EDT für WINDOWS

Benutzerhandbuch

Ausgabe Oktober 2023 (EDT V 4.39)

Änderungsprotokoll

Im Folgenden sind die Erweiterungen des Programms EDT zusammengestellt.

Mai 2022 Verschlüsselung: Extras -> Filetransfer... -> FT-Profil (S. **142**): Die integrierte openFT Schnittstelle unterstützt jetzt auf Anforderung auch die verschlüsselte Kommunikation und Dateiübertragung. Für Verbindungen mit FTP stehen nun auch die Schnittstellen FTP-S und S-FTP zur Verfügung.

März 2020 Kommando `SDFTEST` (S. **244**)**:** Die Online-Syntaxprüfung des Kommandos SDFTEST überprüft nun auch die //Programm-Anweisungen von den Fujitsu Standard-Programmen BINDER, BS2ZIP, HSMS, JMP, JMU, LMS, MAREN, PAMCONV, PERCON, PVSREN, SIR, SORT und SSCM. Software-Voraussetzung: ab OSD V11.0 SP19.1. Die Prüfung der //-Statements kann im Option-Dialog Extras->Highlighting... (S. **149**) auch deaktiviert werden.

Mai 2018 Das Highlighting von SDF-Prozeduren beschränkt sich nicht mehr auf die /BS2000-Kommandos sondern wurde auch auf die //Programm-Anweisungen von diversen STD-Programmen ausgeweitet. Unterstützt werden momentan die Programme: BINDER, BS2ZIP, HSMS, JMP, JMU, LMS, MAREN, PAMCONV, PERCON, PVSREN, SIR, SORT und SSCM. Die Programmeingaben an diese Programme werden zur besseren Unterscheidung mit "leicht schattiertem Hintergrund" optisch abgegrenzt.

Nov 2017 Baumfenster der Arbeitsbereiche - SDF-OUTLINE Struktur-Ausgabe: unterhalb des Arbeitsbereiches werden Struktur-Informationen (LABELs, COMMANDs, STRUCTURE, COMMENTs,...) der eingelesenen Prozedur ausgegeben. Durch Klicken auf einen Eintrag im Outline-Tree, wird an die entsprechende Stelle in der Arbeitsebene positioniert.

Kommando `IF` (S. **280**): @INPUT und @DO sind nun auch als direkte Sub-Kommandos zugelassen.

Kommando `INPUT` (S. **206**): die Option PRINT wird bei geschachtelten INPUT-Prozeduren vererbt.

Kommando `WINGEN button,listbox,table`: als Verarbeitung kann nun neben den Varianten für bereits bestehende Ebenen PROC*n* und PROC*int-var* über INPUT=*'file'* auch direkt die Verarbeitung einer INPUT-Prozedur ausgelöst werden.

April 2017 Extras -> Filetransfer... -> Filetransfer-Profil (S. **142**): Unterstützung des FTP Passive-Mode beim Zugriff auf Remote-Dateien.

Baumfenster der Arbeitsbereiche - ASSEMBLER-OUTLINE Struktur-Ausgabe: unterhalb des Arbeitsbereiches werden Struktur-Informationen (CSECTs, MACRO Calls, USINGs, LABELs,...) der eingelesenen Datei ausgegeben. Durch Klicken auf einen Eintrag im Outline-Tree, wird an die entsprechende Stelle in der Arbeitsebene positioniert.

Mai 2016 Erweiterte Möglichkeiten bei der Verwendung von Programmtasten: neben der Kombination F8 + A-Z stehen nun 10 weitere Programmtasten (S. **133**) zur individuellen Belegung zur Verfügung, die über die Kombination F8 + 0-9 abgerufen werden können.

Dez 2015 Kommando `SDFTEST` (S. **244**)**:** neuer Operand P=*ft-profil*. Damit kann für die online Syntaxprüfung ein beliebiges openFT Filetransfer-Profil verwendet werden.

Kommando `LIST` (S. **207**)**:** neuer Operand A für Dateien, die im ersten Byte ASA-Drucksteuerzeichen enthalten.

Baumfenster der Arbeitsbereiche - COBOL-OUTLINE Struktur-Ausgabe: unterhalb des Arbeitsbereiches wird die hierarchische Struktur (DIVISIONs, SECTIONs,...) der eingelesenen Datei ausgegeben. Durch Klicken auf einen Eintrag im Outline-Tree, wird an die entsprechende Stelle in der Arbeitsebene positioniert.

Änderungsprotokoll

Mai 2015

Syntax-Highlighting: als zusätzliche neue Sprache steht nun auch C/C++ zur Verfügung. Die Sprache wird automatisch anhand der Inhalte oder der Dateiendung erkannt. Das Highlighting kann auch mit dem Kommando SHL (S. **247**), dem Menübefehl Ansicht/Syntax-Highlighting (S. **96**) oder mit der Schaltfläche der Toolbar (S. **164**) aktiviert werden. Die Farben können über den Dialog Extras/Highlighting (S. **149**) eingestellt werden.

User-Toolbar: Der Aufruf von externen Programmen ist nun auch ohne automatische Speicherung und Übergabe des lokalen Dateinamens möglich. Dafür steht im Feld Parameter bei externen Verknüpfungen der Sonderstring *EXE zur Verfügung. Menü Extras/User-Toolbar (S. **157**) .

Kommando `INPUT` (S. **206**)**:** Der Ablauf von INPUT-Prozeduren findet nun generell im Silent-Mode (verborgen) statt, d.h. die intern angelegten Arbeitsbereiche sind unsichtbar.

Neues Kommando `MARK` (S. **209**)**:** Das Kommando MARK bietet die Möglichkeit, bestimmte Zeilen zu kennzeichnen und mit Hinweisen zu versehen.

Neues Kommando `SDFTEST` (S. **244**)**:** Das Kommando SDFTEST überprüft eine BS2000-Prozedur im aktuellen Arbeitsbereich online auf Syntax- und Strukturfehler. Dazu muss ein openFT Filetransfer-Profil mit dem Namen SDFTEST eingerichtet werden. Die fehlerhaften Zeilen werden entsprechend hervorgehoben.

Februar 2015

User-Toolbar: Definition einer individuellen Toolbar mit der Möglichkeit externe Programme oder eine interne EDTW-Prozedur für die Daten in der aktuellen Arbeitsebene aufzurufen. Die Schaltflächen werden über das Menü Extras → User-Toolbar (S. **157**) definiert.

EDT-Parameterersetzung: einige neue Variablen zur Stringersetzung im Zusammenhang mit den User-Buttons. z.B. !ActLine (Zeile, in der sich der Cursor befindet), !ActCol (Spalte, in der sich der Cursor befindet), !Font, !FontSize, ...). Mehr Details siehe S. **344**.

November 2014

openFT: Beginnt das Kommando in den Feldern Local Success bzw. Local Failure des Filetransfer-Profils (S. **140**) mit dem Zeichen "@", wird das nachfolgende EDT-Kommando ausgeführt. Es ist auch zulässig, mit dem Kommando INPUT eine EDT-Prozedur aufzurufen.

Syntax-Highlighting: Der Assembler-Code in Compiler-Listen wird automatisch erkannt und wie in Assembler-Sourcen entsprechend farblich gekennzeichnet.

Oktober 2014

Kommando `DROP` (S. **189**): Mit der neuen Option F kann ein Arbeitsbereich auch außerhalb der Ebene 0 freigegeben werden.

Kommando `INPUT` (S. **206**)**:** Die geschachtelte Nutzung von dem Kommando `INPUT` wird nun unterstützt. Dadurch entfällt in vielen Fällen das dynamische Aufbauen von Prozeduren.

Kommando `IF` (S. **280**) und `GOTO` (S. **279**): Diese Kommandos Können nun auch in INPUT-Prozeduren in der Ebene 0 benutzt werden. Bisher konnten diese Kommandos nur in Prozedurbereichen genutzt werden, die dann mit dem Kommando DO aufgerufen werden mussten.

Neues Kommando #S*nn* (S. **177**): Inhalt einer String-Variablen als Kommando ausführen.

Beim Starten des EDTW werden die Standard-Prozeduren (S. **35**) EDTINITCMD und EDTINIT automatisch ausgeführt. Die Prozedurdateien müssen sich im Ladeverzeichnis und/oder im APPDATA-Verzeichnis befinden.

WINGEN: Bei der Angabe der Koordinaten (x,y,w,h) in den @wingen Anweisungen ist nun auch die Verwendung von Integer-Variablen erlaubt. Zusätzlich ist auch der Bezug auf die Breite und die Höhe eines anderen Objekts sowie komplexe Rechenoperationen zulässig (S. **310**)

Kommando WINGEN FONT (S. 315): Als Fontgröße kann nun auch eine Integervariable angegeben werden.

Kommando READ fern (S. 226): Die höchste Versionsnummer eines Bibliothekelements wurde bisher nicht angezeigt. Diese wird nun ermittelt, im Reiter angezeigt und auch für eventuell folgende Schreibaktionen verwendet.

