisungen @CREATE oder @SETJV, mit der Kurzanweisung J oder bei der Dateneingabe mit Tabulatorexpansion wurde eine Arbeitsdateizeile bzw. Zeichenfolgevariable erzeugt, die mehr als 32768 Zeichen hat. Es wird nach 32768 Zeichen abgeschnitten.
Fehlerschalter: EDT.

```
EDT2401   LINE TRUNCATED AFTER 2048 BYTES
EDT2401   AUSGABE NACH 2048 BYTES ABGESCHNITTEN
```

Bedeutung
Die Zeichenfolge, die bei der Anweisung @CREATE (Format 3 oder 4) als Eingabeaufforderung ausgegeben werden soll, ist zu lang. Sie wird nach 2048 Bytes abgeschnitten.
Fehlerschalter: EDT.

EDT2402 OUTPUT TRUNCATED AFTER (&00) CHARACTERS
EDT2402 AUSGABE NACH (&00) ZEICHEN ABGESCHNITTEN

Bedeutung
Die Zeichenfolge, die bei der Anweisung @CREATE (Format 3 oder 4) als Eingabeaufforderung ausgegeben werden soll, ist zu lang. Sie wird nach (&00) Zeichen abgeschnitten.
Fehlerschalter: EDT.

Maßnahme
Zeichenfolge kuerzen.

EDT2403 OUTPUT TRUNCATED AFTER 4096 BYTES
EDT2403 AUSGABE NACH 4096 BYTES ABGESCHNITTEN

Bedeutung
Die Zeichenfolge, die einer S-Variablen zugewiesen werden soll, ist zu lang. Sie wird nach 4096 Bytes abgeschnitten.
Fehlerschalter: EDT.

EDT2404 NOT ENOUGH LINES FOR HEX MODE
EDT2404 ZUWENIGE ZEILEN FUER HEX-DARSTELLUNG VORHANDEN

Bedeutung
Auf einem geteilten Bildschirm stehen weniger Zeilen zur Verfuegung, als fuer die Darstellung einer Datenzeile im Hexadezimalmodus benoetigt werden.
Fehlerschalter: EDT.

Maßnahme
Arbeitsfenster vergroessern (@PAR SPLIT).

EDT2405 LINE TRUNCATED DURING AUTOSAVE
EDT2405 ZEILE BEI AUTOSAVE ABGESCHNITTEN

Bedeutung
Beim Sichern der Arbeitsdatei mit @AUTOSAVE ist eine Zeile laenger als die maximal zulaessige Satzlaenge der Sicherungsdatei. Die Zeile wird abgeschnitten.
Fehlerschalter: EDT.

EDT2406 MAXIMUM LINE NUMBER
EDT2406 MAXIMALE ZEILENNUMMER

Bedeutung
Die betroffene Anweisung konnte ausgefuehrt werden, aber die neue aktuelle Zeile konnte nicht wie ueblich als letzte Zeile der Arbeitsdatei plus aktuelle Schrittweite festgelegt werden, weil dabei die maximale Zeilennummer (9999.9999) ueberschritten wuerde. Als neue aktuelle Zeile wurde die letzte Zeile der Arbeitsdatei eingestellt.
Fehlerschalter: EDT.

EDT2407 OUTPUT OF SYSTEM COMMAND TRUNCATED
EDT2407 AUSGABE DES SYSTEMKOMMANDOS ABGESCHNITTEN

Bedeutung
Die Ausgabe des mit @SYSTEM abgesetzten Betriebssystem-Kommandos ist zu lang, sie wird abgeschnitten.
Fehlerschalter: EDT.

EDT2408 LOGGING TERMINATED
EDT2408 PROTOKOLL-AUSGABE ABGEBROCHEN

Bedeutung
In das mit @LOG eingestellte Ausgabemedium kann nicht geschrieben werden, die Protokollierung wird ausgeschaltet.
Fehlerschalter: EDT.

EDT2409 CPU TIME SPECIFIED AT EDT START EXCEEDED
EDT2409 BEIM EDT-START ANGEGEBENE CPU-ZEIT UEBERSCHRITTEN

Bedeutung
Die beim Start des EDT angegebene maximale CPU-Laufzeit wurde ueberschritten. Der EDT-Lauf wird fortgesetzt.
Fehlerschalter: EDT.

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. In der Arbeitsdatei wird dafuer die Zeilennummer 0.0001 verwendet.
Fehlerschalter: EDT.

```
EDT2901   CHECK LINE LENGTH
EDT2901   ZEILENLAENGE PRUEFEN
```

Bedeutung
Die Zeilenlaengenpruefung ist eingeschaltet (@CHECK oder @TABS) und es wurde eine Zeile eingegeben oder durch Tabulatorexpansion erzeugt, die laenger ist als die fuer die Pruefung angegebene Zeichenanzahl. Die Zeile wird wie angegeben angelegt.
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 Tabulatorzeichen 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
Moegliche Fehlerursache:
- Auf eine leere Datei auf der Platte wird mit einer @READ-, @GET-, @INPUT- oder @ELIM-Anweisung zugegriffen.
- Eine @SETLIST-Anweisung wird fuer eine leere Arbeitsdatei gegeben.
- Eine bei @COMPARE angegebene Arbeitsdatei 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 groesstmoegliche Zeilennummer (9999.9999) wurde bei der Ausfuehrung 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
Die Anweisung enthaelt einen syntaktischen Fehler.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3003 '(' MISSING
EDT3003 '(' FEHLT

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Fehlende Klammer einfuegen und Anweisung wiederholen.

EDT3004 ')' MISSING
EDT3004 ')' FEHLT

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Fehlende Klammer einfuegen und Anweisung wiederholen.

EDT3040 NAME INVALID OR MISSING
EDT3040 NAME UNGUELTIG ODER NICHT VORHANDEN

Bedeutung
Die Zeichenfolge ist laenger als 8 Zeichen, entspricht nicht den syntaktischen Anforderungen des Operanden oder fehlt ganz.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3050 INVALID SYSLST-NUMMER
EDT3050 UNGUELTIGE SYSLST-NUMMER

Bedeutung
Die in der @LOG-Anweisung angegebene SYSLST-Nummer ist nicht gueltig. Sie muss zwischen 1 und 99 liegen.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

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.

Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3066 WRONG COLUMN NUMBER
EDT3066 FALSCHE SPALTENANGABE

Bedeutung
Die Spaltenangabe in der angegebenen @SETF-Anweisung ist falsch.
Fehlerschalter: EDT.

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
Die Angabe des Operanden der @PAR-Anweisung ist fehlerhaft, nach dem Gleichheitszeichen fehlt ein ON, OFF oder O.
Fehlerschalter: EDT.

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.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3073   TARGET POSITION IS INVALID OR MISSING
EDT3073   ZIELPOSITION FEHLT ODER IST NICHT RICHTIG
```

Bedeutung
Die Anweisung wurde nicht ausgefuehrt, da in der COPY-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.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3078   SPECIFIED NUMBER INVALID (VALID RANGE: 1..256)
EDT3078   FEHLERHAFTER ZAHLENWERT (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.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
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.
Fehlerschalter: EDT.

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.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3082   LINE NUMBER 0 INVALID
EDT3082   ZEILENNUMMER 0 UNZULAESSIG
```

Bedeutung
Die Zeilennummer 0 und das Inkrement 0 sind unzulaessig.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3085   '(&00)' NOT POSSIBLE FOR PLAM ELEMENT TYPE '(&01)'
EDT3085   '(&00)' NICHT MOEGLICH FUER PLAM ELEMENT TYP '(&01)'
```

Bedeutung
Die Element-Typen R, C, H, L, U, F oder daraus abgeleitete freie Typen koennen nicht mit den Anweisungen @COPY, @OPEN, @WRITE oder @INPUT bearbeitet werden.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

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 JOBVARIABLEN-NAME

Bedeutung
Die Zeichenfolge, die als Jobvariablen-Name angegeben wurde,
entspricht nicht der Syntax fuer einen Jobvariablen-Namen, oder
der Jobvariablen-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

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

Bedeutung
Die Zeichenfolge, die als UFS-Dateiname 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.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3093   INVALID STRUCTURED NAME OR STRUCTURED NAME MISSING
EDT3093   STRUCTURED NAME UNGUELTIG ODER NICHT VORHANDEN
```

Bedeutung
Die Zeichenfolge ist laenger als 30 Zeichen, entspricht nicht den syntaktischen Anforderungen des Operanden oder fehlt ganz.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3100   WORK FILE IS EMPTY. STATEMENT NOT PROCESSED
EDT3100   ARBEITSDATEI LEER. ANWEISUNG NICHT AUSGEFUEHRT
```

Bedeutung
Die Anweisung bezieht sich auf eine Zeilennummer, die nicht gefunden werden kann, da die Arbeitsdatei leer ist.

```
EDT3101   INVALID STATEMENT
EDT3101   UNZULAESSIGE ANWEISUNG
```

Bedeutung
Es wurde keine gueltige EDT-Anweisung erkannt.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3106   SPECIFIED WORK FILE INVALID (VALID RANGE: 0..9)
EDT3106   FALSCHE ANGABE DER ARBEITSDATEI. (ZULAESSIGER BEREICH: 0..9)
```

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3110   EQUAL SIGN EXPECTED. STATEMENT NOT PROCESSED
EDT3110   GLEICHHEITSZEICHEN ERWARTET. ANWEISUNG NICHT AUSGEFUEHRT
```

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3111   'TYPE' OPERAND IS MISSING OR INVALID
EDT3111   'TYPE'-OPERAND FEHLT ODER IST UNGUELTIG
```

Bedeutung
Der Operand TYPE wurde ohne gueltigen Operandenwert angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3112   'TYPE' OPERAND ALREADY DEFINED
EDT3112   'TYPE'-OPERAND BEREITS DEFINIERT
```

Bedeutung
Der Operand TYPE wurde mehrfach angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3116   'CODE' OPERAND MISSING OR INVALID
EDT3116   'CODE'-OPERAND FEHLT ODER IST UNGUELTIG
```

Bedeutung
Der Operand CODE wurde ohne gueltigen Operandenwert angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3117   'MODE' OPERAND MISSING OR INVALID
EDT3117   'MODE'-OPERAND FEHLT ODER IST UNGUELTIG
```

Bedeutung
Der Operand MODE wurde ohne gueltigen Operandenwert angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3118   'MODE' OPERAND ALREADY DEFINED
EDT3118   'MODE'-OPERAND BEREITS DEFINIERT
```

Bedeutung
Der Operand MODE wurde mehrfach angegeben.
Fehlerschalter: EDT.

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

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3122 FILE NAME MISSING OR FORMAT OF SPECIFIED FILE NAME INVALID
EDT3122 DATEINAME FEHLT ODER FORMAT DER ANGABE IST FALSCH

Bedeutung
Fehlerschalter: EDT.

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.
Fehlerschalter: EDT.

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.
Fehlerschalter: EDT.

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
Fuer die Arbeitsdatei wurde kein Name angegeben oder der angegebene Name ist unzulaessig.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3128   PLAM LIBRARY NAME INVALID OR MISSING
EDT3128   PLAM-BIBLIOTHEKSNAME FEHLT ODER IST UNGUELTIG
```

Bedeutung
Fehlerschalter: EDT.

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
Die TYPE-Bezeichnung des PLAM-Elementes wurde in der Anweisung nicht oder nicht richtig angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3133 PLAM VERSION SPECIFICATION INVALID
EDT3133 PLAM-VERSIONSBEZEICHNUNG IST UNGUELTIG

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Anweisung mit gueltiger Versionsbezeichnung 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
Fuer die Definition eines Symbols wurde mehr als ein Zeichen angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3170 SYNTAX ERROR IN LINE NUMBER
EDT3170 SYNTAX-FEHLER IN ZEILENNUMMER

Bedeutung
Der Operand entspricht nicht der Syntax fuer eine Zeilennummer.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3172 MODULE NAME TOO LONG
EDT3172 MODULNAME IST ZU LANG

Bedeutung
Der in der @UNLOAD-Anweisung angegebene Modulname ist laenger als
8 Zeichen (kompatibles Format) bzw. laenger als 32 Zeichen (neues Format).
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3173 NUMBER OF WORK FILE FOR COMPARE OPERATION MISSING OR INVALID
EDT3173 NUMMER DER ARBEITSDATEI FUER VERGLEICH FEHLT ODER IST UNGUELTIG

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3174 NAME TOO LONG
EDT3174 NAME ZU LANG

Bedeutung
Die Zeichenfolge, die zur Angabe eines Datei- oder Jobvariablen-Namens
verwendet wurde, ist laenger als vom System erlaubt
(voll qualifiziert: 54, mit Wildcards: 80).
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3175 SYNTAX ERROR IN SPECIFIED RANGE
EDT3175 SYNTAX-FEHLER IN ZEILENBEREICHSANGABE

Bedeutung
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

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 (kompatibles Format) bzw. 32 Zeichen (neues Format) lang sein.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3178 LIBRARY NAME TOO LONG
EDT3178 BIBLIOTHEKSNAME ZU LANG

Bedeutung
Der Bibliotheksname darf maximal 54 Zeichen lang sein.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3179 ENTRY NAME MISSING OR INVALID
EDT3179 ENTRY-NAME FEHLT ODER IST UNGUELTIG

Bedeutung
Die Zeichenfolge entspricht nicht den syntaktischen Anforderungen fuer Entry-Namen oder es ist kein Entry-Name angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

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 eingeben.

EDT3183 LINE NUMBER EXPECTED
EDT3183 ZEILENNUMMER ERWARTET

Bedeutung
Nach dem Schluesselwort TO in den Anweisungen @SHOW, @FSTAT, @STATUS, @SYSTEM oder @STAJV muss eine gueltige Zeilennummer angegeben werden.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3190 SPECIFIED NUMBER INVALID (VALID RANGE: 1..32768)
EDT3190 FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 1..32768)

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3191 OPERAND TOO LONG
EDT3191 OPERAND ZU LANG

Bedeutung
Der angegebene Operand ist laenger als erlaubt.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3192 ILLEGAL CHARACTER IN OPERAND
EDT3192 OPERAND ENTHAELT UNGUELTIGE ZEICHEN

Bedeutung
Der angegebene Operand enthaelt nicht erlaubte Zeichen.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3193 SPECIFIED WORK FILE INVALID (VALID RANGE: 0..22)
EDT3193 FALSCHE ANGABE DER ARBEITSDATEI (ZULAESSIGER BEREICH: 0..22)

Bedeutung
Die angegebene Arbeitsdatei-Nummer liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3194   SPECIFIED NUMBER INVALID (VALID RANGE: 1..9)
EDT3194   FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 1..9)
```

Bedeutung
Die Wert fuer die Satzmarkierung liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3195   SPECIFIED UNICODE VALUE INVALID
EDT3195   UNZULAESSIGER UNICODE-WERT
```

Bedeutung
Die direkte Angabe eines Separatorzeichens, eines Struktursymbols, eines Ersatzzeichens oder eines Fuellzeichens als Unicode-Wert entspricht keinem gueltigen Zeichen.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3196   SPECIFIED HEX OR BINARY VALUE INVALID
EDT3196   UNZULAESSIGER HEX-/BINAERWERT
```