Juli 2014 — Beim Drucken von Arbeitsbereichen mit aktiviertem Syntax-Highlighting werden die Farbinformationen an den Drucker weitergegeben, so dass mit einem Farbdrucker farbige Ausdrucke bzw. bei Schwarz-Weiß-Druckern Ausdrucke mit den entsprechenden Grautönen erstellt werden können.

Mai 2014 — MY-Profile: Definition einer individuelle Arbeitsumgebung durch neue Registerkarte im Navigationsfenster (Einrichtung von Projekten, automatische Verarbeitung nach dem Lesen und Schreiben von Dateien). Die Profile werden über das Menü Extras → MY-Profile (S. 152) definiert.

Juli 2013 — Da die bestehende EDTW.INI im Installationsverzeichnis nicht überschrieben werden darf, weil evtl. Einstellungen geändert wurden, werden neue Parameter mit den Standardwerten in einer zusätzlichen INI-Datei EDTW-OPG.INI zur Verfügung gestellt.

Juni 2013 — Syntax-Highlighting: Farbige Darstellung der wichtigen Elemente der Programmiersprachen Assembler und Cobol und der Kommandosprache SDF.

Die Sprachen werden automatisch anhand der Inhalte oder der Dateiendung erkannt. Auch während der Eingabe wird ein Wort sofort farbig angezeigt, sobald es vollständig eingegeben ist.

Bei der Kommandosprache SDF wird ein Kommando bzw. deren Parameter und Parameterwerte auch schon farbig angezeigt, sobald die Abkürzung eindeutig ist.

Das Highlighting kann auch mit dem Kommando SHL (S. 247), dem Menübefehl Ansicht/Syntax-Highlighting (S. 96) oder mit der Schaltfläche der Toolbar (S. 164) aktiviert werden. Die Farben können über den Dialog Extras/Highlighting (S. 149) eingestellt werden.

Januar 2013 — Im Feld "Remote Success/ftp-cmd" des Filetranfer-Profils kann ein oder mehrere FTP-Kommandos angegeben werden. Diese Kommandos werden vor dem Beginn der Übertragung an den FTP-Server gesendet, z.B. "site sbd=(IBM-273,ISO8859-1)". Mehrere Kommando müssen durch die Zeichen ";;" getrennt werden.

Oktober. 2012 — Neue Optionen "Sortieren" und "Auswahl..." im Menü der rechten Maustaste (S. 54) für Profile und Verzeichnisse im Baumfenster.

Juni 2012 — Neue Kommandos @+ und @- (S. 273) sowie Parameter **:***text* zu den Kommandos @*ln* bzw. @set *ln*(*inc*) (S. 301).

Neues Kommando EDIT SEQUENTIAL (S. 190): Modus zum Erhöhen der Zeilennummer ändern.

April 2012 — Bei der Verschlüsselung (S. 357) von EDT-Prozeduren kann auch ein Passwort angegeben werden.

Januar 2012 — Neue Markierung H (S. 175): Schaltet nur für den ausgewählten Satz den Hexadezimalmodus ein und stellt alle zu diesem Satz gehörenden Bildschirmzeilen auf überschreibbar.

Dezember 2011 — Kommando INPUT (S. 206): Zusätzlicher Suchpfad für die Prozedurdatei über die Umgebungsvariable EDTPATHI.

Sept. 2011 — Neues Kommando RUN (S. 238): Benutzerroutine aufrufen.

Änderungsprotokoll

August 2011 Neue Option "Als UTF8-Datei einlesen, falls UTF8-Zeichenfolgen enthalten sind" im Menü `Optionen / Einstellungen / Code` (S. **116**): Dateien ohne Header werden als UTF8-Datei eingelesen, falls in der Datei die in UTF8 codierten Zeichen X'A0' bis X'FF' vorkommen.

März 2011 Die Dialogbox zum Einfügen von Sonderzeichen (S. **91**) wurde um eine Combobox erweitert, in der Gruppen von UNICODE-Zeichen ausgewählt werden können. Nach der Auswahl einer Gruppe wird auf die Zeichen positioniert und die ausgewählten Zeichen werden hervorgehoben dargestellt.

Februar 2011 Im Baumfenster (S. **98**) links neben dem Arbeitsbereich wird die Liste der Laufwerke automatisch beim Laden des EDTW aktualisiert. Bisher wurde die Liste nur bei einem Doppelklick auf den Button "Laufwerke" aktualisiert.

Erweiterung des Menüs der Rechten Maustaste (S. **52**) um die Funktion "Einfügen Sonderzeichen".

Januar 2011 Bei allen Kommandos, bei denen Stringvariablen als Range-Angabe (S. **339**) zulässig sind, z.B. `@PREIFX #S1-#S10w'xx',` kann nun auch die Angabe `#Snn+#Inn` oder `#Snn-#Inn` verwendet werden, z.B.

```
@PREFIX #S1.-#S1+#I1 w'xx'
@UPPER #S10-#I1.-#S20-#I1
```

Der Inhalt der Integervariablen modifiziert dabei die Nummer der Stringvariablen.

Oktober 2010 Neuer Menüpunkt "`Klammersuche`" für die rechte Maustaste (S. **52**): Wenn sich der Cursor vor bzw. auf einer öffnenden oder schließenden Klammer (rund "`()`", eckig "**[]**" oder geschweift "`{}`") befindet, wird die dazugehörende schließende bzw. öffnende Klammer gesucht. Die gesamten Daten von der öffnenden bis zur schließenden Klammer werden markiert. Alle in diesem Bereich enthaltenen Klammern der gleichen Art werden mit einer FIND-Markierung versehen. Diese Funktion war bisher nur über die Tastenkombination `<Strg> "5"` (S. **47**) erreichbar.

Im "Baum-Fenster" für die FT-Profile (S. **100**) werden im Menü der rechten Maustaste zusätzliche Informationen des fernen Systems angezeigt.

März 2010 Beim Kommando `ON` (S. **210**) und `S` (S. **241**) kann als Suchbegriff wie beim CFS auch die Variante L'string' angegeben werden. Unabhängig von der Einstellung der Suchoptionen wird der Suchbegriff nur gefunden, wenn die Klein-/Großschreibung übereinstimmt.

Sept. 2009 Neues EDT-Kommando C*nnn* (S. **178**) zum Positionieren auf eine Spalte. Bisher konnte der Spaltenbereich nur relativ mit dem Kommando ">" bzw. "<" nach links und rechts verschoben werden.

Juni 2009 Neues Kommando `SYMBOLS` (S. **252**) wie im EDT des BS2000 zum Ändern der Jokerzeichen für das Kommando `ON..FIND` (S. **210**)**.**

Da zum Anzeigen der HTMLHELP-Datei EDTW.CHM intern Funktionen des Internet-Explorers benutzt werden, kann es bei Netzlaufwerken zu Problemen kommen, wenn aufgrund von Sicherheitseinstellungen der Zugriff blockiert wird. Die Themen werden dann entweder nicht angezeigt oder es kommt eine Fehlermeldung. Um diese Probleme zu vermeiden, wird die Datei EDTW.CHM, wie die Datei EDTW.INI, in das lokale Verzeichnis %APPDATA% übertragen.

März 2009 Grafische Oberfläche (S. 98) zum Einlesen von Dateien über Filetransfer: Links neben dem Fenster für die Arbeitsbereiche kann wahlweise ein Fenster aktiviert werden (S. 100), in dem entweder eine Liste aller Dateien (wie im Explorer), eine Liste mit den geöffneten Arbeitsbereichen oder eine Liste der Filetransfer-Profile (S. 140) und fernen Dateien in Baumstruktur angezeigt wird. Mit den Schaltflächen "Fenster", "FT-Profile" und "Laufwerke" am untersten Rand des Baumfensters kann zwischen den drei Ansichten umgeschaltet werden.

Dezember 2008 Neue Option `Originaldatei überschreiben ohne Temp-File` im Menü `Optionen / Einstellungen / Sicherung` (S. 108): Diese Option bewirkt, dass beim Überschreiben von bestehenden Dateien, insbesondere bei SAMBA-Laufwerken (Unix-Dateisysteme), die Datei-Attribute nicht verloren gehen.

Oktober 2008 Kommando `READ` (S. 220) und `READ` über Filetransfer (S. 226): XML-Dateien werden anhand des XML-Headers erkannt und ggf. als UNICODE-Datei eingelesen.

Filetransfer mit openFT: Unterstützung openFT für Unix mit dem neuen Profil-Typ (S. 140) openFT-Unix. Die User-ID und der Dateiname werden nicht in Großbuchstaben umgewandelt und der Dateiname wird nicht um die User-ID ergänzt. Die Daten werden immer im Textmodus übertragen, wobei in der Regel keine Code-Umwandlung durchgeführt wird. Die Code-Darstellung für den Arbeitsbereich wird nach dem Lesen einer Datei automatisch auf ISO8859-1 oder auf UNICODE eingestellt. Nicht benötigte Felder des Profils werden schreibgeschützt dargestellt.

Neue Option im Menü `Code / Shell / Dateien ohne Header als UTF-Dateien einlesen` (S. 116): Mit dieser Option kann erreicht werden, dass Dateien, deren Codierung nicht automatisch erkannt werden kann, als UTF-8-Dateien eingelesen werden.

Filetransfer mit Profil FTP_MVS: Beim Übertragen von Elementen eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended) wird im PDS-Diretory das Datum und die Uhrzeit der Änderung versorgt.

Sept. 2008 Filetransfer mit openFT: Berücksichtigung der CCS EDF03IRV und EDF03DRV (S. 139).

Juli 2008 Die HELP-Datei wurde vom Format "WINHELP" (hlp-file) auf "HTMLHELP" (chm-file) umgestellt. Dadurch ergeben sich komfortablere Navigations- und Suchfunktionen. Außerdem wird das WINHELP-Format unter Windows VISTA nur noch optional unterstützt.

Juni 2008 Erweiterung Filetransfer mit openFT:

a) Das Kommando `FSTAT` (S. 200) ist jetzt auch mit openFT-Profilen zulässig. PLAM-Bibliotheken werden als Verzeichnis dargestellt und können mit einem Folge-FSTAT angezeigt werden.
b) Kommando `READ` (S. 226) und `WRITE` (S. 264): BS2000-UNICODE-Dateien (UTF8, UTFE und UTF16) können mit openFT ab Version 10 gelesen und geschrieben werden. Die Kodierung wird automatisch anhand des CCS erkannt.
c) Neuer Parameter CCS zum Kommando `WRITE` (S. 264).
d) Die Angabe der openFT-Version im FT-Profil ist nicht mehr notwendig. Die Version wird automatisch ermittelt.

Januar 2008 Erweiterung der Datei `codepage`.txt (S. 359): Der Code ANSI wurde um die Sonderzeichen auf Position 82-9F erweitert. Der bisherige Code ANSI wurde als ISO8859-1 neu aufgenommen. Bei der Installation mit dem SETUP-Programm ist zu beachten, dass eine bereits bestehende Datei `codepage.txt` nicht überschrieben wird, da die Datei auch vom Benutzer angepaßt werden kann. Soll die aktuelle `codepage.txt` übernommen werden, so muss die Datei `codepage.opg` nach `codepage.txt` kopiert werden.

November 2007 Neues Kommando `PAR TRUE-FALSE=LOCAL|GLOBAL` (S. 219): Modus für die Abfrage mit `IF .TRUE./.FALSE.` (S. 282) einstellen.

Änderungsprotokoll

Oktober 2007 Menü Optionen / Einstellungen / Code / Code-Darstellung / Option Auto (S. 116): Optimierung der Funktion zur automatischen Erkennung der Codierung einer Datei.

Sept. 2007 Kommando LIST (S. 207): Neue Option "C" für die Auswertung von Drucksteuerzeichen in Spalte 1. Die bisherige Option "C" wurde in "S" (Scalezeile) und die Option "G" in "L" (Long) umbenannt.

August 2007 **Version 4.0: Unterstützung von UNICODE-Dateien:**

Die Daten der Arbeitsbereiche werden intern als Zwei-Byte-Zeichen gespeichert. Beim Lesen einer Datei werden die Daten jedoch nicht umcodiert, sondern im Ursprungscode gespeichert und editiert. Daten aus Dateien im Ein-Byte-Code enthalten im ersten Byte demnach immer X'00'.

Daten in Stringvariablen und in der Zwischenablage werden immer in UNICODE codiert und bei Verwendung in Arbeitsbereichen in den Zielcode übersetzt. Beim Kopieren und Verschieben von Daten aus UNICODE-Arbeitsbereichen in andere Arbeitsbereiche und umgekehrt wird ggf. eine Umcodierung vorgenommen.

Neue Optionen zum Kommando READ (S. 220) und WRITE (S. 261):
CHAR: Datei im 1-Byte-Format (Std., falls kein UNICODE-Header)
UTF: Datei im UTF-8-Format
UCB: UTF-16 Unicode Big-Endian (höherwertiges Byte zuerst)
UCL: UTF-16 Unicode Little-Endian (niederwertiges Byte zuerst)
HEA: Beim Schreiben Header voranstellen (Standard)
NOHEA: Datei ohne Header schreiben
Falls einzulesende Dateien einen UNICODE-Header (Byte Order Mark) enthalten, wird automatisch die richtige Codierung verwendet.

Die Bedeutung des Zeichenfolge-Typs U'1234' für Unicode-Strings (S. 341) wurde geändert. Die Zeichenfolge darf nur in einem UNICODE-Arbeitsbereich verwendet werden.

Kommando CODE (S. 181), Menü Funktionen / Code-Umsetzung (S. 83), Menü Ansicht / Anzeige-Code (S. 94) und Menü Optionen / Code-Standard (S. 117): Als Anzeige-Code bzw. Ziel- oder Sendecode kann auch UNICODE angegeben werden.

Das Kommando HEX (S. 206) wurde um die Optionen "2" (2-zeilig) und "4" (4-zeilig) erweitert. Das Kommando HEX 4 ist aber nur in einem UNICODE-Arbeitsbereich zulässig.

Kommando COMPARE (S. 184): Arbeitsbereiche für das Compare-Protokoll und der Protokoll-Arbeitsbereich 32 werden automatisch in Unicode-Codierung angezeigt, falls Daten in Unicode-Codierung enthalten sind.

Kommando VAR LOAD (S. 258) kann alte und neue (UNICODE) Dateien zum Laden von String-Variablen verarbeiten.

Mit dem Menü Funktionen/Einfügen Sonderzeichen... (S. 91) können alle zulässigen Zeichen eines Zeichensatzes in das Datenfenster oder in die Kommandozeile eingefügt werden.

August 2007 Kommando WINGEN ATTR (S. 319): Die Schriftfarbe und Hintergrundfarbe kann nun auch für Buttons geändert werden.

Kommando WINGEN LISTBOX (S. 328) und WINGEN TABLE (S. 333): Es kann nun auch eine Aktion für einen Doppelklick definiert werden.

Das Kommando STRIP (S. 251) ist nun auch für Stringvariable anwendbar.

Das Kommando HALT (S. 205) wurde um die Optionen ISAVE und INOSAVE ergänzt. Damit kann, insbesondere in Prozeduren, bestimmt werden, ob evtl. geänderte Einstellungen in der INI-Datei gespeichert werden sollen.

Juli 2007

Mit dem Menüpunkt Funktionen / Einfügen Sonderzeichen... (S. 91) kann nun auch in die Kommandozeile oder in das Datenfenster ein Sonderzeichen eingefügt werden.

Die Routine zum Vergleichen von zwei Arbeitsbereichen mit dem Kommando COMPARE (S. 184) wurde optimiert. Bisher konnte es vorkommen, dass in bestimmten Konstellationen der Vergleich abgebrochen wurde. In diesem Fall wurde die folgende Meldung ausgegeben:

"Vergleich ist unvollständig. Zu viele Unterschiede vorhanden."

Juni 2007

Erweiterung des Filetransfer mit openFT: Im FT-Profil kann nun auch ein CCS (S. 144) definiert werden. Das CCS wird im BS2000 als Dateiattribut übernommen. Außerdem wird es für die Code-Konvertierung ausgewertet.

Beim FT mit Profil-Typ FTP_MVS können beim Kommando WRITE, Parameter *mvs_fileattr* (S. 264) die Dateiattribute LRECL, RECFM und BLKSIZE angegeben werden.

April 2007

Dateien, die über die Funktion EDTW des URLServers oder über das BS2000-Programm FILESEND zu einem PC gesendet und dann in den EDTW eingelesen werden, können nun auch über einen Proxy-Server empfangen und/oder zurückgesandt werden. Dadurch ist es möglich, Dateien zu einem PC zu senden, der im BS2000 nicht bekannt ist. Ebenfalls ist es möglich, zum Empfangen im BS2000 statt eines dynamischen Ports nur einen zentralen Port zu verwenden.

Die Proxy-Server für diesen Service werden im Rahmen des Programmpakets OPGCOM zur Verfügung gestellt.

Januar 2007

Erweiterung des SET-Kommandos zum Rechnen mit Datum und Uhrzeit. Dazu werden Datums- und Zeitangaben in Sekunden seit 1.1.1970 umgewandelt. Danach kann es verändert werden und wieder in eine Zeichenfolge umgewandelt werden.

a) SET *intvar* = TIME [*string*] (S. 291)
b) SET *intvar* = DAY [*intvar*] (S. 292)
c) SET *strvar*|*linevar* = DATE|TIME [*intvar*] (S. 299)

Dezember 2006

Neue Option OPENMULTI zum Kommandos DIALOGBOX (S. 275): Damit können auch mehrere Dateien ausgewählt werden. Die ausgewählten Dateinamen werden wahlweise in eine Stringvariable oder in eine Datei geschrieben.

Das Kommando ERASE bzw. UNSAVE (S. 257) funktioniert jetzt auch beim Löschen von entfernten Dateien, soweit der Filetransfer mit FTP durchgeführt wird. Das Löschen kann auch mit der Markierung E (S. 204) in einer FSTAT-Liste (S. 200) erfolgen.