Bedeutung
Der angegebene Hex- oder Binaerwert entspricht keinem gueltigen Zeichen.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3197   SPECIFIED NUMBER INVALID (VALID RANGE: 0..32768)
EDT3197   FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 0..32768)
```

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3198   SPECIFIED NUMBER INVALID (VALID RANGE: 0..99999999)
EDT3198   FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 0..99999999)
```

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3199 SPECIFIED NUMBER INVALID (VALID RANGE: 0..31)
EDT3199 FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 0..31)

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3200 SPECIFIED NUMBER INVALID (VALID RANGE: 1..8)
EDT3200 FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 1..8)

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3201 SPECIFIED NUMBER INVALID (VALID RANGE: 0..65535)
EDT3201 FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 0..65535)

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3202 SPECIFIED NUMBER INVALID (VALID RANGE: 1..65535)
EDT3202 FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 1..65535)

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3203 'KEY' OPERAND ALREADY DEFINED
EDT3203 'KEY'-OPERAND BEREITS DEFINIERT

Bedeutung
Der Operand KEY wurde mehrfach angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3204   'CODE' OPERAND ALREADY DEFINED
EDT3204   'CODE'-OPERAND BEREITS DEFINIERT
```

Bedeutung
Der Operand CODE wurde mehrfach angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3205   'KEY' OPERAND IS MISSING OR INVALID
EDT3205   'KEY'-OPERAND FEHLT ODER IST UNGUELTIG
```

Bedeutung
Der Operand KEY wurde ohne gueltigen Operandenwert angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3206   SPECIFIED NUMBER INVALID (VALID RANGE: 0..255)
EDT3206   FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 0..255)
```

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

```
EDT3207   SPECIFIED NUMBER INVALID (VALID RANGE: 0..256)
EDT3207   FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 0..256)
```

Bedeutung
Der angegebene Zahlenwert liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT3209   SPECIFIED WORK FILE INVALID (VALID RANGE: 1..22)
EDT3209   FALSCHE ANGABE DER ARBEITSDATEI (ZULAESSIGER BEREICH: 1..22)
```

Bedeutung
Die angegebene Arbeitsdatei-Nummer liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3210 'LENGTH' OPERAND IS MISSING OR INVALID
EDT3210 'LENGTH'-OPERAND FEHLT ODER IST UNGUELTIG

Bedeutung
Der Operand LENGTH wurde ohne gueltigen Operandenwert angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3211 'LENGTH' OPERAND ALREADY DEFINED
EDT3211 'LENGTH'-OPERAND BEREITS DEFINIERT

Bedeutung
Der Operand LENGTH wurde mehrfach angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3212 'MARK' OPERAND IS MISSING OR INVALID
EDT3212 'MARK'-OPERAND FEHLT ODER IST UNGUELTIG

Bedeutung
Der Operand MARK wurde ohne gueltigen Operandenwert angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3213 'MARK' OPERAND ALREADY DEFINED
EDT3213 'MARK'-OPERAND BEREITS DEFINIERT

Bedeutung
Der Operand MARK wurde mehrfach angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3214 LINK NAME IS MISSING OR INVALID
EDT3214 DATEIKETTUNGSNAME FEHLT ODER IST UNGUELTIG

Bedeutung
Der angegebene Dateikettungsname hat keinen gueltigen Wert.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3215 MISSING CLOSING QUOTE IN STRING
EDT3215 BEGRENZERSYMBOL AM ENDE DER ZEICHENFOLGE FEHLT

Bedeutung
Eine Zeichenfolge wurde ohne schliessendes Begrenzersymbol angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3216 INVALID RANGE (LOWER > UPPER)
EDT3216 UNGUELTIGE BEREICHSANGABE (UNTERE GRENZE > OBERE GRENZE)

Bedeutung
Die erste in der @SETSW-Anweisung angegebene Schalternummer ist groesser als die zweite.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3217 TOO MANY DIGITS IN NUMBER
EDT3217 ZAHL ENTHAELT ZU VIELE ZIFFERN

Bedeutung
Die Zahl fuer die Nummerierung mit der @SEQUENCE-Anweisung darf maximal 8 Stellen haben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3218 'TO' OPERAND EXPECTED
EDT3218 'TO'-OPERAND ERWARTET

Bedeutung
In der Anweisung @COPY oder @MOVE fehlt ein TO-Operand.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3219 'TYPE' OPERAND NOT ALLOWED
EDT3219 'TYPE'-OPERAND NICHT ERLAUBT

Bedeutung
Der Operand TYPE ist nur fuer DVS-Dateien erlaubt.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3220 'KEY' OPERAND NOT ALLOWED
EDT3220 'KEY'-OPERAND NICHT ERLAUBT

Bedeutung
Der Operand KEY ist nur fuer DVS-Dateien erlaubt.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3221 'CODE' OPERAND NOT ALLOWED
EDT3221 'CODE'-OPERAND NICHT ERLAUBT

Bedeutung
Der Operand CODE ist nur fuer POSIX-Dateien erlaubt.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3222 INCREMENT MISSING OR INVALID
EDT3222 INKREMENT FEHLT ODER IST FALSCH

Bedeutung
Die Inkrement-Angabe fehlt oder der Wert des Inkrements ist ungueltig.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3223 UFS FILE NAME MISSING OR INVALID
EDT3223 UFS-DATEINAME FEHLT ODER IST UNGUELTIG

Bedeutung
Die Angabe des UFS-Dateinamens fehlt, oder sie entspricht nicht der Syntax fuer einen gueltigen Dateinamen im POSIX-Dateisystem.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3224 SPECIFIED NUMBER INVALID (VALID RANGE: 1..2048)
EDT3224 FEHLERHAFTER ZAHLENWERT (ZULAESSIGER BEREICH: 1..2048)

Bedeutung
Der Wiederholungsfaktor n der #-Anweisung liegt nicht im zulaessigen Wertebereich.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3901 ILLEGAL BINARY CONSTANT
EDT3901 UNZULAESSIGE BINAER-KONSTANTE

Bedeutung
Eine Zeichenfolge mit einem ’B’ vor dem ersten Hochkomma oder die Texteingabe nach @INPUT BINARY enthaelt einen Fehler.
Die Zeichen muessen die Ziffern 0 oder 1 sein, und
die Zeichenfolge darf nicht leer sein.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung oder Daten eingeben.

EDT3902 ILLEGAL HEX CONSTANT
EDT3902 UNGUELTIGE HEX-KONSTANTE

Bedeutung
Eine Zeichenfolge mit einem ’X’ vor dem 1. Hochkomma oder die Texteingabe nach @INPUT HEX 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.

Maßnahme
Korrigierte Anweisung oder Daten eingeben.

EDT3903 INVALID RANGE
EDT3903 UNGUELTIGER BEREICH

Bedeutung
Die Zeilennummern in einer Bereichsangabe sind fehlerhaft, oder nach einem Bindestrich (-) folgt keine zweite Zeilennummer.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3904 INVALID SUBSTRING
EDT3904 UNGUELTIGE TEILZEICHENFOLGE

Bedeutung
Eine Teilzeichenfolge in einer @SET-Anweisung ist fehlerhaft.
Sie sollte dem Syntax In oder +/-int entsprechen.
Fehlerschalter: EDT.

EDT3905 INVALID VARIABLE
EDT3905 UNGUELTIGE VARIABLE

Bedeutung
Eine Zeilennummer-, Zeichenfolge- oder Ganzzahlvariable wurde fehlerhaft angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3906 LINE NUMBER INVALID
EDT3906 UNGUELTIGE ZEILENNUMMER

Bedeutung
Moegliche Ursachen:
- Der Inhalt der Ganzzahl-Variablen in der Anweisung @SET (Format 3) stellt keinen gueltigen Zeilennummern-Wert dar.
- Der Inhalt der Ziel-Zeilennummern-Variablen in der Anweisung @SET (Format 4 oder 5) stellt keinen gueltigen Zeilennummern-Wert dar.
- Die Zielangabe in der Anweisung @GETJV stellt keinen gueltigen Zeilennummern-Wert dar.
- Die Zeilennummer ist fuer die KEYLEN einer durch @OPEN real geoeffneten Datei zu gross.

Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3907 EMPTY STRING NOT PERMISSIBLE
EDT3907 LEERE ZEICHENFOLGE NICHT ERLAUBT

Bedeutung
Die Zeichenfolge, die in einer Anweisung direkt oder indirekt (z.B. in einer EDT-Zeichenfolgevariablen oder in einer SDF-P-Variablen) angegeben wurde, ist leer. Dies ist an dieser Stelle nicht erlaubt.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3908 STRING MISSING OR INVALID
EDT3908 ZEICHENFOLGE FEHLT ODER IST UNGUELTIG

Bedeutung
Eine Zeichenfolge fehlt in einer Anweisung oder ist ungueltig.
Im Kompatibilitaetsmodus kann die Fehlerursache auch sein, dass in einer Ein-/Ausgabe-Anweisung kein Dateiname angegeben ist und kein lokaler und kein globaler @FILE-Eintrag definiert ist.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

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

Bedeutung
In der @PARAMS-Anweisung wurde ein formaler Operand mindestens zweimal angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3911 DUPLICATE KEYWORD
EDT3911 SCHLUESSELWORT MEHRFACH ANGEGEBEN

Bedeutung
In einer @DO-Anweisung wurde ein Schluesselwort mindestens zweimal angegeben.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3922 INVALID COLUMN (RANGE)
EDT3922 SPALTEN-(BEREICH) UNGUELTIG

Bedeutung
Der fuer eine Spalte angegebene Wert ist ungueltig, oder die Angabe der Spalte bzw. des Spaltenbereichs ist syntaktisch fehlerhaft.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3951 PROCEDURE NUMBER > 22
EDT3951 PROZEDUR-NUMMER > 22

Bedeutung
Fehlerschalter: EDT.

EDT3952 INVALID SYMBOL
EDT3952 UNGUELTIGES SYMBOL

Bedeutung
Fuer die Definition des Symbols muss ein Sonderzeichen angegeben werden.
Fehlerschalter: EDT.

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 ohne weitere Informationen.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

EDT3999 (&00)
EDT3999 (&00)

Bedeutung
Die externe Routine meldet einen Syntaxfehler in der angegebenen Anweisung mit der Beschreibung (&00).
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung eingeben.

```
EDT4200   MACRO '(&00)': DMS ERROR CODE: '(&01)'
EDT4200   MAKRO '(&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   MACRO '(&00)': JVS ERROR CODE: '(&01)'
EDT4201   MAKRO '(&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   MACRO '(&00)': SDF-P ERROR CODE: '(&01)'
EDT4202   MAKRO '(&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   MACRO '(&00)': XHCS ERROR CODE: '(&01)'
EDT4203   MAKRO '(&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 dem BS2000-Handbuch "XHCS" entnommen werden.
Die Ausfuehrung einer @INPUT-Datei wird durch diesen Fehler abgebrochen.
Fehlerschalter: DVS.

```
EDT4204   MACRO '(&00)': TIAM ERROR CODE: '(&01)'
EDT4204   MAKRO '(&00)': TIAM-FEHLERCODE: '(&01)'
```

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   MACRO '(&00)': BLS ERROR CODE: '(&01)'
EDT4205   MAKRO '(&00)': BLS-FEHLER-CODE: '(&01)'
```

Bedeutung

Alle Fehler des Binder-Lader-Systems werden in dieser Form ausgedruckt wobei gilt:
(&00): BLS-Makro (BIND), bei dessen Ausfuehrung der Fehler auftritt.
(&01): Fehlercode in hexadezimaler Form.
Naehere Information ueber den BLS-Fehler kann dem BS2000-Handbuch 'Binder und Lader' bzw. dem BS2000-Handbuch 'Makroaufrufe 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)'
```

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   MACRO '(&00)': SDF ERROR CODE: '(&01)'
EDT4207   MAKRO '(&00)': SDF-FEHLERCODE: '(&01)'
```

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.

```
EDT4208   MACRO '(&00)' RETURNS ERROR CODE '(&01)'
EDT4208   MAKRO '(&00)' LIEFERT FEHLERCODE '(&01)'
```

Bedeutung
Bei der Ausfuehrung des Makros (&00) ist der Fehler mit dem Fehlercode (&01) aufgetreten. Der Fehlercode wird als Meldungsnummer ausgegeben, falls eine solche zur Verfuegung steht. Falls nicht, wird der vollstaendige Makro-Rueckkehrcode (Subcode2, Subcode1, Maincode) ausgegeben.
Stehen zwei Rueckkehrcodes zur Verfuegung, werden beide, durch / getrennt, ausgegeben.
Naehere Information kann dem entsprechenden BS2000-Handbuch entnommen werden.
Fehlerschalter: EDT, DVS.

```
EDT4209   FUNCTION '(&00)' RETURNS ERROR '(&01)'
EDT4209   FUNKTION '(&00)' LIEFERT FEHLER '(&01)'
```

Bedeutung
Bei der Ausfuehrung der Funktion (&00) wurde der Fehler (&01) in der C-Variablen ERRNO gemeldet.
Naehere Information kann dem entsprechenden BS2000-Handbuch entnommen werden.
Fehlerschalter: EDT, DVS.

```
EDT4300   ERROR AT SYSTEM COMMAND: ERROR CODE '(&00)'
EDT4300   FEHLER BEI SYSTEMKOMMANDO: FEHLERCODE '(&00)'
```

Bedeutung
Bei Aufruf eines Systemkommandos mittels @SYSTEM-Anweisung lieferte der CMD-Makro einen Fehler.
Naehere Information ueber die Fehlerursache kann mit dem 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)
```

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.

```
EDT4312   CHECK TAB COLUMNS IN LINE (&00)
EDT4312   TABULATORSPALTEN IN ZEILE (&00) PRUEFEN
```

Bedeutung
Da die CHECK-Funktion fuer @TABS eingeschaltet ist, wurde bemerkt, dass in der angegebenen Zeile eine Rueckwaertstabellierung vorgenommen wurde, d.h. dass der angegebene Text ueberschrieben wurde.
Die Bearbeitung der Anweisung @TABS RANGE wird abgebrochen.
Fehlerschalter: EDT.

```
EDT4313   LINE (&OO) > 256 CHARACTERS
EDT4313   ZEILE (&OO) > 256 ZEICHEN
```

Bedeutung
Die Laenge der angegebenen Zeile wuerde 256 Zeichen ueberschreiten.
Die Abarbeitung der Anweisung @TABS RANGE wird abgebrochen.
Fehlerschalter: EDT.

```
EDT4314   LINE (&OO) > 32768 CHARACTERS
EDT4314   ZEILE (&OO) > 32768 ZEICHEN
```

Bedeutung
Die Laenge der angegebenen Zeile wuerde 32768 Zeichen ueberschreiten.
Die Bearbeitung der Anweisung @TABS RANGE wird abgebrochen.
Fehlerschalter: EDT.

```
EDT4315   RANGE SYMBOL MUST NOT BE USED AS STATEMENT SYMBOL
EDT4315   BEREICHSSYMBOL DARF NICHT ALS ANWEISUNGSSYMBOL VERWENDET WERDEN
```

Bedeutung
Fuer die Neudefinition des Anweisungssymbols mit der Anweisung @: wurde das aktuelle Bereichssymbol (siehe @RANGE) angegeben. Die Anweisung wird nicht ausgefuehrt.
Fehlerschalter: EDT.