Oktober 2006

Beim Kommando FSTAT (S. 200) für entfernte MVS-Dateien werden auch über HSM migrierte Dateien mit dem Attribut "Migrated" angezeigt.

Sept. 2006

Neuer Parameter "P" zum Kommando WINDEF (S. 301) für die verborgene Eingabe eines Passwortes.

Neue Parameter zu den Kommandos FILE (S. 192), FSTAT (S. 200), READ (S. 226) und WRITE (S. 264) für ferne Dateien zur Eingabe der Zugangsdaten.

Juni 2006

Neue Option UTF zum Kommando REFORMAT (S. 234) und UNFORMAT (S. 255): Mit der Option UTF können in einem Arbeitsbereich Daten vom Format UTF-8 in ISO 8859 und umgekehrt umgewandelt werden.

Änderungsprotokoll

März 2006 Neues Kommando DIALOGBOX (S. 275): Mit dem Kommando kann zur Auswahl eines Dateinamens die Dialogbox OPEN und SAVE AS erzeugt werden. Der ausgewählte Dateiname wird in eine Stringvariable geschrieben.

Februar 2006 Kommando WINGEN FONT (S. 315): Es können zusätzliche Attribute wie fett, kursiv, unterstrichen, blinkend usw. angegeben werden.

In einigen Listboxen des EDTW und in den durch WINGEN (S. 328) erzeugten Listboxen wird ein horizontaler Scrollbar ausgegeben, wenn die enthaltenen Daten breiter sind als die Listbox.

Januar 2006 Die Filetransfer-Profile (S. 136) werden in der Listbox Extras / Filetransfer nach Namen sortiert.

Die Liste der letzten Kommandos (siehe Button 

) kann mit dem Button 

nach dem Kommandonamen sortiert werden.

Die Liste der zuletzt verwendeten Dateien (Menü Datei / weitere Dateien) kann mit dem Button 

nach Namen sortiert werden.

Neue Funktionen für WINGEN:

Neues Kommando WINGEN PROGRESS (S. 330) zur Definition eines PROGRESS-BAR (Anzeige des Programmfortschritts) und WINGEN ATTRP (S. 319) zur Änderung der aktuellen Werte des PROGRESS-BAR's.

Neue Option ASYNC und CLOSE zum Kommando WINGEN OUT (S. 320) für die Ausgabe und zum Schließen von asynchronen Fenstern.

Neue Option WAIT zum Kommando WINGEN OUT (S. 320) für die Ausgabe eines synchronen Fensters, das nach Ablauf der angegebenen Sekunden automatisch geschlossen wird.

Neue Option PASS zum Kommando WINGEN EDIT (S. 326) für die verborgene Eingabe eines Passwortes.

Neue Systemschaltflächen (S. 317) in der Titelzeile.

Neue Anweisung IF WINGEN .... EXISTING (S. 280) zur Abfrage, ob ein asynchrones Fenster noch existiert.

Dezember 2005 Neue EDT-System-Variablen (S. 344) !RLIB und !RELEM (Name Bibliothek und Element von PLAM-Bibliothek bzw. MVS-PDS-Archiv)

November 2005 Beim Kommando READ (S. 226) für ferne Dateien kann auch ein Zeilen- und Spaltenbereich angegeben werden.

Aus einer FSTAT-Liste (S. 204) kann eine Datei in einen anderen Arbeitsbereich mit dem Kommando READ *zlnr(arb)* eingelesen werden.

Oktober 2005 Dialogbox Kommandogedächtnis (S. 135): Mit dem Button Kopieren können Zeilen aus der Kommandoliste in die Zwischenablage kopiert werden.

Kommando LIST (S. 207): Berücksichtigung von Seitenvorschub

Sept. 2005 Für den Filetransfer (S. 226) mit FTP zu bzw. von einem MVS-Host kann ein vollqualifizierter Dataset Name mit dem Prefix "mvs:" oder "$" angegeben werden, so dass die Verdoppelung der Hochkommas entfallen kann.

Nach einem entfernten FSTAT (S. 200) mit MVS-Dateien kann in der Markierungsspalte bei PO-Dateien durch Angabe eines Arbeitsbereichs ein Folge-FSTAT erzeugt werden.

Nach einem lokalen FSTAT (S. 197) kann in der Markierungsspalte bei ZIP-Dateien durch Angabe eines Arbeitsbereichs ein Folge-FSTAT erzeugt werden. Das Gleiche gilt, wenn im Tree-Fenster ein Doppelklick auf ein ZIP-Archiv gemacht wird.

August 2005

Kommando ON...SEARCH (S. 212) und S (S. 241): Als Suchbegriff kann auch eine Stringvariable und als Spaltenangabe kann eine Integervariable angegeben werden.

Juli 2005

Schreiben von BS2000-Dateien über das BS2000-Programm FILESEND und URLServer: Falls Daten von den BS2000-Programmen FILESEND oder URLServer von EDTW gelesen werden, können diese Daten mit dem Kommando WRITE wieder in die BS2000-Datei zurück geschrieben werden. Auf diesem Wege können auch ISAM-Dateien, PAM-Dateien und delta-gespeicherte LMS-Elemente editiert werden. Weitere Informationen zu diesem Thema siehe Kommando WRITE ferne Dateien (S. 270) und Manual OPGCOM.PDF http://www.opg.de/produkte/frames/opgcom.pdf.

Neues Kommando STATUS FSEND (S. 250) zum Anzeigen der Header-Informationen von FILESEND-Dateien.

Januar 2005

Lesen und Schreiben von ZIP-Elementen mit den Kommandos READ (S. 220) und WRITE (S. 261). Bei beiden Kommandos können alle Optionen angegeben werden, wie z.B. OPEN, BINARY, KEY, *rngcol* zum Kommando READ und OVERWRITE, UPDATE, KEY, X, *rngcol* zum Kommando WRITE.

Sept. 2004

Kommando FSTAT (S. 197): Mittels der Option A (Archive) können Elemente eines ZIP-Archivs angezeigt und mit Kommandos in der Markierungsspalte weiterverarbeitet werden.

Kommando FSTAT (S. 197) und COMP (S. 184): Durch Angabe der Option NF (no focus) wird der Ergebnis-Arbeitsbereich in Prozeduren nicht angezeigt.

Kommando PROC (S. 286) Variante FREE | USED | STACK: Neue Optionen NA (Not used Areas) für die Ermittlung der nicht benutzten Arbeitsbereiche. Alle Informationen können auch in String- oder Integer-Variablen übertragen werden.

Neues Kommando SETF (S. 246): Mit diesem Kommando kann die Anzeige auf einen anderen Arbeitsbereich umgeschaltet werden. Zusätzlich ist die Positionierung auf eine Zeile und Spalte möglich. Das Kommando kann im Gegensatz zum BS2000-EDT auch in einer Prozedur angegeben werden. Damit kann bestimmt werden, was während des Ablaufs einer Prozedur und am Ende der Prozedur angezeigt wird.

Kommando WINGEN (S. 307): Bei allen Elementen für die Dialogbox kann die Schriftart, die Schriftgröße und die Farbe angegeben werden. Dazu wurden die neuen Varianten WINGEN FONT, WINGEN COLOR und WINGEN STD eingeführt. Die Attribut-Angaben wurden um die Angaben, Schriftart, Schriftgröße und Farbe ergänzt.

Kommando PAR FPOS (S. 217): Mit den Werten Y und N kann die Positionierung nach dem ON&FIND-Kommando ein- und ausgeschaltet werden.

Kommandos PROC (S. 286) und DO (S. 277): Die Nummer des Arbeitsbereichs kann auch in einer Integer-Variablen angegeben werden.

Kommando GOTO (S. 279) und IF ... GOTO: Das Sprungziel kann auch in einer Stringvariablen, die den Namen des Labels enthält, angegeben werden.

Kommando PAR: Neue Varianten IGNORE-LINE-ERR=Y|N (S. 217) und MULTIREAD=#Sn,#In,#In (S. 218).

Kommando MSGBOX (S. 283): Es kann zusätzlich eine Ergebnis-Variable angegeben werden (bisher immer #S0).

Kommando CAT (S. 178) und CUT (S. 186): Es können zusätzlich Ergebnis-Variable angegeben werden (bisher immer #I0 und #I1).

Kommando ON#Sn-#SnFIND (S. 210) Die Nummer der Stringvariable mit dem ersten gefundenen Suchbegriff wird in die Variable #I99 übertragen. Bisher wurde in diesen Fällen die Nummer der String-Variablen in die Line-Variable #L0 übertragen.

Mai 2004 Erweiterung der Kommandos REFORMAT (S. 234) und UNFORMAT (S. 255) um die Varianten LF4 (4 Byte langes Satzlängenfeld) und LF2+ (2 Byte langes Satzlängenfeld + X'4040').