```
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
Innerhalb einer @INPUT-Prozedur ist die Anweisung @INPUT 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 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 WORK FILE
EDT4906 '(&00)' FUER AKTUELLE ARBEITSDATEI NICHT MOEGLICH

Bedeutung
Die Anweisung (&00) bezieht sich auf die aktuelle Arbeitsdatei 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

Bedeutung
Innerhalb von @DO-Prozeduren ist die Anweisung @DROP nicht erlaubt.
Fehlerschalter: EDT.

EDT4908 INVALID COMMAND
EDT4908 UNGUELTIGES KOMMANDO

Bedeutung
Das in der @SYSTEM-Anweisung angegebene Kommando ist fehlerhaft oder kann nicht ueber das CMD-Makro abgegeben werden.
Fehlerschalter: DVS.

Maßnahme
Korrigiertes Kommando wiederholen oder mit @SYSTEM ins System verzweigen.

EDT4909 WORK FILE ALREADY ACTIVE
EDT4909 ARBEITSDATEI BEREITS AKTIV

Bedeutung
In der @PROC-Anweisung wurde die Nummer der aktuellen Arbeitsdatei angegeben.
Fehlerschalter: EDT.

```
EDT4910   S-VARIABLE MUST BE OF TYPE LIST
EDT4910   S-VARIABLE NICHT VOM TYP LIST
```

Bedeutung
Die SDF-P-Variable, die in einer @GETLIST- oder @SETLIST-Anweisung oder im Kompatibilitaetsmodus in einer @LOG-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 geoeffnet 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.

```
EDT4916   FILE '(&OO)' NOT IN CATALOG
EDT4916   DATEI '(&OO)' 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 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-, @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 DER AUSFUEHRUNG EINER PROZEDURDATEI

Bedeutung
Innerhalb von @DO- oder @INPUT-Prozeduren ist die angegebene Anweisung nicht erlaubt (z.B. @INPUT, @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 geoeffneten 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.
Im Kompatibilitaetsmodus ist eine moegliche Fehlerursache, dass der in Hochkommata eingeschlossene oder in einer Zeichenfolgevariablen angegebene Dateiname mit einem Leerzeichen beginnt.
Fehlerschalter: EDT, DVS.

```
EDT4924   INVALID FORMAL OPERAND
EDT4924   UNGUELTIGER FORMALER OPERAND
```

Bedeutung
Ein formaler Operand in einer @PARAMS-Anweisung ist ungueltig.
Fehlerschalter: EDT.

```
EDT4925   STATEMENT ONLY PERMITTED IN WORK FILE 0
EDT4925   ANWEISUNG NUR IN ARBEITSDATEI 0 MOEGLICH
```

Bedeutung
Eine Datei kann nur in der Arbeitsdatei 0 real geoeffnet 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 geoeffnet (@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 "EDT8001 EDT ABNORMAL BEENDET".
Fehlerschalter: EDT, DVS.

EDT4928 INVALID VALUE
EDT4928 UNGUELTIGER WERT

Bedeutung
Moegliche Ursachen sind:
- In der Anweisung @COMPARE (Format 1) ist der Wert von int2 groesser als der von int1.
- Die in der Anweisung @SET (Format 3) nach STRING angegebene Zeichenfolge kann nicht als Zeilennummer interprepiert werden.
- Ein Wert in einem @PARAMS-Schluesselwort oder einem @DO-Operanden ist ungueltig.

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
Eine Zeilennummer wurde fuer die Definition einer Zeichenfolge angegeben, aber die Zeile ist in der Arbeitsdatei nicht vorhanden, oder in der @COMPARE-Anweisung wurde ein ungueltiger Zeilenbereich angegeben.
Fehlerschalter: EDT.
Wird bei der Ausfuehrung einer EDT-Prozedur (@DO oder @INPUT) eine Zeichenfolge durch eine nicht existierende Zeilennummer definiert, und ist die Protokollierung der Prozedur ueber den PRINT-Operanden nicht eingeschaltet, so wird diese Meldung nicht ausgegeben und der EDT-Fehlerschalter nicht gesetzt. Die Anweisung wird aber in keinem Fall ausgefuehrt.

EDT4933 MODULE LOADING NOT POSSIBLE
EDT4933 LADEN DES MODULS NICHT MOEGLICH

Bedeutung
Es ist nicht moeglich, den Modul (z.B. EDTSTRT) mit der @RUN- oder @USE-Anweisung zu laden.
Fehlerschalter: EDT.

EDT4934 FILE HAS ACCESS METHOD SAM
EDT4934 DATEI HAT ZUGRIFFSMETHODE 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 FILE IS OPENED REAL
EDT4935 DATEI IST REAL GEOEFFNET

Bedeutung
Es wurde versucht, eine Anweisung auszufuehren, die nicht erlaubt ist solange die Datei durch @OPEN real geoeffnet 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

Bedeutung
Eine Datei mit diesen Katalogeigenschaften kann mit @GET, @SAVE oder @OPEN 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 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 und der Software-Tabulatoren (im Unicode-Modus) angegeben werden, muessen aufsteigend sein.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT4941 NO POSITIONS DEFINED
EDT4941 POSITIONEN NICHT DEFINIERT

Bedeutung
Vor der Aktivierung 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- oder @GOTO-Anweisung (im Kompatibilitaetsmodus auch eine @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

Bedeutung
Ein Wechsel des kodierten Zeichensatzes ist nur moeglich, wenn alle Arbeitsdateien leer sind. Entweder wurde eine @CODENAME-Anweisung eingegeben, oder eine Datei mit einem CCS verschieden vom aktuellen CCS sollte eingelesen oder geoeffnet werden (@READ, @OPEN,..).
Fehlerschalter: EDT.

Maßnahme
Geoeffnete 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 AN DIESER DATENSTATION NICHT MOEGLICH

Bedeutung
Moegliche Ursachen:
- Es wurde versucht, mit einer @VDT-Anweisung das Bildschirmformat zu aendern, was nur fuer eine Datensichtstation 9763 moeglich ist.
- An einer Datensichtstation 3270 wurde versucht, Hardware-Tabulatoren zu definieren.

Fehlerschalter: EDT.

EDT4946 OVERFLOW ERROR
EDT4946 UEBERLAUF-FEHLER

Bedeutung
Bei der Berechnung eines ganzzahligen Ausdrucks 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 Schluesselwortoperanden angegeben.
Fehlerschalter: EDT.

EDT4949 PROCEDURE FILE IS EMPTY
EDT4949 PROZEDURDATEI IST LEER

Bedeutung
Eine mit @DO gestartete EDT-Prozedur ist leer.
Fehlerschalter: EDT.

EDT4950 WORK FILE IS UNDEFINED
EDT4950 ARBEITSDATEI IST NICHT DEFINIERT

Bedeutung
In der Anweisung @DO wurde keine Arbeitsdatei angegeben und es wurde vorher keine Arbeitsdatei mit @END verlassen.
Im Kompatibilitaetsmodus kann auch die Ursache sein, dass in einer @DO- oder @COMPARE-Anweisung eine Arbeitsdatei angegeben wird, die noch nicht belegt ist.
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.
Fehlerschalter: EDT.

EDT4952 NO SEPARATOR DEFINED
EDT4952 KEIN TRENNZEICHEN DEFINIERT

Bedeutung
Der Anwender gab @SEPARATE ohne Operand AT ein. Da auch mit @PAR SEPARATOR kein Satztrennzeichen vordefiniert ist, kann die Anweisung nicht bearbeitet werden.
Fehlerschalter: EDT.

Maßnahme
Geben Sie das Trennzeichen in der Anweisung @SEPARATOR direkt nach dem Operand AT ein oder definieren Sie es mit @PAR SEPARATOR vor.

EDT4953 NO CHARACTER FOR TABULATOR DEFINED
EDT4953 KEIN TABULATOR-ZEICHEN DEFINIERT

Bedeutung
Der Anwender gab @TABS RANGE ein, ohne vorher ein Tabulatorzeichen aktiviert zu haben.
Fehlerschalter: EDT.

Maßnahme
Definieren und aktivieren Sie ein Tabulatorzeichen und Tabulator-positionen, bevor Sie @TABS RANGE wiederholen.

```
EDT4954    REQM ERROR. PLEASE ACKNOWLEDGE. REPLY (Y=YES)
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 geschachtelte, nicht beendete Prozedurdateien vorhanden sind.
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 daraufhin mit der Meldung "EDT8001 EDT 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-Fehler aus.
Fehlerschalter: DVS.

EDT4959 WORK FILE ALREADY ACTIVE
EDT4959 ARBEITSDATEI BEREITS AKTIV

Bedeutung
Eine Arbeitsdatei, in der gerade eine @DO-Prozedur ausgefuehrt wird (aktive Arbeitsdatei), kann nicht zur aktuellen Arbeitsdatei gemacht werden.
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 "EDT8001 EDT ABNORMAL BEENDET".
Fehlerschalter: wird nicht gesetzt.

EDT4961 TOO MANY PROCEDURE FILES ACTIVE
EDT4961 ZU VIELE AKTIVE PROZEDUR-DATEIEN

Bedeutung
Es koennen maximal 22 Prozedurdateien gleichzeitig aktiv sein.
Die @DO-Anweisung wird abgewiesen.
Fehlerschalter: EDT.

EDT4962 TOO MANY NESTED PROCEDURE FILES
EDT4962 ZU VIELE GESCHACHTELTE PROZEDUR-DATEIEN

Bedeutung
Es koennen maximal 22 Prozedurdateien geschachtelt werden.
Die @PROC-Anweisung wird abgewiesen.
Im Kompatibilitaetsmodus wird auch die @INPUT-Anweisung abgewiesen.
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 LINE NUMBER AREA EMPTY
EDT4964 SPEICHERBEREICH FUER ZEILENNUMMERN IST LEER

Bedeutung
Eine '@'-Anweisung (oder im Unicode-Modus eine @SET-Anweisung, Format 6) wurde eingegeben, um das zuletzt abgelegte Wertepaar Zeilennummer/ Schrittweite zur aktuellen Zeilennummer und Schrittweite zu machen. Es sind jedoch keine Eintraege im Speicherbereich mehr vorhanden.
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-Operand 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-Operand wurde fuer eine SAM-Datei eingegeben.
Fehlerschalter: EDT.

EDT4968 WORK FILE NOT EMPTY
EDT4968 ARBEITSDATEI NICHT LEER

Bedeutung
Es befinden sich noch Zeilen in der Arbeitsdatei. Das Oeffnen zur realen Bearbeitung mit @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
Die @ELIM-Anweisung kann nur auf ISAM-Dateien angewendet werden.
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 momentan nicht zugelassen ist.
Moegliche Ursachen sind:
- Ablauf unter Kennungen mit bestimmten Privilegien
- Ablauf in einer nicht-unterbrechbaren Prozedur
- Anweisung wurde vom rufenden Programm gesperrt.

Fehlerschalter: EDT.

```
EDT4977    'RECORD-FORMAT=*UNDEFINED' ILLEGAL
EDT4977    'RECORD-FORMAT=*UNDEFINED' NICHT ZULAESSIG

EDT4978    INVALID IN F-MODE
EDT4978    NICHT ZULAESSIG FUER F-MODUS
```

Bedeutung
Die Anweisung ist im Bildschirm-Dialog-Modus nicht erlaubt.

```
EDT4980    ILLEGAL OR UNKNOWN CCS NAME
EDT4980    UNZULAESSIGER ODER UNBEKANNTER CCS-NAME
```

Bedeutung
Der in der Anweisung @CODENAME angegebene CCS-Name, der CCS der einzulesenden Datei bzw. des Bibliothekselementes (@READ, @OPEN, @COPY, @INPUT) oder der im CODE-Operanden einer Anweisung im Unicode-Modus angegebene CCS-Name ist im System nicht bekannt.
Fehlerschalter: EDT.

```
EDT4981    RECORD-SIZE > 256. FILE NOT WRITTEN
EDT4981    RECORD-SIZE > 256. DATEI NICHT GESCHRIEBEN
```

EDT4982 REQUESTED JOB VARIABLE NOT CATALOGED
EDT4982 JOBVARIABLE NICHT IM KATALOG

Bedeutung
In einer @STAJV-, @ERAJV- oder @GETJV-Anweisung wurde der Name einer Jobvariablen angegeben, die nicht im Katalog vorhanden ist, oder in einer @STAJV-Anweisung wurde kein Name angegeben und in der aktuellen Kennung sind keine Jobvariablen vorhanden.
Fehlerschalter: EDT, DVS.

EDT4983 CHANGE OF OPERATION MODE NOT POSSIBLE
EDT4983 WECHSEL DES BETRIEBSMODUS NICHT MOEGLICH

Bedeutung
Es wurde eine Anweisung zur Umschaltung vom Kompatibilitaets-Modus in den Unicode-Modus (@MODE, @CODENAME) oder umgekehrt (@MODE) angegeben. Die Umschaltung ist aber nur moeglich, wenn alle Arbeitsdateien leer sind und keine Dateien geoeffnet sind.
Wird die Anweisung zum Umschalten in den Kompatibilitaets-Modus aus einer EDT-Startprozedur oder von SYSDTA gelesen, so muss zusaetzlich der Zeichensatz der Eingabedatei im Kompatibilitaets-Modus verwendbar sein.
Fehlerschalter: EDT.

Maßnahme
Geoeffnete Dateien schliessen (@CLOSE), Arbeitsdateien loeschen (@DELETE) und danach Anweisung wiederholen. Gegebenenfalls Eingabedatei korrigieren.

EDT4984 NON-NUMERIC KEY
EDT4984 NICHT-NUMERISCHER SCHLUESSEL

Bedeutung
Beim Einlesen wurde auf einen Satz zugegriffen, dessen Schluessel sich nicht in eine EDT-Zeilennummer umwandeln laesst. Die Verarbeitung der Anweisung wurde abgebrochen.
Fehlerschalter: EDT, DVS.

EDT4985 WRONG VERSION FOR FILE (&00), CORRECT VERSION (&01)
EDT4985 FALSCHE VERSION FUER DATEI (&00), KORREKTE VERSION (&01)

Bedeutung
Die Datei (&00) wurde bei einem Schreib-Zugriff mit falscher Versionsnummer angegeben. Die richtige Versionsnummer ist (&01).
Die Anweisung wird nicht ausgefuehrt.
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.

```
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.

```
EDT5122   NO FILE NAME
EDT5122   KEIN DATEINAME
```

Bedeutung
In der Anweisung @WRITE oder @XWRITE wurde kein Dateiname angegeben und es ist keine Datei zum Zurueckschreiben geoeffnet.
Fehlerschalter: EDT.

```
EDT5126   '(&OO)' NOT POSSIBLE: WORK FILE 0 IS OPEN
EDT5126   KEIN '(&OO)' 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.

```
EDT5177   NO FILE TO CLOSE
EDT5177   KEINE OFFENE DATEI VORHANDEN
```

Bedeutung
Es wurde eine @CLOSE-Anweisung angegeben, aber es ist keine Datei geoeffnet.
Fehlerschalter: EDT.

```
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 geoeffnet ist.
Fehlerschalter: EDT.

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
Weder in der Anweisung selbst noch mit der Anweisung @PAR LIBRARY wurde ein Bibliotheksname definiert.
Fehlerschalter: EDT.

```
EDT5188   NUMBER OF LINES NOT PERMISSIBLE
EDT5188   ZEILENANZAHL NICHT ERLAUBT
```

Bedeutung
In der Anweisung SPLIT oder im Operanden SPLIT der @PAR-Anweisung wurde eine Zeilenanzahl fuer das zweite Arbeitsfenster angegeben, bei der eines der Arbeitsfenster eine Zeilenanzahl kleiner 2 haette.
Fehlerschalter: EDT.

```
EDT5189   '(&00)' NOT POSSIBLE: A FILE IS OPENED IN WORK FILE 9
EDT5189   '(&00)' NICHT MOEGLICH: IN ARBEITSDATEI 9 IST EINE DATEI GEOEFFNET
```

Bedeutung
Eine (&00)-Anweisung konnte nicht ausgefuehrt werden, da in Arbeitsdatei 9 eine Datei geoeffnet ist.

Maßnahme
Die in der Arbeitsdatei 9 geoeffnete 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, @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
Moegliche Ursachen:
- Die Datei, die mit einer @COPY-, @OPEN- oder @WRITE-Anweisung angesprochen wurde, kann vom EDT nicht verarbeitet werden, da sie eine andere Zugriffsmethode als SAM oder ISAM hat.
- Eine noch nicht existierende Datei, der der Dateikettungsname EDTISAM und die Zugriffsmethode SAM zugeordnet sind, soll mit der Anweisung @SAVE geschrieben werden.

Fehlerschalter: EDT, DVS.

Maßnahme
Datei in eine SAM- oder ISAM-Datei konvertieren bzw. Datei-Zugriffsmetode aendern.

```
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
```

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
```

Bedeutung
Die ISAM-Datei wurde geschrieben. Danach trat beim Restaurieren 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 Operanden LIBRARY der @OPEN-, @COPY-, @WRITE-, @DELETE-, @INPUT- oder @SHOW-Anweisung angegeben wurde oder die in einer @PAR-Anweisung als LIBRARY vordefiniert wurde, kann mit PLAM nicht zugegriffen werden.
Fehlerschalter: EDT.

```
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 angegebene Datei (&00) existiert bereits. Die Anweisung wurde nicht ausgefuehrt.
Fehlerschalter: EDT.

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 FEHLERCODE '(&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
```

Bedeutung
Die in der Anweisung angesprochene Bibiothek kann nicht bearbeitet werden, da sie nicht existiert.
Fehlerschalter: EDT.

```
EDT5268   MEMBER '(&00)' IS LOCKED
EDT5268   ELEMENT '(&00)' IST GESPERRT
```

Bedeutung
Das angegebene Element konnte nicht angesprochen werden, da es entweder gegen unberechtigte Zugriffe geschuetzt oder bereits geoeffnet ist.
Fehlerschalter: EDT.

```
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
```

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
```

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
```

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 '(&OO)' DOES NOT EXIST
EDT5275   'COPY' NICHT MOEGLICH: ELEMENT '(&OO)' NICHT GEFUNDEN
```

Bedeutung
Das angegebene Element konnte nicht in den virtuellen Datenbereich kopiert werden, da das Element nicht existiert.
Die Anweisung (@COPY, @INPUT Format2) wurde nicht ausgefuehrt.

Maßnahme
PLAM-Typ des Elementes ueberpruefen.

```
EDT5278   FILE '(&OO)' PROTECTED BY PASSWORD
EDT5278   DATEI '(&OO)' DURCH PASSWORT GESCHUETZT
```

Bedeutung
Fehlerschalter: EDT, DVS.

Maßnahme
Sich mit dem Dateieigentuemer in Verbindung setzen.

```
EDT5279   FILE '(&OO)' LOCKED
EDT5279   DATEI '(&OO)' GESPERRT
```

Bedeutung
Die in der Anweisung angesprochene Datei ist bereits geoeffnet oder im Kompatibilitaetsmodus durch ACCESS=READ geschuetzt.
Fehlerschalter: EDT, DVS.

```
EDT5281   FILE '(&OO)' DOES NOT EXIST
EDT5281   DATEI '(&OO)' EXISTIERT NICHT
```

Bedeutung
Die in der Anweisung angesprochene Datei kann nicht bearbeitet werden, da sie nicht existiert.
Fehlerschalter: EDT.

```
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.
Fehlerschalter: EDT.

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
```

Bedeutung
Das angesprochene Bibliothekselement ist nicht editierbar. Der angegebene Benutzertyp entspricht nicht einem der PLAM-Typen S,M,P,J,D oder 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.
Fehlerschalter: EDT.

EDT5289 LINK NAME FOR JOB VARIABLE NOT DEFINED
EDT5289 LINKNAME FUER JOBVARIABLE 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 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 (@STAJV oder @ERAJV) mit einem teilqualifizierten 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.

EDT5295 NO MORE MEMORY AVAILABLE
EDT5295 KEIN SPEICHERPLATZ MEHR VORHANDEN

Bedeutung
Fuer die Dateibearbeitung ist nicht mehr genuegend virtueller Speicher vorhanden.
Fehlerschalter: EDT.

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

Bedeutung
Die angegebene UFS-Datei (&00) kann nicht bearbeitet werden, da sie nicht existiert.
Fehlerschalter: EDT.

EDT5311 UFS FILE '(&00)' ALREADY EXISTS
EDT5311 UFS DATEI '(&00)' EXISTIERT BEREITS

Bedeutung
Die Datei, die in einer @OPEN-, @WRITE-, @XOPEN- oder @XWRITE-Anweisung angegeben wurde, kann nicht mit MODE=NEW bearbeitet werden, da sie bereits als UFS-Datei vorhanden ist.
Fehlerschalter: EDT.

EDT5312 INVALID ACCESS TO UFS FILE '(&00)'
EDT5312 UNGUELTIGER ZUGRIFF AUF UFS DATEI '(&00)'

Bedeutung
Die UFS-Datei, die in einer @OPEN-, @COPY-, @WRITE-, @INPUT-, @XOPEN-, @XCOPY- oder @XWRITE-Anweisung angegeben wurde, kann nicht geoeffnet 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 UFS-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

Bedeutung
Der Anwender gab @SDFTEST PROGRAM ein oder markierte eine Bildschirmzeile, die mit '//' beginnt, mit der Kurzanweisung T, es ist aber noch kein Programmname definiert.
Fehlerschalter: EDT.

Maßnahme
Anweisung @SDFTEST mit dem Operand PROGRAM=name eingeben oder einen Programmnamen mit der Anweisung @PAR SDF-PROGRAM=name definieren.

EDT5321 SDF: PROGRAM NAME UNKNOWN
EDT5321 SDF: PROGRAMMNAME UNBEKANNT

Bedeutung
Der 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

Bedeutung
Die Bearbeitung von @SDFTEST oder der Kurzanweisung T wurde abgebrochen. Eine moegliche Ursache ist, dass die zu testende Anweisung mehr als 255 Fortsetzungszeilen hat.
Fehlerschalter: EDT.

EDT5323 SDF: EXTERNAL PROGRAM NAME NOT SUPPORTED
EDT5323 SDF: EXTERNER PROGRAMM-NAME NICHT UNTERSTUETZT

Bedeutung
Die Angabe des externen Programm-Namens in der @SDFTEST- oder @PAR SDF-PROGRAM-Anweisung ist mit der aktuellen SDF-Version nicht moeglich.
Fehlerschalter: EDT.

Maßnahme
Internen Programm-Namen verwenden.

EDT5324 SDF: SYNTAX TEST ABORTED BY USER
EDT5324 SDF: SYNTAX-TEST VOM BENUTZER ABGEBROCHEN

Bedeutung
Die Syntaxpruefung von SDF-Kommandos oder SDF-Anweisungen mit @SDFTEST wurde vom Benutzer abgebrochen.
Fehlerschalter: EDT.

EDT5325 SDF: LINE TOO LONG
EDT5325 SDF: ZEILE ZU LANG

Bedeutung
Die zur Syntaxpruefung zu uebergebende Zeile ueberschreitet die von SDF zugelassene maximale Laenge (16379 Zeichen). Die Syntaxpruefung wird abgebrochen.
Fehlerschalter: EDT.

EDT5326 SDF: OUTPUT TOO LONG
EDT5326 SDF: AUSGABE ZU LANG

Bedeutung
Die Ausgabe einer von SDF gepruefften Zeile ueberschreitet die von SDF zugelassene maximale Laenge (16379 Zeichen). Die Syntaxpruefung wird abgebrochen.
Fehlerschalter: EDT.

EDT5327 CANNOT CONVERT TO TERMINAL CCS
EDT5327 KONVERTIEREN IN DEN TERMINAL-ZEICHENSATZ NICHT MOEGLICH

Bedeutung
Der aktuelle Zeichensatz kann nicht in den Terminal-Zeichensatz konvertiert werden. Es ist kein SDF-Fehlerdialog moeglich. Die Syntaxpruefung wird abgebrochen.
Fehlerschalter: EDT.

EDT5340 CANNOT GET S-VARIABLE
EDT5340 S-VARIABLE KANN NICHT AUSGEGEBEN WERDEN

Bedeutung
Die Anweisung @GETVAR oder @GETLIST konnte nicht ausgefuehrt werden, weil der Variablen kein Wert zugewiesen ist oder die Liste kein Element enthaelt.
Fehlerschalter: EDT.

EDT5341 S-VARIABLE LONGER THAN 256 CHARACTERS
EDT5341 S-VARIABLE LAENGER ALS 256 ZEICHEN

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

Bedeutung
Die @GETVAR- oder @SETVAR-Anweisung konnte nicht ausgefuehrt werden, da der Typ der angegebenen Variablen nicht mit dem Wert des Operanden an der rechten 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

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: EDT.

```
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
Beim Aufbau der Daten fuer die Bildschirmausgabe trat ein nicht behebbarer Format-Fehler auf. Der EDT wird beendet.

```
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.
Fehlerschalter: EDT.

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 <ln> 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 <ln> 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. Die Kurzanweisung A/B/O kann nicht ausgefuehrt werden.

```
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 am Ende der Arbeitsdatei mit einer Anweisung oder Kurzanweisung wird nicht ausgefuehrt, da die groesste moegliche Zeilennummer ueberschritten werden muesste.
Fehlerschalter: EDT.

```
EDT5365   NO INSERT: RENUMBERING INHIBITED
EDT5365   KEIN EINFUEGEN: NEUNUMMERIEREN VERBOTEN
```

Bedeutung
Die gewuenschten Datenzeilen koennen nicht eingefuegt werden, ohne bestehende Datenzeilen neu zu nummerieren. Dies ist nicht moeglich, da in einer @PAR-Anweisung der Operand RENUMBER=OFF 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 ohne P-Tasten aufgerufen.

```
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.

```
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 USER SYMBOLS ARE PERMITTED
EDT5373   MAXIMAL 5 BENUTZERSYMBOLE MOEGLICH
```

Bedeutung
Mit der @USE-Anweisung koennen maximal 5 Benutzeranweisungssymbole definiert werden.
Fehlerschalter: EDT.

Maßnahme
Ein Symbol mit @USE-Anweisung loeschen und dann neu definieren.

```
EDT5375   NO 'USE' ENTRY DEFINED WITH SPECIFIED SYMBOL
EDT5375   KEINE 'USE'-ROUTINE MIT ANGEGEBENEM SYMBOL VORHANDEN
```

Bedeutung
Es wurde versucht, eine nicht definierte USE-Anweisungsroutine zu deaktivieren.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