März 2004 Erweiterung der Option OPEN zum Kommando READ (S. 220):

a) Es können nun Dateien bis zu einer Größe von 1.024 GB (1 Terrabyte) eingelesen werden (bisher 2 GB).

b) Auch im OPEN-Modus ist beim Einlesen die Angabe von Zeilen- und Spaltenbereichen zulässig.

c) Nach dem Einlesen einer Datei im OPEN-Modus können noch weitere Dateien in den gleichen Arbeitsbereich eingelesen werden. Die Daten werden dabei in die temporäre Plattendatei geschrieben.

Die Verwaltungsbereiche für die Daten wurden um über 40 % gekürzt, so dass größere Datenmengen eingelesen werden können. Pro Datensatz werden nur noch 42 statt bisher 76 Bytes benötigt.

Dezember 2003 Neue Option OPEN zum Kommando READ (S. 220), READ mehrfach (S. 224) und READ remote (S. 226) für große Dateien: Die Daten werden nicht in den Arbeitsspeicher eingelesen. Im Speicher wird nur ein Verzeichnis der Datensätze mit den Plattenadressen angelegt. Benötigte Daten der Eingabedatei werden bei Bedarf direkt von der Platte gelesen. Neue bzw. geänderte Datensätze werde in eine temporäre Datei geschrieben. Beim Zurückschreiben der geänderten Datei mit dem Kommando WRITE werden die Daten aus der Eingabedatei und der temporären Datei zusammengefügt und in die Ausgabedatei geschrieben.

Zum Kommando DIALOG (S. 275) kann wahlweise ein Hinweistext angegeben werden, der zum Beginn des Dialog-Modus in einer Messagebox ausgegeben wird. Außerdem kehrt das Kommando RETURN immer in den Arbeitsbereich zurück, der beim Kommando DIALOG aktiv war.

Mit dem Kommando PROC (S. 286) kann auch in den Arbeitsbereich 0 gewechselt werden.

August 2003 Bei den Kommandos IF .TRUE./.FALSE./.EMPTY. (S. 282) kann zusätzlich ein Arbeitsbereich angegeben werden. Die True- und False- Bedingung bezieht sich dann auf das letzte ON-Kommando für den entsprechenden Arbeitsbereich.

Die Abfrage mit IF .TRUE. und IF .FALSE. bezog sich bisher auf das letzte ON-Kommando im aktuellen Arbeitsbereich. Im BS2000-EDT bezieht sich die TRUE- bzw. FALSE-Abfrage global auf das letzte ON-Kommando. Im Menü Optionen / Einstellungen / Verschiedenes (S. 127) wurde eine neue Option eingeführt, mit der das Verhalten des BS2000-EDT eingestellt werden kann.

April 2003 Beim Einlesen von mehreren Dateien in einen Arbeitsbereich (Menü Datei / Öffnen mehrfach oder Kommando READ (S. 224) mit Jokerzeichen "*" oder "?") werden die Dateien alphabetisch nach Namen sortiert.

Februar 2003 Bereichsangaben beim Lesen im Binärmodus:

Beim Lesen (S. 220) von Dateien im Binärmodus (read'*file*'b) kann nun auch ein Bereich angegeben werden, um einen Teil einer großen Datei einzulesen.

Dateien, die mit dem Kommando UNFORMAT (S. 255) in Sätze mit Satzlängenfeld geändert werden, erhalten einen Header-Satz. Werden solche Dateien mit dem EDT eingelesen, wird automatisch das Kommando REFORMAT LF und ggf. das Kommando CODE EBC durchgeführt. Das gleiche gilt für Dateien, die im Rahmen des binären FT und vom URLServer erzeugt werden.

Alle Start-Parameter (S. 35), die zusätzliche Angaben erfordern, wie z.B. -c *file*, -i*inputfile* oder -p*inifile*, können mit oder ohne Blank nach dem Schalter angegeben werden.

Die INI-Datei wurde bisher immer in das Systemverzeichnis WINDOWS oder WINNT geschrieben. Falls für das Systemverzeichnis nur der Administrator Schreibrechte besitzt oder mehrere Benutzer unterschiedliche INI-Dateien benutzen wollen, führt dies zu Problemen. Deshalb wird jetzt das Verzeichnis %APPDATA%\OPG benutzt. Die Variable APPDATA wird in der Regel vom System benutzerspezifisch versorgt. Ist die Variable nicht vorhanden, wird wie bisher das Systemverzeichnis benutzt. Bei der ersten Benutzung des neuen Pfades wird die alte INI-Datei kopiert.

Januar 2003 — Erweiterung des binären Filetransfers:

Beim fernen READ (S. 226) und WRITE (S. 264) im Binärmodus (Parameter M=B) können jetzt auch Dateien mit einer Satzlänge > 2048 verarbeitet werden.

Sept. 2002 — Neue Tastenkombination <Strg> "5" (S. 47): Wenn sich der Cursor vor bzw. auf einer öffnenden oder schließenden Klammer (rund "()", eckig "[]" oder geschweift "{}") befindet, wird die dazugehörende schließende bzw. öffnende Klammer gesucht. Die gesamten Daten von der öffnenden bis zur schließenden Klammer werden markiert. Alle in diesem Bereich enthaltenen Klammern der gleichen Art werden mit einer FIND-Markierung versehen.

August 2002 — Neue Variante des Kommandos COLUMN (S. 183): Mit der Option REPLACE kann ein Spaltenbereich mit einem String ersetzt werden, der auch kürzer oder länger sein kann. Zusätzlich ist bei allen Varianten die neue Option NOSTRIP hinzugekommen, mit der das Löschen der Leerzeichen am Ende der Zeile verhindert werden kann.

Bei der Option Optionen / Einstellungen / Suchen (S. 110) "Frühere Treffer zurücksetzen" und Kommando DMA ON (S. 188) werden alle Treffermarkierungen im gesamten Arbeitsbereich zurückgesetzt. Bisher wurden nur die Treffermarkierungen im Suchbereich gelöscht.

Maus-Markierungen in Views (S. 260) werden für jede View getrennt ausgewertet. Bisher war für alle Views eines Arbeitsbereichs nur eine Markierung möglich.

Juli 2002 — Die mit openFT zum BS2000 übertragen Dateien werden nun auch bei BLK-CONTRL=DATA mit der richtigen BUF-LEN angelegt. Bisher wurde in diesen Fällen immer BUF-LEN=(STD,2) erzeugt.

Im Dialog Datei/Öffnen (S. 57) und Datei/Anhängen (S. 59) können auch mehrere Dateien durch Benutzung der Taste Strg bzw. Shift markiert werden. Alle markierten Dateien werden bei Öffnen in verschiedenen Arbeitsbereichen, bei Anhängen in den gleichen Arbeitsbereich eingelesen.

Juni 2002 — Das Lesen und Schreiben von großen Dateien wurde wesentlich beschleunigt.

Mai 2002 — Neue Option "I" (insensitively) zum Kommando SORT (S. 248). Die Sortierung erfolgt damit unabhängig von der Klein- / Großschreibung. Die Option kann für jeden Spaltenbereich gesondert angegeben werden.

Änderungsprotokoll

März 2002 — Die Cursorform (S. **96**) kann nun für den Einfüge- und Überschreibe-Modus getrennt eingestellt werden. Bisher war die Cursorform im Überschreibemodus immer im Format "Voll".

Januar 2002 — Neue Varianten des Kommandos `ON&FIND` (S. **213**) zum Suchen von Sätzen mit bestimmten Satzende-Kennzeichen:
`on&FIND *DOS / *UNIX / *NO.`

Im Protokoll zum FTP (S. **136**) werden Passwörter verschlüsselt (Zeichen '*') dargestellt.

Im Überschreibe-Modus wechselt die Form des Cursors.

Neues Kommando `SPLIT` (S. **249**)**:** Es können 2 oder 3 Arbeitsbereiche nebeneinander oder untereinander positioniert werden.

Erweiterung Kommando `DO` (S. **277**)**:** Als Erhöhungswert für das Schleifensymbol kann auch *n*L (um *n* Zeilen erhöhen) angegeben werden.

November 2001 — Neue Option A zum Kommando `STT`**:** Alle aufeinander folgende Leerzeichen werden unabhängig von der Tabulatorschrittweite in ein Tabulatorzeichen umgewandelt.

Oktober 2001 — Bisher wurden Dateien, die mit openFT zum BS2000 übertragen wurden, immer mit `BUF-LEN=STD(2)` bzw. bei `BLK-CONTR=DATA` mit `STD(3)` angelegt. Die Blocklänge wird nun in Abhängigkeit des längsten Satzes ermittelt. Dateien, die nur Sätze bis zu einer Länge 2024 enthalten, werden demnach mit `BUF-LEN=STD(1)` angelegt und erhalten somit das Standardformat.

Juli 2001 — Nach einem einstellbaren Zeitintervall und bei einem Wechsel des Arbeitsbereichs wird geprüft, ob die eingelesenen Dateien inzwischen außerhalb des Arbeitsbereichs geändert worden sind. Auf Anforderung werden die Daten neu eingelesen. Das Zeitintervall kann unter `Optionen / Einstellungen / Sicherung` (S. **106**) eingestellt werden.