```
EDT5376   STATEMENT BUFFER EMPTY
EDT5376   ANWEISUNGSPUFFER LEER
```

Bedeutung
Die SHIH-Anweisung kann nicht ausgefuehrt werden, da keine Anweisungen im Anweisungspuffer gespeichert sind.

EDT5380 SOME JOB VARIABLES NOT ERASED
EDT5380 EINIGE JOBVARIABLEN NICHT GELOESCHT

Bedeutung
Es wurde versucht, alle Jobvariablen zu loeschen, deren Name teilqualifiziert oder mit Wildcards angegeben wurde,
doch das Loeschen einiger Jobvariablen wurde vom System nicht zugelassen.
Moegliche Ursachen:
- Jobvariable ist nur lesend zugreifbar.
- Jobvariable ist als MONJV geschuetzt.
Fehlerschalter: EDT.

EDT5381 JOB VARIABLES NOT ERASED
EDT5381 JOBVARIABLEN NICHT GELOESCHT

Bedeutung
Mehrere Jobvariablen werden duch die Zeichenfolge-Angabe in der Anweisung @ERAJV im Stapelbetrieb bezeichnet und der Operand ALL ist nicht angegeben.
Die Anweisung @ERAJV wird nicht ausgefuehrt.
Fehlerschalter: EDT.

EDT5400 NOT SUPPORTED ON THIS INTERFACE
EDT5400 AN DIESER SCHNITTSTELLE NICHT UNTERSTUETZT

Bedeutung
Die Anweisung ist in diesem Arbeitsmodus oder an dieser Unterprogramm-Schnittstelle nicht erlaubt.
Fehlerschalter: EDT.

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

Bedeutung
Ein Anweisungsfilter kann nur dann definiert werden, wenn die Anweisung @USE an der Unterprogrammschnittstelle angegeben wird.
Fehlerschalter: EDT.

EDT5410 UNDEFINED ERROR IN USER PROGRAM
EDT5410 UNDEFINIERTER FEHLER IM ANWENDER-PROGRAMM

Bedeutung
Der EDT erhaelt vom Anwenderprogramm einen undefinierten Returncode ohne weitere Informationen.
Fehlerschalter: EDT.

Maßnahme
Anwenderprogramm korrigieren.

EDT5419 (&00)
EDT5419 (&00)

Bedeutung
Der EDT erhaelt vom Anwenderprogramm einen undefinierten Returncode mit der Beschreibung (&00).
Fehlerschalter: EDT.

Maßnahme
Anwenderprogramm korrigieren.

EDT5442 TERMINAL BUFFER TOO SMALL. SCREEN FORMAT SET TO F1
EDT5442 PUFFER DER DATENSICHTSTATION ZU KLEIN. BILDSCHIRMFORMAT F1 EINGESTELLT

Bedeutung
Die Ausgabe im eingestellten Bildschirmformat ist laenger als der Puffer der Datensichtstation. Es wird auf das Format F1 umgeschaltet.
Fehlerschalter: EDT.

EDT5443 TERMINAL BUFFER TOO SMALL. CHANGED TO L-MODE
EDT5443 PUFFER DER DATENSICHTSTATION ZU KLEIN. L-MODUS EINGESTELLT

Bedeutung
Die Bildschirmausgabe im F-Modus ist laenger als der Puffer der Datensichtstation. Es wird in den L-Modus umgeschaltet.
Fehlerschalter: EDT.

EDT5444 FILE WRITE NOT POSSIBLE. RECORD LONGER THAN (&00) BYTES
EDT5444 DATEI KANN NICHT GESCHRIEBEN WERDEN. SATZ IST LAENGER ALS (&00) BYTES

Bedeutung
Beim Schreiben mit einer der Anweisungen @WRITE, @CLOSE oder @SAVE oder beim Schreiben in eine real geoeffnete Datei ist ein Satz
laenger als die maximal zulaessige Satzlaenge (&00) der Datei.
Die Satzlaenge kann durch den Blockungsfaktor oder durch feste Satzlaenge begrenzt sein. Es ist zu beachten, dass die Satzlaenge in Bytes bei Unicode-Zeichensaetzen groesser sein kann als die Anzahl der Zeichen der Arbeitsdateizeile und vom gewaehlten Zeichensatz abhaengen kann.
Das Schreiben der Datei wird abgebrochen bzw. die real geoeffnete Datei wird geschlossen.
Fehlerschalter: EDT.

EDT5445 DUPLICATE KEYS IN REAL OPENED FILE DETECTED
EDT5445 REAL GEOEFFNETE DATEI ENTHAELT MEHRFACH AUFTRETENDE SCHLUESSEL

Bedeutung
In einer durch @OPEN real geoeffneten Datei wurden mehrfach auftretende Schluesselwerte gefunden. Die Bearbeitung der Datei wird abgebrochen.
Fehlerschalter: EDT.

EDT5446 LINE NUMBER TOO BIG
EDT5446 ZEILENNUMMER ZU GROSS

Bedeutung
In einer durch @OPEN real geoeffneten Datei kann eine Zeilennummer nicht als Schluessel verwendet werden, da sie fuer die angegebene Schluessel-laenge der Datei zu gro? ist.
Fehlerschalter: EDT.

EDT5447 PRINTING FAILED
EDT5447 DRUCKEN FEHLGESCHLAGEN

Bedeutung
Die sofortige Ausgabe auf den Drucker konnte nicht ausgefuehrt werden, da bei der Ausfuehrung der Systemfunktion zum Drucken ein Fehler aufgetreten ist.
Fehlerschalter: EDT, DVS.

EDT5448 CANNOT USE TEMPORARY FILE
EDT5448 TEMPORAERE DATEI KANN NICHT VERWENDET WERDEN

Bedeutung
Fuer die sofortige Ausgabe eines Arbeitsdateibereichs auf den Drucker wird eine temporaere Datei benoetigt. Diese steht jedoch wegen der aktuell eingestellten BS2000-Systemparameter nicht zur Verfuegung.
Fehlerschalter: EDT.

EDT5449 NOT ENOUGH LINES FOR HEX MODE
EDT5449 ZUWENIGE ZEILEN FUER HEX-DARSTELLUNG VORHANDEN

Bedeutung
Auf einem geteilten Bildschirm stehen weniger Zeilen zur Verfuegung, als fuer die Darstellung einer Datenzeile im Hexadezimalmodus benoetigt werden.
Fehlerschalter: EDT.

Maßnahme
Arbeitsfenster vergroessern (@PAR SPLIT).

EDT5450 NATIONAL EDT: INTERNAL ERROR. RETURNCODE = X'(&00)' AT CCS (&01)
EDT5450 NATIONALER EDT: INTERNER FEHLER. FEHLERCODE = X'(&00)' BEI CCS (&01)

Bedeutung
Die Ausfuehrung des Benutzer-CCS (&01) wurde an einer nationalen Sichtstation wegen eines internen Fehlers abgebrochen.
(&00): Fehlercode.

Maßnahme
Systemkundendienst verstaendigen.

EDT5451 CANNOT OPEN TAPE FILE
EDT5451 BANDDATEI KANN NICHT GEOEFFNET WERDEN

Bedeutung
Eine Datei auf Magnetband kann nicht mit der Anweisung @OPEN geoeffnet werden.
Fehlerschalter: EDT.

EDT5452 CHANGE OF CCS NOT POSSIBLE FOR REAL OPENED FILES
EDT5452 FUER REAL GEOEFFNETE DATEIEN KANN CCS NICHT GEAENDERT WERDEN

Bedeutung
Der Zeichensatz der Arbeitsdatei, in der eine Datei zur realen Bearbeitung geoeffnet ist, soll geaendert werden, oder der Zeichensatz der real zu oeffnenden Datei ist verschieden vom Zeichensatz der aktuellen Arbeitsdatei. Die Anweisung @CODENAME bzw. @OPEN wird nicht ausgefuehrt.
Fehlerschalter: EDT.

EDT5453 SOME CHARACTER CANNOT BE CONVERTED
EDT5453 EINIGE ZEICHEN KOENNEN NICHT KONVERTIERT WERDEN

Bedeutung
Die Zeichenfolge enthaelt Zeichen, die im Ziel-Zeichensatz nicht enthalten sind. Sie kann nicht in den Ziel-Zeichensatz konvertiert werden.
Fehlerschalter: EDT.

Maßnahme
Ersatzzeichen definieren mit @PAR SUBSTITUTION-CHARACTER.

EDT5454 ILLEGAL BYTE SEQUENCE IN INPUT DATA
EDT5454 UNGUELTIGE BYTEFOLGE IN EINGABE-DATEN

Bedeutung
Eine Datei, eine Job-Variable, eine S-Variable oder ein an der Unterprogramm-Schnittstelle gelieferter Satz enthaelt eine Bytefolge, die in dem gewuenschten Zeichensatz nicht als Zeichen interpretiert werden kann. Die Datei, die Variable oder der Satz wird nicht eingelesen.
Fehlerschalter: EDT.

EDT5456 SOME CHARACTERS CANNOT BE CONVERTED AND SOME LINES MUST BE TRUNCATED
EDT5456 EINIGE ZEICHEN KOENNEN NICHT KONVERTIERT WERDEN UND EINIGE ZEILEN SIND ZU LANG

Bedeutung
Bei der Pruefung mit @CHECK (Format 2) wurden sowohl Zeichen gefunden, die im Ziel-Zeichensatz nicht enthalten sind, als auch Zeilen oder Zeichenfolgevariable, bei denen die maximal zulaessige Laenge ueberschritten wird.
Fehlerschalter: EDT.

Maßnahme
Ersatzzeichen definieren mit @PAR SUBSTITUTION-CHARACTER.
Zeilen oder Zeichenfolgevariable aufteilen oder kuerzen.

EDT5457 FILE CCS DIFFERENT FROM WORK FILE CCS, FILE NOT WRITTEN
EDT5457 CCS DER DATEI VERSCHIEDEN VOM CCS DER ARBEITSDATEI, DATEI NICHT GESCHRIEBEN

Bedeutung
Der Zeichensatz der Arbeitsdatei unterscheidet sich vom Zeichensatz der Datei, in die die Arbeitsdatei geschrieben werden soll. Es wird nicht geschrieben.
Fehlerschalter: EDT.

Maßnahme
Operand CODE in der Anweisung @CLOSE oder @WRITE angeben.

```
EDT5458   CCS CANNOT BE CHANGED
EDT5458   CCS KANN NICHT GEAENDERT WERDEN
```

Bedeutung
In der Anweisung @CREATE wird ein Zeichensatz fuer die aktuelle Arbeitsdatei angegeben, diese hat jedoch bereits einen anderen Zeichensatz. Die Anweisung @CREATE wird nicht ausgefuehrt.
Fehlerschalter: EDT.

```
EDT5459   CANNOT CREATE LINE NUMBER FROM ISAM KEY
EDT5459   ZEILENNUMMER KANN NICHT AUS ISAM-SCHLUESSEL GEBILDET WERDEN
```

Bedeutung
Beim Einlesen einer ISAM-Datei, deren Schluesselposition vom Standard abweicht, deren Schluessellaenge zu gross ist oder deren Schluessel nicht numerisch ist, soll die Zeilennummer aus dem ISAM-Schluessel gebildet werden. Die Datei wird nicht eingelesen.
Fehlerschalter: EDT.

Maßnahme
Datei mit KEY=DATA einlesen.

```
EDT5460   CANNOT CONVERT HEX OR BIN CHARACTER TO WORK FILE CCS
EDT5460   HEX-/BINAERZEICHEN KOENNEN NICHT IN ARBEITSDATEI-CCS KONVERTIERT WERDEN
```

Bedeutung
Eine Hex- oder Binaereingabe entspricht keinem gueltigen Zeichen im Zeichensatz der Arbeitdatei. Die Eingabe wird abgewiesen.
Fehlerschalter: EDT.

```
EDT5461   QUOTE SYMBOL MUST BE DIFFERENT TO WILDCARD SYMBOLS
EDT5461   BEGRENZERSYMBOL DARF KEIN JOKERZEICHEN SEIN
```

Bedeutung
Ein Begrenzersymbol fuer Zeichenfolgen (Hochkomma, Anfuehrungszeichen) soll in ein Zeichen umdefiniert werden, das schon als Jokersymbol (siehe @SYMBOLS) verwendet wird. Die Anweisung @QUOTE wird nicht ausgefuehrt.
Fehlerschalter: EDT.

```
EDT5462   SOME LINES MUST BE TRUNCATED
EDT5462   EINIGE ZEILEN SIND ZU LANG
```

Bedeutung
Bei der Pruefung mit @CHECK (Format 2) wurden Zeilen oder Zeichenfolgevariable gefunden, bei denen die maximal zulaessige Laenge ueberschritten wird.
Fehlerschalter: EDT.

Maßnahme
Zeilen oder Zeichenfolgevariable aufteilen oder kuerzen.

EDT5463 TABULATOR POSITION NUMBER EXCEEDS MAXIMUM VALUE SUPPORTED BY HARDWARE
EDT5463 ANZAHL DER TABULATORPOSITIONEN GROESSER ALS VON DER HARDWARE ERLAUBT

Bedeutung
In der Anweisung @TABS wurden mehr Hardwaretabulator-Positionen definiert, als mit der Datensichtstation moeglich sind. Die Anweisung wird nicht ausgefuehrt.
Fehlerschalter: EDT.

EDT5465 CANNOT CREATE ISAM KEY FROM LINE NUMBER
EDT5465 ISAM-SCHLUESSEL KANN NICHT AUS ZEILENNUMMER GEBILDET WERDEN

Bedeutung
Beim Schreiben einer ISAM-Datei, deren Schluesselposition vom Standard abweicht oder deren Schluessellaenge zu gross ist, soll der ISAM-Schluessel aus der Zeilennummer gebildet werden. Die Datei wird nicht geschrieben.
Fehlerschalter: EDT.

Maßnahme
Datei mit KEY=DATA schreiben.

EDT5466 CANNOT IGNORE ISAM KEY
EDT5466 ISAM-SCHLUESSEL KANN NICHT IGNORIERT WERDEN

Bedeutung
Beim Einlesen einer ISAM-Datei, deren Schluesselposition vom Standard abweicht, soll der ISAM-Schluessel nicht mit uebernommen werden. Die Datei wird nicht eingelesen.
Fehlerschalter: EDT.

Maßnahme
Datei mit KEY=DATA einlesen.

EDT5467 NO FILE OPEN
EDT5467 KEINE DATEI GEOEFFNET

Bedeutung
In der Anweisung @CHECK wurde CODE=*FILE angegeben, obwohl keine Datei geoeffnet ist.
Fehlerschalter: EDT.

Maßnahme
Operand CODE mit anderem Wert eingeben.

```
EDT5468   LENGTH OF ISAM KEY CHANGED, FILE NOT WRITTEN
EDT5468   LAENGE DES ISAM-SCHLUESSELS GEAENDERT, DATEI NICHT GESCHRIEBEN
```

Bedeutung
Wird waehrend der Verarbeitung einer mit @OPEN geoeffneten ISAM-Datei der Zeichensatz von dem oder in den Zeichensatz UTF16 geaendert, bzw. geschieht dies implizit durch entsprechende Angabe des CODE-Operanden in der Anweisung @WRITE oder @CLOSE, kann die Datei nicht zurueckgeschrieben werden, da sich die Laenge des Schluesselfeldes dadurch aendern wuerde.
Fehlerschalter: EDT.

```
EDT5469   INITIALIZATION ROUTINE MISSING
EDT5469   INITIALISIERUNGSROUTINE FEHLT
```

Bedeutung
Zu einer Anwenderroutine im V17-Format muss eine Initialisierungsroutine definiert sein. Die Anweisung @RUN wird nicht ausgefuehrt.
Fehlerschalter: EDT.

```
EDT5470   VERSION ERROR IN INITIALIZATION ROUTINE
EDT5470   VERSIONS-FEHLER IN INITIALISIERUNGSROUTINE
```

Bedeutung
Die Initialisierungsroutine zu einer Anweisungsroutine oder zu einer Anwenderroutine unterstuetzt die Version des Kontrollblocks nicht (Routine meldet Returncode EUPVEERR).
Die Anweisung wird nicht ausgefuehrt.
Fehlerschalter: EDT.

```
EDT5471   RUNTIME ERROR IN INITIALIZATION ROUTINE
EDT5471   LAUFZEIT-FEHLER IN INITIALISIERUNGSROUTINE
```

Bedeutung
In der Initialisierungsroutine zu einer Anweisungsroutine oder zu einer Anwenderroutine ist ein Fehler aufgetreten (Routine meldet Returncode EUPRTERR).
Fehlerschalter: EDT.

```
EDT5472   MINIMUM LINE NUMBER
EDT5472   MINIMALE ZEILENNUMMER
```

Bedeutung
Mit der Anweisung @- soll eine aktuelle Zeilennummer eingestellt werden, die kleiner als die kleinstmoegliche Zeilennummer (0.0001) ist.
Fehlerschalter: EDT.

EDT5473 CCS OF STRING VARIABLES INCONSISTENT
EDT5473 CCS VON ZEICHENFOLGEVARIABLEN INKONSISTENT

Bedeutung
Soll ein Bereich von Zeichenfolgevariablen mit der Anweisung @SEQUENCE (Format 3) geprueft werden, so muessen entweder alle einen Unicode-Zeichensatz oder alle einen 7-Bit- oder 8-Bit-Zeichensatz haben.
Die Anweisung wird nicht ausgefuehrt.
Fehlerschalter: EDT.

EDT5474 MODIFIED LINE > 32768 CHARACTERS
EDT5474 MODIFIZIERTE ZEILE > 32768 ZEICHEN

Bedeutung
Eine Arbeitsdateizeile wuerde bei der Bearbeitung mit einer der Anweisungen @COLUMN, @PREFIX, @SUFFIX zu lang werden. Die Ausfuehrung der Anweisung wird abgebrochen.
Fehlerschalter: EDT.

EDT5475 LINE NUMBER OUT OF RANGE
EDT5475 ZEILENNUMMER NICHT IM ZULAESSIGEN BEREICH

Bedeutung
Bei einer indirekt oder als arithmetischer Ausdruck angegebenen Zeilennummer liegt der Wert ausserhalb des zulaessigen Bereiches 0000.0001..9999.9999.
Fehlerschalter: EDT.

Maßnahme
Korrigierte Anweisung wiederholen.

EDT5476 CANNOT DELETE AN ACTIVE WORK FILE
EDT5476 AKTIVE ARBEITSDATEI KANN NICHT GELOESCHT WERDEN

Bedeutung
Eine Arbeitsdatei, in der gerade eine @DO-Prozedur ausgefuehrt wird (aktive Arbeitsdatei), kann nicht geloescht werden.
Fehlerschalter: EDT.

EDT5477 STRING CONTENT INVALID
EDT5477 INHALT DER ZEICHENFOLGE UNZULAESSIG

Bedeutung
Eine als Ganzzahl zu interpretierende Zeichenfolge in der Anweisung @SET (Format 1) oder eine als Zeilennummer zu interpretierende Zeichenfolge in der Anweisung @SET (Format 3) hat einen unzulaessigen Inhalt.
Fehlerschalter: EDT.

EDT5478 @PARAMS STATEMENT INVALID
EDT5478 @PARAMS-ANWEISUNG UNGUELTIG

Bedeutung
Ein syntaktischer Fehler ist in der @PARAMS-Anweisung einer auszufuehrenden @DO-Prozedur aufgetreten.
Fehlerschalter: EDT.

EDT5479 @PARAMS STATEMENT NOT FIRST LINE OF PROCEDURE
EDT5479 @PARAMS-ANWEISUNG NICHT IN DER ERSTEN ZEILE DER PROZEDUR

Bedeutung
Die @PARAMS-Anweisung muss in der ersten Zeile einer auszufuehrenden @DO-Prozedur stehen.
Fehlerschalter: EDT.

EDT5480 LINK NAME '(&00)' NOT DEFINED
EDT5480 DATEIKETTUNGSNAME '(&00)' NICHT DEFINIERT

Bedeutung
Der in einer @COPY-, @OPEN-, @WRITE- oder @INPUT-Anweisung angegebene Dateikettungsname ist nicht definiert.
Fehlerschalter: EDT, DVS.

EDT5481 INVALID RECORD-FORMAT
EDT5481 'RECORD-FORMAT' UNGUELTIG

Bedeutung
Es wird auf eine Datei zugegriffen, die entweder das RECORD-FORMAT-Attribut UNDEFINED oder das RECORD-FORMAT-Attribut FIXED und zusaetzlich eine ungerade Satzlaenge und den Zeichensatz UTF16 hat.
Die Anweisung wird nicht ausgefuehrt.
Fehlerschalter: EDT, DVS.

EDT5482 INVALID ACCESS TO FILE '(&00)'
EDT5482 UNGUELTIGER ZUGRIFF AUF DATEI '(&00)'

Bedeutung
Die Datei (&00), die in einer Ein-/Ausgabe-Anweisung angegeben wurde, konnte nicht geoeffnet werden, da der Zugriff nicht erlaubt wurde.
Fehlerschalter: EDT, DVS.

EDT5483 INVALID ACCESS TO MEMBER '(&00)' IN LIBRARY '(&01)'
EDT5483 UNGUELTIGER ZUGRIFF AUF ELEMENT '(&00)' IN BIBLIOTHEK '(&01)'

Bedeutung
Das Element (&00) der Bibliothek (&01), das in einer Ein-/Ausgabe-Anweisung angegeben wurde, konnte nicht geoeffnet werden, da der Zugriff nicht erlaubt wurde.
Fehlerschalter: EDT.

EDT5484 NO @FILE ENTRY DEFINED
EDT5484 KEIN @FILE-EINTRAG DEFINIERT

Bedeutung
In einer Ein-/Ausgabe-Anweisung ist kein Dateiname angegeben, und es ist weder ein lokaler noch ein globaler @FILE-Eintrag definiert.
Fehlerschalter: EDT.

EDT5485 INPUT TOO LONG
EDT5485 EINGABE ZU LANG

Bedeutung
Bei der Eingabe einer Anweisung mit indirekten Operanden uebersteigt die Gesamtlaenge aus Operation und der Zeichenfolge in der Zeichenfolge-variablen die maximale Satzlaenge von 32768 Zeichen.
Fehlerschalter: EDT.

EDT5486 WILDCARD SYMBOL MUST BE DIFFERENT TO QUOTE SYMBOLS
EDT5486 JOKERZEICHEN DARF KEIN BEGRENZERSYMBOL SEIN

Bedeutung
Ein Jokerzeichen (ASTERISK, SLASH) soll in ein Zeichen umdefiniert werden, das schon als Begrenzersymbol fuer Zeichenfolgen (siehe @QUOTE) verwendet wird. Die Anweisung @SYMBOLS wird nicht ausgefuehrt.
Fehlerschalter: EDT.

EDT5487 CCS '(&OO)' NOT SUPPORTED BY TERMINAL
EDT5487 CCS '(&OO)' VON DER DATENSICHTSTATION NICHT UNTERSTUETZT

Bedeutung
Der in der Anweisung @CODENAME ...,TERMINAL angegebene Kommunikations-Zeichensatz kann nicht eingestellt werden, da er von der verwendeten Datensichtstation nicht unterstuetzt wird.
Fehlerschalter: EDT.

EDT5488 TERMINAL NOT SUPPORTED
EDT5488 DATENSICHTSTATION NICHT UNTERSTUETZT

Bedeutung
Die verwendete Datensichtstation wird vom EDT (Unicode-Modus) nicht unterstuetzt.

EDT5489 FILES MUST BE DIFFERENT
EDT5489 DATEIEN MUESSEN VERSCHIEDEN SEIN

Bedeutung
Fuer die reale Bearbeitung soll die Kopie einer Datei geoeffnet werden (AS-Operand), fuer die Quelle und das Ziel ist aber die gleiche Datei angegeben.
Fehlerschalter: EDT.

EDT5490 ILLEGAL CCS NAME SPECIFIED
EDT5490 UNGUELTIGER CCS-NAME ANGEGEBEN

Bedeutung
An der Unterprogrammschnittstelle wurde ein ungueltiger Zeichensatzname angegeben.
Fehlerschalter: EDT.

EDT5491 XHCS MISSING OR HAS WRONG VERSION
EDT5491 XHCS NICHT VORHANDEN ODER HAT FALSCHE VERSION

Bedeutung
Das Subsystem XHCS ist nicht vorhanden oder es hat eine Version, die fuer den EDT nicht hinreichend ist. Der EDT kann nicht gestartet werden.
Fehlerschalter: EDT.

EDT5492 ILLEGAL RECORD FORMAT
EDT5492 SATZ-FORMAT UNGUELTIG

Bedeutung
Das Bibliothekselement enthaelt Format-B-Saetze. Diese koennen vom EDT nicht bearbeitet werden.
Fehlerschalter: EDT.

EDT5493 STRING CANNOT BE CONVERTED
EDT5493 ZEICHENFOLGE KANN NICHT KONVERTIERT WERDEN

Bedeutung
Einige Zeichen einer Zeichenfolge in der @ON-Anweisung sind im Zeichensatz der durchsuchten Zeile nicht definiert. Die Zeichenfolge kann nicht konvertiert werden. Die Anweisung wird abgebrochen.
Fehlerschalter: EDT.

EDT5494 'FORCE' OPERAND NOT ALLOWED
EDT5494 'FORCE'-OPERAND NICHT ERLAUBT

Bedeutung
Der Operand FORCE in der Anweisung @CODENAME ist fuer Unicode-Zeichensaetze nicht erlaubt.
Fehlerschalter: EDT.

EDT5495 NO FILES CORRESPONDING TO SPECIFIED OPERANDS
EDT5495 KEINE DATEI ENTSPRICHT DEN ANGEGEBENEN AUSWAHLKRITERIEN

Bedeutung
Fuer die in der @SHOW- oder @FSTAT-Anweisung angegebene Auswahl wurden keine Dateien gefunden.
Fehlerschalter: EDT.

```
EDT5496   INVALID VALUE FOR ATTRIBUTE (&00) IN TFT
EDT5496   UNGUELTIGER WERT FUER ATTRIBUT (&00) IN TFT
```

Bedeutung
Die ueber einen Dateikettungsnamen spezifizierte Datei kann nicht neu angelegt werden, weil das Attribut (&00) in der Task File Table einen ungueltigen Wert hat.
Fehlerschalter: EDT, DVS.

```
EDT5497   '(&00)' NOT POSSIBLE FOR PLAM ELEMENT TYPE '(&01)'
EDT5497   '(&00)' NICHT MOEGLICH FUER PLAM ELEMENT TYP '(&01)'
```

Bedeutung
Die Element-Typen R, C, H, L, U, F oder daraus abgeleitete freie Typen koennen nicht mit den Anweisungen @COPY, @OPEN, @WRITE oder @INPUT bearbeitet werden.
Fehlerschalter: EDT.

```
EDT5498   CANNOT WRITE TO SYSLST
EDT5498   AUSGABE NACH SYSLST NICHT MOEGLICH
```

Bedeutung
Beim Schreiben nach SYSLST ist ein Fehler aufgetreten, die Ausgabe wird abgebrochen. EDT-Fehlermeldungen werden jetzt in SYSOUT protokolliert.
Fehlerschalter: EDT.

```
EDT5499   WORK FILES MUST BE DIFFERENT
EDT5499   ARBEITSDATEIEN MUESSEN VERSCHIEDEN SEIN
```

Bedeutung
In der Anweisung @COMPARE wurde zweimal die gleiche Arbeitsdatei angegeben.
Fehlerschalter: EDT.

```
EDT5500   STATEMENT SEQUENCE PROCESSING INTERRUPTED BY USER
EDT5500   BEARBEITUNG DER ANWEISUNGSFOLGE VOM BENUTZER ABGEBROCHEN
```

Bedeutung
Eine Anweisungsfolge wurde im F-Modus-Dialog, im L-Modus-Dialog (Block-Modus) oder an der Unterprogrammschnittstelle eingegeben. Die Bearbeitung wurde vom Benutzer durch die Eingabe von /INFORM-PROGRAM bzw. /INTR abgebrochen.
Im F-Modus-Dialog wird der nicht verarbeitete Rest der Anweisungsfolge in der Kommandozeile ausgegeben.

```
EDT5501   STATEMENT '(&00)' INTERRUPTED BY USER
EDT5501   ANWEISUNG '(&00)' VOM BENUTZER ABGEBROCHEN
```

Bedeutung
Die Ausfuehrung der Anweisung (&00) wurde vom Benutzer durch die Eingabe von /INFORM-PROGRAM bzw. /INTR abgebrochen.

```
EDT5502   PROCEDURE INTERRUPTED BY USER
EDT5502   PROZEDUR VOM BENUTZER ABGEBROCHEN
```

Bedeutung
Die Ausfuehrung einer @DO- oder @INPUT-Prozedur wurde vom Benutzer durch die Eingabe von /INFORM-PROGRAM bzw. /INTR abgebrochen.

```
EDT5990   (&00)
EDT5990   (&00)
```

Bedeutung
Die Test-Routine meldet einen Fehler mit der Beschreibung (&00).
Fehlerschalter: EDT.

```
EDT5991   RUNTIME ERROR IN EXTERNAL STATEMENT
EDT5991   LAUFZEIT-FEHLER IN EXTERNER ANWEISUNG
```

Bedeutung
Die externe Routine meldet einen Laufzeitfehler in der Abarbeitung der angegebenen Anweisung ohne weitere Informationen.
Fehlerschalter: EDT.

```
EDT5999   (&00)
EDT5999   (&00)
```

Bedeutung
Die externe Routine meldet einen Laufzeitfehler in der Abarbeitung der angegebenen Anweisung mit der Beschreibung (&00).
Fehlerschalter: EDT.

```
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.

EDT8002 (&00) TO EDT UNSUCCESSFULLY. RETURNCODE = X'(&01)'
EDT8002 FEHLER BEI (&00) DES EDT. RETURNCODE = X'(&01)'

Bedeutung
Beim Nachladen des Moduls EDT trat ein Fehler auf. Das Makro (&00) lieferte den Fehlercode (&01).
Ist der Fehlercode = X'0C010104', so ist der Aufruf unter der Wartungskennung nur moeglich, wenn der EDT als Subsystem vorgeladen ist.

EDT8003 NO VIRTUAL MEMORY AVAILABLE
EDT8003 KEIN VIRTUELLER SPEICHER VERFUEGBAR

Bedeutung
Die Initialisierung des EDT konnte nicht durchgefuehrt werden, da nicht genuegend Speicher fuer die internen Daten des EDT vorhanden ist.

Maßnahme
Virtuellen Adressraum ueberpruefen.

EDT8005 ERROR ON EDT INITIALIZATION
EDT8005 FEHLER BEI INITIALISIERUNG DES EDT

Bedeutung
Bei der Initialisierung des EDT trat ein Fehler auf.

EDT8006 ERROR ON INSTALLATION OF EDT
EDT8006 FEHLERHAFTE INSTALLATION DES EDT

Bedeutung
Die Bestandteile der EDT-Installation sind inkonsistent oder der Start des EDT ist in der aktuellen Betriebssystem-Version nicht moeglich.

Maßnahme
EDT-Installation ueberpruefen und korrigieren.

EDT8100 EDT INTERRUPTED BY USER
EDT8100 EDT VOM BENUTZER UNTERBROCHEN

Bedeutung
Diese Meldung dient als Information fuer SDF-P-Prozeduren.
EDT ist geladen, wurde aber durch @SYSTEM (ohne Operand) oder durch K2 unterbrochen.

EDT8101 USER TERMINATED EDT ABNORMALLY
EDT8101 EDT VOM ANWENDER ABNORMAL BEENDET

Bedeutung
Der Anwender beendete den EDT mit der Anweisung @HALT ABNORMAL.

```
EDT8200   STXIT ROUTINE FOR RUNOUT ACTIVATED
EDT8200   STXIT ROUTINE FUER RUNOUT AKTIVIERT
```

Bedeutung
Das Ende der Programmlaufzeit wurde erreicht, deshalb wurde der EDT beendet.

```
EDT8292   UNRECOVERABLE READ ERROR. PROGRAM ABORTED
EDT8292   NICHT BEHEBBARER LESE-FEHLER. PROGRAMM-ABBRUCH
```

Bedeutung
Beim Lesen von SYSDTA oder vom Bildschirm ist ein nicht behebbarer Fehler aufgetreten. Der EDT wird beendet.

```
EDT8293   UNRECOVERABLE WRITE ERROR. PROGRAM ABORTED
EDT8293   NICHT BEHEBBARER SCHREIB-FEHLER. PROGRAMM-ABBRUCH
```

Bedeutung
Beim Schreiben nach SYSOUT ist ein nicht behebbarer Fehler aufgetreten. Der EDT wird beendet.

```
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)
```

Bedeutung
An der Adresse (&00) trat eine Programmunterbrechung aus der Ereignisklasse "Programmfehler" oder "Nicht behebbarer Programmfehler" auf.
Die naehere Ursache wird durch das Unterbrechungsgewicht (&01) angegeben.

Maßnahme
Systemverwalter verstaendigen.

14 Logistik

In diesem Abschnitt werden Voraussetzungen und Verfahren beschrieben, um den EDT V17.0A zu installieren und in Betrieb zu nehmen.

14.1 Softwarevoraussetzungen

EDT V17.0A ist für Betriebssystemversionen ab OSD-BC V6.0 freigegeben. Zur Unterstützung des Unicode-Modus müssen folgende Subsysteme im System installiert und gestartet sein:

XHCS-SYS	ab V2.0 A14
CRTE-BASYS	ab V1.6 A10
VTSU	ab V13.2 A05
TIAM	ab V13.1 C03
SYSFILE	ab V15.0 C00

Für einzelne Funktionen wird zusätzlich benötigt:

JV	ab V14.0, wenn mit Jobvariablen gearbeitet werden soll
SDF	ab V4.6, wenn die Anweisung @SDFTEST oder die Kurzanweisung T benutzt werden soll
SDF-P	ab V2.3, wenn mit S-Listenvariablen gearbeitet werden soll
POSIX	ab V6.0 A39, wenn POSIX-Dateien verarbeitet werden sollen

Ein komfortables Arbeiten mit Unicode-Dateien ist im Dialogbetrieb nur möglich, wenn eine Datensichtstation bzw. eine Terminal-Emulation benutzt wird, die Unicode-Zeichen verarbeiten kann, z.B. unter Windows die Emulation MT9750^{TM} ab V7.0.