Juni 2001 — Mit dem Kommando `READ` (S. **220**) `und WRITE` (S. **261**) können auch Daten von STDIN gelesen und nach STDOUT oder STDERR geschrieben werden. Beim Starten von EDTW kann durch den Schalter `-stdin` (S. **35**) das Einlesen von STDIN erreicht werden.

Mai 2001 — Mit dem Button  bzw. der neuen Option "ALL" des Kommandos `WRITE` (S. **261**) können alle Arbeitsbereiche gespeichert werden.

Neues Kommando `SHOWNIL` (S. **247**) zur übersichtlicheren Darstellung von NIL-Zeichen, z.B. bei Daten im UNICODE-Format.

Neuer String-Typ U (Unicode): Jedes Zeichen wird in 2 Byte lange Unicode-Zeichen umgewandelt.

Die Aktionen der Menüzeile, die einem EDT-Kommando entsprechen (z.B. Datei/öffnen, Funktionen/Sortieren, usw.), werden in das Kommandogedächtnis aufgenommen.

April 2001 — Mit openFT-BS2 kann auch eine Generation aus einer Datei-Generationsgruppen übertragen werden. Die Generation wird wie im BS2000 beim entfernten Namen in Klammern angegeben, z.B. `fgg(*0003)`.

März 2001 — Mit den neuen Kommandos `VAR` (S. **258**) können alle Variablen in einen Speicherbereich oder in eine Datei gesichert und wieder geladen werden.

Die Einstellungen zum Menü `Ansicht / Statuszeile` und Toolbars werden in der INI-Datei gespeichert und werden somit auch für den nächsten Programmstart berücksichtigt.

Februar 2001 — Alle Kommandos (S. **177**) und Parameter können bis zur Eindeutigkeit wie im BS2000 beliebig abgekürzt werden (bisher war nur die kürzeste und längste Form zulässig).

Die Verwendung von Strings wurde global erweitert. In allen Kommandos, in denen eine Zeichenfolge (S. 341) verwendet wird, können mehrere Zeichenfolgen miteinander verknüpft werden. Als Verknüpfungszeichen ist das Zeichen "+" vorgesehen, z.B. `'xxx'+#s1+#s2+#l1:50-60:`

Neues Kommando SETENV (S. 245): Mit diesem Kommando können Umgebungsvariable gesetzt werden. Diese Umgebungsvariablen können allerdings nur in vom EDT erzeugten Prozessen (z.B. durch Kommando SYS) und in EDT-Prozeduren ausgewertet werden.

Der Inhalt von Umgebungsvariablen kann in jedem String verwendet werden. Dazu wurden die EDT-System-Variablen (S. 344) um die Variante `!%`*env*`%` erweitert, z.B. `@crea1:'!%TMP%'` oder `@set#s1='!%TMP%'`.

Neue Kommandos COPYFILE (S. 186) und MOVEFILE (S. 210) zum Kopieren bzw. Verschieben von Dateien sowie MKDIR (S. 209) und RMDIR (S. 238) zum Anlegen und Löschen von Verzeichnissen.

Januar 2001

In einem Arbeitsbereich, in dem Dateinamen nach dem Kommando FSTAT (S. 197) stehen, können folgende neue Aktionen in der Markierungsspalte angegeben werden:

a) Die Markierung F (freier Arbeitsbereich) bewirkt das Einlesen der Datei bzw. bei Verzeichnissen einen neuen FSTAT mit dem markierten Unterverzeichnis. Die Daten werden in den nächsten freien Arbeitsbereich eingelesen.

b) Mit der Markierung E (Erase) wird sowohl die Zeile aus der Dateienliste als auch die Datei auf dem Datenträger gelöscht (wie EDT-Kommando UNSAVE (S. 257)).

Das Kommando FSTAT (S. 197) mit einem FTP-Profil (Profil-Typ FTP-BS2000 oder FTP-POSIX) ist jetzt auch für BS2000- und POSIX-Dateien möglich. Der Zugriff über FTP wurde für die verschiedenen Plattformen optimiert. Deshalb muss im Profil genau der Typ der FTP-Verbindung angegeben werden (FTP-UNIX, FTP-BS2000 oder FTP-POSIX). Die Weiterverarbeitung von Einträgen in einem FSTAT-Arbeitsbereich über die Markierungsspalte ist auch bei entfernten Dateien möglich.

Wird der Name von EDT-Prozedurdateien beim Kommando INPUT (S. 206) bzw. beim Laden des EDTW mit dem Schalter `-i` ohne Verzeichnis angegeben, so wird die Datei neben dem aktuellen Verzeichnis zusätzlich im Ladeverzeichnis und in den Verzeichnissen der Variablen PATH gesucht.

Mit dem Kommando UNSAVE (S. 257) können auch Dateien auf fremden Rechnern gelöscht werden.

Dez. 2000

Unterstützung von "Alternate Data Streams" (ADS) auf NTFS-Dateisystemen: Die ADS werden bei den Kommandos READ (S. 220) und WRITE (S. 261) nach dem Dateinamen durch Doppelpunkt getrennt, angegeben, z.B. `datei:stream1`. Die Option ADS zum Kommando FSTAT bewirkt, dass die Alternate Data Streams zusätzlich in der FSTAT-Liste angezeigt werden. Mit der Option ADSO werden nur die Dateien mit Alternate Data Streams angezeigt.

Die DDE-Verknüpfung (S. 35) mit dem Windows-Explorer (Doppelklick auf Datei) wurde überarbeitet (Probleme Windows 2000). Zusätzlich wurde die Verknüpfungsart PRINT (im Explorer rechte Maus-Taste/Drucken) und PRINTTO (Datei mit der Maus auf ein Druckersymbol ziehen) eingeführt.

Oktober 2000

Neue Varianten des Kommandos SET (S. 297) zum Konvertieren von Character-Strings in Hexa-Strings und umgekehrt: `SET #S`*n* `CONVX/CONVC` *string*.

Sept. 2000

In einem Arbeitsbereich, in dem Dateinamen nach dem Kommando FSTAT (S. 197) stehen, kann auch bei einem Verzeichnis die Markierung 0-9 (FSTAT für Unterverzeichnis in neuen Arbeitsbereich) eingegeben werden.

Neue Option SIZE zum Kommando LIMITS (S. 207) zur Ermittlung der Anzahl der Bytes eines Arbeitsbereichs.

Neue Option K (Kilo Separator) zu den Kommandos SET #S*n*=C#I*n* (S. 293) und SET #L*n*=C#I*n* (S. 298) für die Umwandlung von Zahlen in Strings mit Tausender-Trennzeichen.

Neues Kommandos SET #S*n* VARSUBST #S*n* (S. 297) zum Ersetzen von EDT-System-Variablen (!file usw.) in Stringvariablen.

August 2000 Erweiterung der FT-Profile (S. 140):

a) Die Möglichkeiten für die Bearbeitung von Dateien auf anderen Rechnern wurden um die Variante des Filetransfers mit FTP erweitert. Das Kommando FSTAT (S. 197) mit FTP für ferne Dateien funktioniert vorerst nicht für BS2000-Dateien.

b) Im FT-Profil kann nun auch die Codepage für die Anzeige der entfernten Dateien angegeben werden.

c) Für die Behandlung von Leersätzen wurde eine neue Option aufgenommen.

d) In den Feldern Host, User-ID und Abrechnungsnummer kann ein Fragezeichen eingetragen werden. Dies bewirkt, dass bei der Ausführung des Kommandos READ oder WRITE die entsprechenden Daten abgefragt werden (bisher nur für das Logon-Passwort und dem Datei-Passwort möglich).

e) Bei Verwendung des 8-Bit Zeichencodes EDF041 im BS2000 kann die ausgelieferte Datei EDF041.TrT mit der Translate-Tabelle für die Konvertierung EDF041 < -- > ISO8859 im Filetransfer-Profil eingetragen werden.

Der Wertebereich der Integer-Variablen wurde vergrößert. Statt des bisherigen Maximalwertes von 2.147.483.647 (2 GB -1) können nun Zahlen bis 9.223.372.036.854.775.807 (8.388.607 Terrabyte) verarbeitet werden. Die SET-Kommandos SET #S*n*=C#I*n* (S. 293) und SET #L*n*=C#I*n* (S. 298) wurden um die Variante "CL" (Char Long) für die Aufbereitung von langen Zahlen erweitert.

Juli 2000 Bei den Einstellungen der Schriftart zum Drucken und der Schriftart für die Fenster der Arbeitsbereiche können nun auch die Optionen "Fett" und "Kursiv" eingestellt werden.

Die Option KEY für die Kommandos READ (S. 220) und WRITE (S. 261) wurde wie im BS2000-EDT implementiert.

Juni 2000 Die Funktionen für die Code-Umwandlung wurden um neue Code-Varianten erweitert. Außerdem wurden die Codetabellen in die Datei codepage.txt ausgelagert. Dadurch ist es möglich, eigene Code-Varianten zu definieren. Unicode-Zeichen > 256, die über den normalen Zeichenumfang hinausgehen, z.B. Umrahmungszeichen, Sonderzeichen usw. wie sie im Code ASCIIDOS vorkommen, können dargestellt werden, soweit sie im Windows-Zeichensatz enthalten sind. Folgende Kommandos und Menübefehle sind betroffen:

- Kommando CODE (S. 181)
- Menü Ansicht/Code (S. 94)
- Menü Funktionen/Code (S. 83)
- Menü Optionen/Einstellungen/Code (S. 116)
- Kapitel 17 Code-Tabellen (S. 359)

Die Informationen zur Umwandlung von Klein- in Großbuchstaben und umgekehrt (Kommando UPPER (S. 258) und LOWER (S. wurden ebenfalls in die Codetabellen (Abschnitt LOWCAPTAB) aufgenommen. Dadurch werden alle Buchstaben (auch Umlaute und Sonderzeichen, wie á, â usw.) in allen Code-Varianten in die Konvertierung einbezogen.

Bisher wurde ein fehlerhaftes Kommando in der Kommandozeile nach der Fehlermeldung gelöscht. Nun bleibt das Kommando erhalten.

Mai 2000

Die Behandlung der Satz-Trennzeichen (X'0D0A' bzw. X'0A') wurde optimiert. Es können nun auch Dateien bearbeitet werden, in denen beide Varianten vorkommen. Bei gemischten Dateien wird für jeden Satz in der Satznummer das Satzende-Kennzeichen angezeigt. Weitere Informationen siehe Abschnitt Satzstruktur (S. **40**).

April 2000

Im Menü `Extras/Filetransfer` (S. **136**) kann mit der Option International/Deutsch gewählt werden, ob bei Benutzung der Standard-Translate-Tabelle die Umlaute berücksichtigen werden sollen.

März 2000

Neues Kommando `WINGEN` (S. **307**): Mit diesem Kommando können Sie umfangreiche Dialogboxen mit allen üblichen grafischen Elementen, wie z.B. Buttons, Listbox, Combobox, Radio-Button usw., erstellen. Dieses Kommando funktioniert nur ab Version WINDOWS-NT 4.0 bzw. Windows Me. Es können beliebig viele Grafikelemente enthalten sein. Die Buttons können mit EDT-Prozeduren verbunden werden, die beim Anklicken mit der Maus ausgeführt werden. Damit können Sie Ihre EDT-Prozeduren mit einem professionellen User-Interface aufwerten. Das Kommando wird wegen des Umfangs in einem eigenen Kapitel 11 beschrieben.

Beim Kommando `SORT` (S. **248**) können nun auch mehrere Spaltenbereiche angegeben werden.

Februar 2000

Neue Varianten des Kommandos `IF` (S. **280**):

Prüfen, ob eine Zeile markiert ist: `IF` *line-var* `= SELECTED ...`
Prüfen, ob eine Zeile existiert: `IF` *line-var* `= EXIST ....`

Januar 2000

In einem Multi-Arbeitsbereich (mehrere Dateien in einem Arbeitsbereich) werden per rechtem Mausclick (S. **53**) zusätzliche Funktionen für die Bearbeitung der aktuellen Datei zur Verfügung gestellt.

Neben den bisherigen EDT-Variablen können nun zusätzlich 100 Gleitpunkt-Variable (`#F00 - #F99`) verwendet werden. Für die Gleitpunkt-Variablen (S. **337**) wurden einige SET-Kommandos (S. **292**) angepaßt bzw. neue Varianten eingeführt.

Neue Variante des Kommandos `SET` (S. **292**): Mit `SET #I`*n*`=RECORDS` wird die Anzahl der Sätze des Arbeitsbereichs ermittelt.

Inhaltsverzeichnis

1 Einleitung

Zielsetzung, Zielgruppen und Konzept des Handbuchs

Das vorliegende Handbuch beschreibt den Aufruf, die Bedienung sowie die Dialog- und Prozedur-Kommandos des Programms EDT für WINDOWS.

Sie kennen das Programm EDT aus der Mainframe-Welt des BS2000 und wollen unter WINDOWS die gleiche Oberfläche und den gleichen Funktionsumfang nutzen. Sie wollen insbesondere auch EDT-Prozeduren aus dem BS2000 unter WINDOWS verwenden.

Erwähnt sei an dieser Stelle, dass das Programm EDT und die vom BS2000 gewohnte Dateibearbeitungsoberfläche CFS auch unter dem Betriebssystem SINIX und vielen anderen UNIX-Plattformen zur Verfügung stehen.

Welche Vorkenntnisse sind nötig?

Für die Arbeit mit dem Programm EDT benötigen Sie Grundkenntnisse des MS-WINDOWS. Kenntnisse des EDT aus dem BS2000 sind wünschenswert, aber nicht Voraussetzung. Zur Anwendung der Prozedursprache des EDT sollten Ihnen die Prozeduranweisungen des EDT im BS2000 bekannt sein. Für eine weitergehende Beschreibung der in diesem Handbuch dargestellten Kommandos verweisen wir auf das EDT-Benutzerhandbuch der Fujitsu Technology Solutions GmbH.

Allgemeine Vereinbarungen

Im vorliegenden Buch werden bestimmte Zeichen (sogenannte Metazeichen) zur Darstellung der verschiedenen Eingabemöglichkeiten verwendet.

Formale Darstellung	Erläuterung	Beispiel

XXXXXX YYYYYY	Worte in Großbuchstaben bezeichnen Kommandos, die in dieser Form eingegeben werden müssen. Die Eingabe kann aber in Kleinbuchstaben erfolgen.
	
	In großen Großbuchstaben wird die minimale Abkürzungsmöglichkeit angegeben.
		EDIT LONG
		
		Einzugeben ist:
		`edit long`
		
		`e l`

kursiv	Kursive Kleinbuchstaben bezeichnen Variablen, die bei der Eingabe durch aktuelle Werte ersetzt werden müssen, d.h. ihr Inhalt kann von Fall zu Fall verschieden sein. Die Eingabe erfolgt in der Regel in Kleinbuchstaben, bei Bedarf, z.B. bei Dateinamen, Suchbegriffen in Groß- und Kleinbuchstaben.	READ '*file*' Einzugeben ist: `read 'datei'` usw.
[.....]	Eckige Klammern schließen Wahleingaben ein. Diese können angegeben oder weggelassen werden. Steht bei Wahleingaben ein Komma innerhalb der Klammer, so wird es nur bei Verwendung dieser Eingabe verlangt. Runde Klammern (..) müssen stets so eingegeben werden wie beschrieben.	HALT [N] Einzugeben ist: `halt n` oder `halt`

Formale Darstellung	Erläuterung	Beispiel
AAAAAA	Die Unterstreichung hebt den Standardwert (Voreinstellung) hervor. Dies ist der Wert, der eingesetzt wird, wenn Sie keine Angabe gemacht haben.	LOWER ON|OFF Einzugeben ist: `low on` oder gleichwertig `low` `low off` kann zu `low o` abgekürzt werden
|	Der senkrechte Strich trennt alternativ zu verwendende Eingaben.	PAR FPOS=[-|+] *n* Einzugeben ist z.B.: `par fpos=-4` oder `par fpos=+4`
{ ..... }	Geschweifte Klammern schließen Alternativen ein, d.h. aus den eingeschlossenen Angaben muss eine ausgewählt werden.	COPY *rng* { A | B } *ln* Einzugeben ist: `copy 1-2 a 20` oder `copy 1-2 b 20`
_	Dieses Symbol kennzeichnet ein Leerzeichen (x'20').	
`F1`, `F2`, `F3` usw.	Diese Symbole kennzeichnen die entsprechenden Tasten F1 bis F9.	
`<Enter>`	Taste ENTER	
`<Esc>`	Taste Esc	
`<Pos1>`	Taste Pos1 oder Home	
`<Ende>`	Taste Ende oder End	
`<Einfg>`	Taste Einfg oder Ins	
`<Entf>`	Taste Entf oder Del	
`<Page_up>`	Taste PgUp oder "Bild nach oben"	
`<Page_down>`	Taste PgDn oder "Bild nach unten"	
`<Tab_left>`	Taste "Tabulator links"	
`<Tab_right>`	Taste "Tabulator rechts"	
`<Cursor_up>`	Taste "Cursor nach oben"	
`<Cursor_down>`	Taste "Cursor nach unten"	
`<Cursor_left>`	Taste "Cursor nach links"	
`<Cursor_right>`	Taste "Cursor nach rechts"	
`<Back>`	Taste "Zeichen Rückwärts"	
`<Strg>`	Taste "Strg" oder "Ctrl"	
`<Alt>`	Taste "Alt"	

Kurzbeschreibung des Programms EDT

EDT ist ein Editor zur Bearbeitung von ASCII-Dateien unter WINDOWS. EDT ist bezüglich der Bedienung und der angebotenen Funktionen weitgehend kompatibel zum BS2000-EDT.