14.2 Lieferumfang

Zum Produkt EDT V17.0A gehören folgende Release-Items:

Release-Item	Inhalt
SYSPRG.EDT.170.EDTU	Phase für EDT (Starten im Unicode-Modus)
SYSPRG.EDT.170	Phase für EDT (Starten im Kompatibilitäts-Modus)
SYSLNK.EDT.170	EDT-Modulbibliothek
SYSLNK.EDT.170.INIT	Bibliothek mit dem EDT-Initialisierungsmodul
SYSLIB.EDT.170	Benutzer-Makrobibliothek
SYSMES.EDT.170	Systemmeldungsdatei
SYSSSC.EDT.170	Subsystemdeklarationen für SSCM
SYSSII.EDT.170	Struktur- und Installationsinformation
SYSRMS.EDT.170	Korrekturdepot für RMS
SYSREP.EDT.170	REP-Datei
SYSFGM.EDT.170.D	Freigabe-Mitteilung (deutsch)
SYSFGM.EDT.170.E	Freigabe-Mitteilung (englisch)
SYSSDF.EDT.170	Syntaxdatei für SDF (START-EDT / START-EDTU)
SINPRC.EDT.170	Prozedur zum Installieren einer Privatversion
SYSSMB.EDT.170	Symbolinformation für die Diagnose mit DAMP

Hinweis

Die bisher ausgelieferten SYSNRF-Dateien sind nicht mehr erforderlich. Stattdessen werden die REPs für den EDT mit den entsprechenden REP-Kennzeichen versehen (S für *Selectable Unit* bzw. U für *Selectable Unit Optional*).

Die SYSACF-Datei zur Vergabe des Alias-Namens $.EDTLIB wird nicht mehr ausgeliefert. Anstatt die SYSACF-Datei in den System-Alias-Katalog einzumischen, lässt sich der gleiche Zweck mit dem Systemverwalter-Kommando

```
/ADD-ALIAS-CATALOG-ENTRY ALIAS-FILE-NAME=$.EDTLIB,FILENAME=$.SYLNK.EDT.170
```

erreichen.

Bei der Installation mit IMON wird die Phase für das Starten im Kompatibilitätsmodus (SYSPRG.EDT.170) unter dem Namen EDT in der Installationskennung abgelegt.

14.3 Produktstruktur

Die Datei SYSSII.EDT.170 enthält die Struktur und Installationsinformation. Den Produktbestandteilen werden logische Namen (*Logical ID*) zugeordnet, mit deren Hilfe über IMON der aktuelle Installationsort bestimmt werden kann.

Release Item	Logical ID
SYSPRG.EDT.170.EDTU	SYSPRG.EDTU
SYSPRG.EDT.170	SYSPRG
SYSLNK.EDT.170	SYSLNK
SYSLNK.EDT.170.INIT	SYSLNK.INIT
SYSLIB.EDT.170	SYSLIB
SYSMES.EDT.170	SYSMES
SYSSSC.EDT.170	SYSSSC
SYSSII.EDT.170	SYSSII
SYSRMS.EDT.170	SYSRMS
SYSREP.EDT.170	SYSREP
SYSFGM.EDT.170.D	SYSFGM.D
SYSFGM.EDT.170.E	SYSFGM.E
SYSSDF.EDT.170	SYSSDF
SINPRC.EDT.170	SINPRC
SYSSMB.EDT.170	SYSSMB
*DUMMY.EDTSTART	SYSDAT.EDTSTART

Die Modulbibliothek SYSLNK.EDT.170 enthält folgende Module:

Modul	Funktion	Unicode /Kompatibilitäts-Modus
EDTSTRT	Treiber für START–EDT und START–EDTU	U/K
EDTU	Großmodul für EDT	U
EDT	Großmodul für EDT	K
EDTXCODIR	Nachlademodul für CODE-Anweisung	K
EDTXPKEY	Nachlademodul für PKEY-Anweisung	K
EDTSSLNK	DSSM Nachlademodul	U/K
IEDTGLE	Binde-/Nachlademodul für EDT als Unterprogramm	U/K
CODTAB	Codiertabelle für CODE-Anweisung	K
EDTCON	Anschlussmodul für EDT, Umschalten der Betriebsmodi	U/K
EDCNATA	Nachlademodul für nationale Anwendung	K

Die Modulbibliothek SYSLNK.EDT.170.INIT enthält folgenden Modul:

Modul	Funktion	Unicode /Kompatibilitäts-Modus
IEDTCRT	Modul zur Initialisierung des EDT und zum Umschalten zwischen Unicode- und Kompatibilitäts-Modus.	U/K

Einzelheiten zu den Komponenten, die nur für den Kompatibilitäts-Modus erforderlich sind, werden an dieser Stelle nicht mehr erläutert. Sie finden sie im Handbuch EDT V16.6B [2].

14.4 Installation

Der EDT kann sowohl öffentlich mit Hilfe des SOLIS-Verfahrens installiert werden als auch von einem Benutzer als private Installation unter einer beliebigen Kennung vorgenommen werden.

14.4.1 Öffentliche Installation

Die Standardinstallation erfolgt durch das Verfahren SOLIS, das wiederum den Installationsmonitor IMON voraussetzt. Der EDT V17.0 kann nicht mehr in Systemen ohne IMON eingesetzt werden.

In bestehenden Programmen, die den EDT über die L-Modus Unterprogramm-Schnittstelle aufrufen, ist eventuell der Bibliotheksname $EDTLIB fest einprogrammiert. Derartige Programme können mit EDT V17.0 nur im Kompatibilitäts-Modus arbeiten. Damit diese Programme ohne Änderung ablauffähig sind, muss entweder eine Kopie der Bibliothek SYSLNK.EDT.170 unter dem Namen $TSOS.EDTLIB abgelegt werden oder es ist ein entsprechender Eintrag im systemglobalen Alias-Katalog zu machen (siehe Handbuch Systembetreuung [13] Abschnitt ACS - Alias-Katalog-System).

Die REP-Datei SYSREP.EDT.170 muss mehrbenutzbar (USER-ACCESS=SPECIAL) sein. Nur dann werden die Korrekturen beim Nachladen des EDT mitgeladen.

Aus Performancegründen kann der EDT als Subsystem geladen werden (siehe unten).

Start-Prozedur EDTSTART

Für jede öffentlich installierte EDT-Version kann über IMON eine eigene, auf allen Kennungen geltende EDT-Startprozedur installiert werden. Der Installationsort der Prozedurdatei kann frei gewählt werden (siehe auch Abschnitt EDT-Startprozedur). 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 bekannt gegeben 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 EDT-Startprozedur $.EDTSTART, sofern diese existiert.

EDT als Subsystem

Der EDT besteht aus drei Subsystemen: EDTCON, EDTU und EDT. Alle EDT-Subsysteme sind adressmodusunabhängig, laufen also sowohl im 24-Bit-Adressierungsmodus als auch im 31-Bit-Adressierungsmodus. Die Entscheidung, ob ein Subsystem nach oben oder unten geladen werden soll, ergibt sich daher aus der Überlegung, in welchem Adressierungsmodus die häufigsten Nutzer der EDT-Unterprogramm-Schnittstelle ablaufen.

Das Subsystem EDTCON (bestehend aus den Modulen EDTCON, IEDTGLE und EDTSSLNK) wird im unteren Adressraum geladen. Der Modul EDTCON stellt im EDT V17.0A nur noch einen Adapter dar, der die bisherigen Entries des EDT in kompatibler Form anbietet und den eigentlichen Initialisierungsmodul IEDTCRT aus der Bibliothek SYSLNK.EDT.170.INIT in den Adressraum des Benutzers nachlädt. Allerdings wird der Modul EDTCON nicht mehr als OM ausgeliefert sondern als LLM.

Die Subsysteme EDTU und/oder EDT können im oberen Adressraum geladen werden. Das Subsystem EDTU enthält die Module, die beim Ablauf des EDT im Unicode-Modus benötigt werden, das Subsystem EDT die äquivalenten Module für den Kompatibilitäts-Modus. Je nach Einsatz-Szenario kann es sinnvoll sein, nur eines der beiden Subsysteme vorzuladen. Beim Umschalten des Betriebsmodus würde dann das jeweils andere Subsystem als private Kopie nachgeladen werden. Das Vorladen dieser beiden Subsysteme ist aber nur möglich, wenn das Subsystem EDTCON zuvor vorgeladen wurde.

Soll der EDT als Haupt- oder Unterprogramm im 24-Bit-Adressierungsmodus ablaufen, wird vom Modul IEDTCRT entweder die Verbindung zu einem unten geladenen EDT hergestellt oder der EDT privat im unteren Adressraum nachgeladen.

In IEDTCRT ist auch der Mechanismus zum Umschalten der Betriebsmodi des EDT realisiert. D.h. je nach Aufruf des EDT als Hauptprogramm über /START-EDTU oder /START-EDT bzw. nach der verwendeten Version der Unterprogramm-Schnittstelle wird in das Subsystem EDTU (Unicode-Modus) oder in das Subsystem EDT (Kompatibilitäts-Modus) verzweigt. Beim expliziten oder impliziten Wechsel des Betriebsmodus durch Benutzereingaben werden ebenfalls Funktionen von IEDTCRT benutzt.

Das Subsystem EDTCON kann mit

```
/START-SUBSYSTEM SUBSYSTEM-NAME=EDTCON,SYNC=*YES
```

zuerst vorgeladen werden. Danach können die Subsysteme EDTU und/oder EDT mit

```
/START-SUBSYSTEM SUBSYSTEM-NAME=EDTU
```

bzw.

```
/START-SUBSYSTEM SUBSYSTEM-NAME=EDT
```

in beliebiger Reihenfolge gestartet werden.

14.4.2 Private Installation

Die Datei SINPRC.EDT.170 enthält eine Prozedur, mit deren Hilfe auf einer beliebigen Benutzerkennung eine Privatversion installiert werden kann. Eine Privatversion sollte nur zu Testzwecken installiert werden und nicht zur Koexistenz von zwei EDT-Versionen.

Vor dem Aufruf dieser Prozedur müssen mindestens folgende Dateien auf der Installationskennung des privaten EDT bereitgestellt werden:

- SYSPRG.EDT.170 und SYSPRG.EDT.170.EDTU (Starterphasen)
- SYSLNK.EDT.170 (Modulbibliothek)
- SYSLNK.EDT.170.INIT (Modulbibliothek mit EDT-Initialisierungsroutine)
- SYSREP.EDT.170 (REP-Datei)

Aus Kompatibilitätsgründen wird auch eine Datei namens EDT als Starterphase für den Kompatibilitätsmodus verwendet (wie in einer öffentlichen Installation), wenn keine Datei SYSPRG.EDT.170 auf der Installationskennung vorhanden ist. Empfohlen wird aber, den Namen SYSPRG.EDT.170 zu verwenden, da nur so mehrere Privatinstallationen verschiedener EDT-Versionen auf einer Installationskennung möglich sind.

Die Prozedur kann von der Installationskennung oder von TSOS aufgerufen werden:

```
/CALL-PROCEDURE FROM-FILE=SINPRC.EDT.170, -
/PROCEDURE-PARAMETERS=(USERID=userid,ORIG=Y|N, -
/EDT=edt,EDTU=edtu,EDTLIB=edtlib,INILIB=inilib,
/REPFILE=repfile)
```

Die Prozedurparameter können direkt oder über *Parameter-Prompting* angegeben werden. Die Parameter EDT, EDTU, EDTLIB, INILIB und REPFILE müssen nur angegeben werden, wenn ORIG=N spezifiziert wurde.

userid	Installationskennung unter der die private Installation des EDT erfolgen soll.
ORIG=Y	legt fest, dass die bereit gestellten Originaldateien (unter der Installationskennung) verändert werden sollen.
ORIG=N	legt fest, dass Kopien der bereit gestellten Originaldateien angelegt werden sollen und die Veränderungen nur an den Kopien erfolgen.
edt	Name der Kopie für die private Starterphase für den Kompatibilitäts-Modus (falls nicht auf Originaldateien gearbeitet wird).
edtu	Name der Kopie für die private Starterphase für den Unicode-Modus (falls nicht auf Originaldateien gearbeitet wird).
edtlib	Name der Kopie für die private Modulbibliothek (falls nicht auf Originaldateien gearbeitet wird).
inilib	Name der Kopie für die private Modulbibliothek mit dem EDT-Initialisierungsmodul (falls nicht auf Originaldateien gearbeitet wird).
repfile	Name der Kopie für die private REP-Datei für den EDT (falls nicht auf Originaldateien gearbeitet wird).

Die private EDT-Version kann mit dem Kommando

```
/START-PROGRAM FROM-FILE=$userid.name-private-starterphase
```

gestartet werden.

Durch die Installations-Prozedur werden für die private EDT-Version alle Bezüge auf die in IMON definierten logischen Namen stillgelegt. Dies gilt insbesondere auch für die eventuell definierte Zuordnung zu SYSDAT.EDTSTART. Eine darüber zentral definierte EDT-Startprozedur wird also von der privaten Version nicht ausgeführt.

Programme, die diese private EDT-Version als Unterprogramm aufrufen wollen, müssen entweder das Anschlussmodul IEDTGLE aus der erstellten Modulbibliothek fest einbinden oder mit Hilfe des ENTRY-Namens IEDTGLE dynamisch nachladen. Dabei ist unbedingt im Makro BIND der Parameter SHARE=NO anzugeben.

Programme, die die private EDT-Version als Unterprogramm aufrufen wollen und den EDT mit Hilfe des Makro-Aufrufs BIND (nicht LINK) aus $.EDTLIB nachladen, können eine private Modulbibliothek mit folgendem Kommando zuweisen:

```
/SET-FILE-LINK LINK-NAME=BLSLIBnn,FILE-NAME=bibliothek
```

wobei nn=00..99 ist. Dieses Verfahren ist nur möglich, falls der EDT nicht als Subsystem geladen ist.

Fachwörter

@DO-Prozedur

Eine @DO-Prozedur ist eine EDT-Prozedur, die in einer Arbeitsdatei gespeichert ist. Sie kann mit der Anweisung @DO zum Ablauf gebracht werden. In @DO-Prozeduren stehen einige Anweisungen zur Ablaufsteuerung zur Verfügung. Bei ihrem Aufruf können Parameter übergeben werden und sie können geschachtelt werden.

@INPUT-Prozedur

Eine @INPUT-Prozedur ist eine EDT-Prozedur, die in einer Datei oder in einem Bibliothekselement gespeichert ist. Sie kann mit der Anweisung @INPUT zum Ablauf gebracht werden. In @INPUT-Prozeduren stehen die Anweisungen zur Ablaufsteuerung nicht (direkt) zur Verfügung, sie können nicht ineinandergeschachtelt werden und es können keine Parameter übergeben werden. Es können aber @DO-Prozeduren aufgerufen werden.

Anweisung

Eingaben an den EDT sind entweder Datensätze oder Anweisungen. Mit Anweisungen können Funktionen des EDT ausgelöst bzw. Einstellungen vorgenommen werden. Um Anweisungen von Datensätzen unterscheiden zu können, müssen Anweisungen im Line-Modus mit dem EDT-Anweisungssymbol eingeleitet werden. Im F-Modus werden Anweisungen in der Anweisungszeile eingegeben. Daneben gibt es noch die Kurzanweisungen, die in der Kurzanweisungsspalte eingegeben werden.

Anweisungspuffer

Der EDT speichert die letzten im F-Modus eingegebenen Anweisungen in einem Puffer, aus dem diese wieder zurückgeholt werden können.

Anweisungszeile

Ein Feld des Arbeitsfensters im F-Modus. Eingaben in der Anweisungszeile werden als Anweisungen interpretiert. Das EDT-Anweisungssymbol kann normalerweise weggelassen werden.

Arbeitsdatei

Eingabe und Bearbeitung von Daten geschieht im EDT immer in einer Arbeitsdatei. In Arbeitsdateien können z.B. Daten eingefügt, geändert und gelöscht werden. Der Inhalt der Arbeitsdateien kann am Bildschirm angezeigt werden. Sollen Inhalte einer Datei (DVS-Datei, Bibliothekselement oder POSIX-Datei) bearbeitet werden, müssen sie zunächst in eine Arbeitsdatei übernommen werden. Nach der Bearbeitung kann der Inhalt einer Arbeitsdatei in eine Datei geschrieben werden.
Der EDT kann 23 Arbeitsdateien verwalten. Die Arbeitsdateien sind in Sätzen organisiert, denen eine Zeilennummer zugeordnet ist.

Arbeitsdatei, aktuelle

Eine Arbeitsdatei ist die aktuelle Arbeitsdatei. In ihr werden Daten eingegeben und Anweisungen wirksam. Im F-Modus wird ein Ausschnitt der aktuellen Arbeitsdatei am Bildschirm angezeigt.

Arbeitsdatei, aktive

Eine aktive Arbeitsdatei ist eine Arbeitsdatei, die eine @DO-Prozedur enthält, die gerade ausgeführt wird. Wenn @DO-Prozeduren geschachtelt werden, können mehrere Arbeitsdateien aktiv sein.

Arbeitsdatei, leere

Eine **leere** Arbeitsdatei ist eine Arbeitsdatei, die keine Sätze enthält. Auch eine leere Arbeitsdatei kann noch Eigenschaften besitzen, die nicht dem Initialzustand entsprechen, z.B. kann sie noch als belegt gelten oder mit einer Datei verknüpft sein. Erst mit einer Anweisung @DELETE (Format 2) oder mit anderen Anweisungen, die implizit oder explizit Arbeitsdateien vollständig löschen, wird die Arbeitsdatei wieder in den Initialzustand versetzt.

Arbeitsfenster

Im F-Modus wird die aktuelle Arbeitsdatei am Bildschirm dargestellt. Der Bildschirm wird dabei in Felder mit unterschiedlicher Funktion aufgeteilt. Neben dem Datenfenster, in dem der Inhalt der aktuellen Arbeitsdatei dargestellt wird, enthält das Arbeitsfenster u.a. noch eine Anweisungszeile und eine Kurzanweisungsspalte.

Arbeitsmodus

Im EDT stehen zwei Arbeitsmodi für die Bearbeitung von Daten zur Verfügung, der Line-Modus (L-Modus) und der Full-Screen-Modus (F-Modus).
Im L-Modus wird jeweils nur eine Bildschirmzeile zur Eingabe von Daten und Anweisungen angeboten.
Im F-Modus steht der gesamte Bildschirm für die Eingabe von Daten und Anweisungen zur Verfügung.
Die Arbeitsmodi sind nicht zu verwechseln mit den Betriebsmodi des EDT (Kompatibilitäts-Modus und Unicode-Modus). Während die Arbeitsmodi sich auf die Unterschiede bei der Darstellung und der Arbeit mit den Daten beziehen, stellen die Betriebsmodi unterschiedliche EDT-Umgebungen mit eingeschränktem bzw. erweitertem Funktionsumfang dar.

Begrenzersymbole

Literale werden normalerweise durch das Begrenzersymbol `apostrophe` (Standardwert ') eingeschlossen. Beim Suchen mit der Anweisung @ON kann auch ein besonderes Begrenzersymbol verwendet werden, das `quotation mark` (Standardwert "). Damit wird festgelegt, dass eine Zeichenfolge nur dann als Treffer erkannt wird, wenn sie durch Textbegrenzerzeichen begrenzt ist. Die Textbegrenzerzeichen sind eine einstellbare Menge von Zeichen, zu denen Standardmäßig das Leerzeichen, die runden Klammern u.a. gehören.

Benutzeranweisungssymbol

Ein Benutzeranweisungssymbol ist ein spezielles Zeichen, mit dem Benutzeranweisungen gekennzeichnet werden, die durch externe Anweisungsroutinen ausgeführt werden.

Betriebsarten

Es werden zwei Betriebsarten unterschieden, je nach dem ob eine Datensichtstation vorhanden ist oder nicht. Im Dialogbetrieb ist eine Datensichtstation vorhanden, im Stapelbetrieb nicht.

Betriebsmodus

Da die notwendigen Erweiterungen für die Unicode-Unterstützung des EDT nicht kompatibel erfolgen konnten, wurde ein neuer Betriebsmodus des EDT eingeführt. EDT V17.0A kann also in 2 Modi betrieben werden, dem Unicode-Modus und dem Kompatibilitäts-Modus.
Im Unicode-Modus steht eine Reihe von Erweiterungen zur Verfügung. Vor allem können (nur) im Unicode-Modus Unicode-Dateien bearbeitet werden. Allerdings ist dieser Modus nicht in allen Punkten kompatibel zu EDT V16.6B. Der Kompatibilitäts-Modus gewährleistet dagegen die volle Funktionalität des EDT V16.6B, dafür stehen allerdings die Erweiterungen nicht zur Verfügung. EDT V17.0A wird standardmäßig im Kompatibilitäts-Modus gestartet. Mit einer neuen Anweisung kann in den Unicode-Modus umgeschaltet werden.

Bildschirmdialog

Mit der Anweisung @DIALOG, die nur an der Unterprogrammschnittstelle bzw. von SYSDTA eingegeben werden kann, kann der EDT in den Bildschirmdialog versetzt werden. Im Bildschirmdialog ist der bisherige Lesevorgang unterbrochen und der EDT liest seine Eingaben im F-Modus (oder im L-Modus nach Eingabe von @EDIT) von der Datensichtstation. Der Bildschirmdialog kann mit @HALT, @END, @RETURN oder K1 wieder verlassen werden. Der EDT setzt dann den unterbrochenen Lesevorgang fort.

Bildschirmzeile

Zeilen des Datenfensters, in denen die Sätze der aktuellen Arbeitsdatei angezeigt werden.

Datenfenster

Feld des Arbeitsfensters, in dem die aktuelle Arbeitsdatei angezeigt wird. Die Sätze der Arbeitsdatei werden in die Bildschirmzeilen des Datenfensters ausgegeben.

Dialogbetrieb

Der Dialogbetrieb ist die Betriebsart des EDT, bei der eine Datensichtstation vorhanden ist. Nur in dieser Betriebsart kann der EDT im F-Modus arbeiten.

EDT-Anweisungssymbol

Das EDT-Anweisungssymbol ist das spezielle Zeichen, mit dem Anweisungen gekennzeichnet werden. Standardmäßig ist dies das Zeichen @, das daher in diesem Dokument immer verwendet wird.

EDT-Prozeduren

Eine EDT-Prozedur ist eine Folge von Eingaben, Anweisungen und/oder Datensätzen, an den EDT, die in einer Datei (@INPUT-Prozedur) bzw. einer Arbeitsdatei (@DO-Prozedur) abgelegt sind.

EDT-Startprozedur

Die EDT-Startprozedur ist eine spezielle @INPUT-Prozedur, die (falls vorhanden) beim Start des EDT ausgeführt wird.

EDT-Variablen

EDT-Variablen sind Behälter, in denen Werte auch über Arbeitsdateien hinweg gespeichert werden können. EDT-Variablen sind nur während des EDT-Laufs gültig.
Es gibt drei Arten von Variablen, die mit entsprechenden Werten versorgt werden können. Von jeder Variablenart stehen 21 Variablen zur Verfügung:

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

Ersatzzeichen

Wenn Zeichenfolgen aus einem Zeichensatz in einen anderen Zeichensatz konvertiert werden, kann der Fall eintreten, dass Zeichen aus den Quell-Daten im Ziel-Zeichensatz nicht vorhanden sind. Stattdessen wird dann das Ersatzzeichen eingesetzt, falls es definiert wurde. Wenn kein Ersatzzeichen definiert wurde, wird die Konvertierung normalerweise abgelehnt.

Full-Screen-Modus (F-Modus)

Der Full-Screen-Modus (F-Modus) ist ein Arbeitsmodus des EDT. Im F-Modus steht der gesamte Bildschirm in Form eines Arbeitsfensters für die Eingabe von Daten und Anweisungen zur Verfügung. Es besteht die Möglichkeit vom F-Modus in den L-Modus umzuschalten. Der EDT kann nur im F-Modus arbeiten, wenn er im Dialogbetrieb ist.

Kompatibilitäts-Modus

Der Kompatibilitäts-Modus ist ein Betriebsmodus des EDT V17.0A. Im Kompatibilitäts-Modus ist die volle Funktionalität des EDT V16.6B gewährleistet. Allerdings können die erweiterten Funktionen des EDT V17.0A nicht genutzt werden z.B. erlaubt er nicht die Bearbeitung von Unicode-Dateien und die Satzlänge ist weiterhin auf 256 Byte beschränkt.
Ein Wechsel in den Unicode-Modus erfolgt unter gewissen Voraussetzungen implizit, wenn eine Unicode-Datei eingelesen wird, oder explizit, wenn eine @MODE-Anweisung eingegeben wird.

Kommunikations-Zeichensatz

Zeichensatz, in dem der EDT im Unicode-Modus mit der Datensichtstation kommuniziert. Dies kann ein anderer sein als der Zeichensatz der aktuellen und auch jeder anderen Arbeitsdatei. Der Kommunikations-Zeichensatz ist normalerweise optimal an die Möglichkeiten der Datensichtstation angepasst.

Kurzanweisung

Kurzanweisungen sind 1 Zeichen lange Anweisungen, die im F-Modus in der Kurzanweisungsspalte eingegeben werden können.

Kurzanweisungsspalte

Die Kurzanweisungsspalte ist ein Feld des Arbeitsfensters, in dem Kurzanweisungen eingegeben werden können.

Line-Modus (L-Modus)

Der Line-Modus (L-Modus) ist ein Arbeitsmodus des EDT. Im L-Modus erfolgt die Dateibearbeitung zeilenorientiert, d.h. der EDT bietet im Dialogbetrieb nur eine Zeile (die aktuelle Zeile) zur Eingabe an bzw. liest (im Stapel- und im Dialogbetrieb) jeweils eine Zeile von `SYSDTA`. Diese Zeile kann entweder Datensätze oder Anweisungen enthalten und wird nach dem Einlesen sofort verarbeitet.

Musterzeichen

Musterzeichen sind Platzhalter für Zeichengruppen in einem Suchstring. Dabei steht `asterisk` (Standardwert `*`) für eine beliebig lange, auch leere Zeichenfolge, `slash` (Standardwert `/`) steht für genau ein Zeichen.

Satzmarkierungen

Jeder Satz einer Arbeitsdatei kann mit Satzmarkierungen gekennzeichnet werden, die für den Benutzer nicht sichtbar sind. Sie können mit Anweisungen und Kurzanweisungen gesetzt, abgefragt und gelöscht werden. Markierte Sätze können dann z.B. kopiert oder gelöscht werden.

Stapelbetrieb

Der Stapelbetrieb ist die Betriebsart des EDT, wenn keine Datensichtstation vorhanden ist. Der EDT kann dann nur im L-Modus arbeiten.

Unicode-Ersatzdarstellung

Der EDT erlaubt in Anweisungen innerhalb von Literalen, aber auch in Daten, über eine Ersatzdarstellung, Unicode-Zeichen durch die Angabe des Hex-Wertes ihres `UTF16` Codes anzugeben. Damit können über ein Fluchtsymbol alle (unterstützten) Zeichen eingegeben werden, auch wenn der Zeichensatz der Anweisung bzw. der Daten kein Unicode-Zeichensatz ist.

Unicode-Modus

Der Unicode-Modus ist ein Betriebsmodus des EDT. Nur im Unicode-Modus stehen die erweiterten Funktionen des EDT V17.0A zur Verfügung, d.h. nur in diesem Modus können Unicode-Dateien bearbeitet werden, können Sätze länger als 256 Byte sein, gibt es lokale Zeichensätze usw.
Allerdings ist dieser Modus nicht in allen Punkten kompatibel zu EDT V16.6B. Insbesondere gibt es Inkompatibilitäten an der Unterprogrammschnittstelle. Außerdem ist eine Reihe von Inkonsistenzen bereinigt worden.

Zeichensatz

Mit dem EDT V17.0A können Texte in allen Zeichensätzen bearbeitet werden, die XHCS bereitstellt. Zusätzlich zu den in EDT V16.6B bzw. im Kompatibilitätsmodus unterstützten sind dies im Unicode-Modus die 3 Unicode-Zeichensätze, ISO-Zeichensätze und zusätzlich 7-Bit-Zeichensätze.
In jeder Arbeitsdatei kann ein anderer Zeichensatz eingestellt werden, so dass also Texte in unterschiedlichen Zeichensätzen parallel bearbeitet werden können. Für die Kommunikation mit einer Datensichtstation wird ein Kommunikations-Zeichensatz eingestellt.

Zeilennummer

Jedem Satz einer Arbeitsdatei ist eine Zeilennummer zugeordnet, durch die er eindeutig gekennzeichnet ist. Eine Zeilennummer ist die aktuelle Zeilennummer, in die im L-Modus die Dateneingabe erfolgt.

Literatur

Die Handbücher sind online unter *http://manuals.fujitsu-siemens.com* zu finden oder in gedruckter Form gegen gesondertes Entgelt unter *http://FSC-manualshop.com* zu bestellen.

[1] **EDT V17.0A UNICODE-Modus** (BS2000/OSD)
Unterprogrammschnittstellen
Benuterhandbuch

[2] **EDT V16.6B** (BS2000/OSD)
Anweisungen
Benutzerhandbuch

[3] **EDT V16.6** (BS2000/OSD)
Unterprogrammschnittstellen
Benutzerhandbuch

[4] **EDT-ARA** (BS2000/OSD)
Additional Information for Arabic
User Guide

[5] **EDT-FAR** (BS2000/OSD)
Additional Information for Farsi
User Guide

[6] **SDF** (BS2000/OSD)
Einführung in die Dialogschnittstelle SDF
Benutzerhandbuch

[7] **SDF-P** (BS2000/OSD)
Programmieren in der Kommandosprache
Benuterhandbuch

[8] **XHCS** (BS2000/OSD)
8-bit-Code-Verarbeitung im BS2000/OSD)
Benutzerhandbuch

[9] **JV** (BS2000/OSD)
Jobvariablen
Benuterhandbuch

[10] **BS2000/OSD-BC**
Kommandos Band 1-5
Benutzerhandbuch

[11] **BS2000/OSD-BC**
Kommandos Band 6, Ausgabe in S-Variablen und SDF-BASYS
Benutzerhandbuch

[12] **BS2000/OSD-BC**
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

[13] **BS2000/OSD-BC**
Einführung in die Systembetreung
Benutzerhandbuch

[14] **LMS** (BS2000/OSD)
SDF-Format
Benutzerhandbuch

[15] **POSIX** (BS2000/OSD)
Grundlagen für Anwender und Systemverwalter
Benutzerhandbuch

[16] **POSIX** (BS2000/OSD)
Kommandos
Benutzerhandbuch

[17] **ASSEMBH** (BS2000/OSD)
Beschreibung

[18] **ASSEMBH** (BS2000/OSD)
Benutzerhandbuch

Stichwörter

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.