Mit EDT können Sie

- Dateien bearbeiten (ON&FIND/PRINT/CHANGE/DELETE usw.)
- spaltenorientiert suchen und ändern (ON&:nn-mm:FIND/CHANGE)
- Zeichenfolgen austauschen (ON&CHANGE...)
  wahlweise mit Query-Modus, d.h. Abfrage vor jeder Änderung (ON&CHANGE Q ...)
- Dateien vergleichen (COMP)
- Text- und Binärdateien (z.B. EXE, DLL) einlesen und bearbeiten
- Eingabetext automatisch in Großbuchstaben umsetzen (LOW OFF-Modus)
- Dateien im Character- und Hexadezimal-Format bearbeiten
- Dateien jeder Größe und mit beliebiger Satzlänge im ASCII-, ANSI-, EBCDIC-Code bearbeiten
- über Host-Anbindung auch BS2000- und Unix-Dateien bearbeiten
- Änderungen zurücknehmen (UNDO-Funktion)
- automatisch alle EDT-Arbeitsbereiche nach alle n Sekunden sichern
- programmierbare Tasten definieren und speichern (Software Emulation der P-Tasten)
- frühere Eingaben wieder ausführen (Kommando Gedächtnis)
- spezielle PC-Features nutzen (schnelles Scrollen am Bildschirm, nach Suche Treffer farblich hervorheben, Cut & Paste, Drag & Drop)
- den EDT über INI-Dateien selbst konfigurieren
- festgelegte Kommandofolgen in einer Stapeldatei zur Ausführung bringen
- automatisch Dateien verarbeiten mit der mächtigen Prozedursprache des EDT.

EDT bearbeitet die Daten im Speicher. Die echten Daten werden erst beim Zurückschreiben auf die Festplatte geändert. Es gibt keine Einschränkungen bezüglich der Dateigröße. Die einzige Beschränkung liegt in der Größe des freien Bereichs der Festplatte für die Auslagerung der SWAP-Daten (pagefile.sys).

Installation

Um EDT zu installieren, rufen Sie das Programm SETUP.EXE auf. Das Installationsverzeichnis sowie die Programmgruppe werden vom Installationsprogramm angefordert.

Netzwerk-Installation

Sie können das Programm auch auf einem Server im Netz für alle Benutzer installieren. Nach dem ersten Aufruf wird die zentrale Datei `EDTW.INI` in den lokalen Pfad `%APPDATA%\OPG` kopiert und weiter verwendet. Weitere lokale Dateien werden nicht benötigt.

Vor der Freigabe kann vom Administrator die zentrale Datei `EDTW.INI` entsprechend den Firmen-Standards geändert werden, insbesondere die Einstellungen zu `Optionen / Einstellungen / Sicherung` (S. **106**) und `Extras / Filetransfer` (S. **136**). Die Einstellungen zum Filetransfer werden nicht von der zentralen in die lokale Datei `EDTW.INI` kopiert, so dass sie weiterhin zentral gewartet werden können.

Bei der Netzwerk-Installation ist sicherzustellen, dass für jeden Benutzer eine Lizenz erworben wird. Die Anzahl der Lizenzen bezieht sich nämlich nicht auf die max. Anzahl der Benutzer, die das Programm gleichzeitig nutzen dürfen, sondern auf die Anzahl der Benutzer, die Zugriff auf das Programm haben.

Ausgelieferte Dateien

`EDTW.EXE`	Programmroutinen des EDT.
`EDTWDeu.dll`	Deutschsprachige Programmteile.
`EDTWEng.dll`	Englischsprachige Programmteile.
`EDTCODE.EXE`	Programm zur Verschlüsselung von EDT-Prozeduren.
`EDTW-OPG.INI`	Parameter-Datei für EDT. In dieser Textdatei sind alle Standardwerte des EDT definiert. Diese Datei sollte vom Benutzer nicht geändert werden. Diese Datei dient dazu, dass bei einem Update neue Standard-Einstellungen ergänzt werden Können ohne die aktuelle Datei EDTW.INI zu überschreiben.
`EDTW.INI`	Parameter-Datei für EDT. In dieser Textdatei sind bei der Erst-Installation alle Standardwerte des EDT definiert (identisch mit EDTW-OPG.INI). Diese können jederzeit über die Menüs Ansicht und Optionen individuell angepasst werden. Bei einer Update-Installation wird diese Datei nicht schrieben. Nach dem ersten Aufruf des Programms wird die INI-Datei in das persönliche Datenverzeichnis `%APPDATA%\OPG` gesichert und für alle weiteren Aufrufe verwendet.
`EDTW.CHM`	Texte für das Hilfesystem des EDT (F1-Taste bzw. Hilfe-Button oder kontextsensitive Hilfe).
`EDT.DAT`	Zusatzinformationen zur Datei EDTW.EXE.
`EDT2.DAT`	Zusatzinformationen zur Datei EDTW.EXE.
`EDF041.TrT`	Muster Translate-Tabelle für den Filetransfer.
`FTP.BAT`	Musterprozedur für den Filetransfer mit FTP.
`CODEPAGE.TXT`	Textdatei mit der Beschreibung der verschiedenen Code-Varianten (aktive Datei).
`CODEPAGE.OPG`	Textdatei mit der Beschreibung der verschiedenen Code-Varianten (Sicherung Auslieferungsstand).
`UNICODE.TXT`	Textdatei mit der Beschreibung des Unicodes.

`COPYLIZ.EXE`	Hilfsprogramm für die Installation.

Bei der Installation mit dem SETUP-Programm ist zu beachten, dass eine bereits bestehende codepage.txt nicht überschrieben wird, da die Datei auch vom Benutzer angepasst werden kann. Soll die aktuelle `codepage.txt` übernommen werden, so muss die Datei `codepage.opg` nach `codepage.txt` kopiert werden.

Aufruf des Programms EDTW

Startup-Prozeduren

Beim Starten des EDTW werden die Standard-Prozeduren EDTINIT und EDTCMD automatisch ausgeführt. Die Prozedurdateien müssen sich im Ladeverzeichnis und/oder im APPDATA-Verzeichnis befinden.

Die Prozeduren werden in folgender Reihenfolge ausgeführt:

Kommandodatei:	EDTINITCMD im Ladeverzeichnis
Kommandodatei:	EDTINITCMD im APPDATA-Verzeichnis
Input-Datei:	EDTINIT im Ladeverzeichnis
Input-Datei:	EDTINIT im APPDATA-Verzeichnis

beim Programmstart angegebene -c *file*
beim Programmstart angegebene -i *file*

In der Kommandodatei sind nur EDT-Kommandos ohne das Zeichen '@' erlaubt. In der Input-Datei können EDT-Kommandos, Prozedur-Kommandos und Daten enthalten sein. EDT-Kommandos müssen hier mit dem Zeichen '@' beginnen.

Aufruf vom Explorer aus

Falls Sie sich in der Dateienliste des Windows-Explorers befinden, können Sie Dateien einlesen oder drucken:

Einlesen von Dateien:

a) Doppelklick auf den Dateinamen. Voraussetzung ist, dass Sie eine bestimmte Dateinamen-Extension (Dateityp) dem EDT zugeordnet haben. Falls Sie auch Dateien in Pfaden mit Leerstellen (z.B. "Eigene Dateien") einlesen wollen, ist zu beachten, dass die Angabe %1 in Anführungszeichen eingetragen wird ("%1").

   Die Dateitypen müssen im Programm Explorer Menü Ansicht→ Optionen wie folgt definiert werden:

   ohne DDE:

   Feld Anwendung für diesen Vorgang:
   `c:\programme\opg\edtw\edtw.exe "%1"`

   mit DDE:

   Feld Anwendung für diesen Vorgang:
   `c:\programme\opg\edtw\edtw.exe /dde`

   Feld DDE-Nachricht:

   Vorgang `open:` `[open("%1")]`
   Vorgang `print:` `[print("%1")]`
   Vorgang `printto:` `[printto("%1","%2","%3","%4")]`

   Der Dateityp `EDT.Dokument` mit der Endung "`edt`" wird bei der Installation automatisch erzeugt. Hier werden ebenfalls die o. a. Einträge verwendet. Dieser Dateityp enthält zusätzlich den Vorgang `Start`, der das Starten von EDT-Prozeduren regelt.

b) Klicken Sie mit der rechten Maustaste auf eine Datei und wählen Sie die Option "Öffnen mit ..." aus. Im folgenden Dialog kann das Programm EDT ausgewählt werden. Wenn Sie in diesem Fall die Option "Dateien dieses Typs immer mit diesem Programm öffnen" wählen und für den Dateityp einen Namen vergeben, wird in der Registry ein Dateityp erzeugt (Variante ohne DDE), den Sie dann entsprechend anpassen können.

c) Klicken Sie auf die Datei und ziehen Sie die Datei in das EDT-Fenster. Falls das EDT-Fenster nicht sichtbar ist, ziehen Sie zuerst die Datei auf das Icon in der Taskleiste. Daraufhin wird das EDT-Fenster angezeigt und Sie können die Datei in das EDT-Fenster ziehen.

d) Doppelklick auf Dateien mit der Erweiterung `edt`, die EDT-Prozeduren enthalten. In diesem Fall wird der EDTW mit der Option `-i%1` gestartet, d.h. die ausgewählte Datei wird als EDT-Prozedur gestartet. Der Dateityp wird bei der Installation erzeugt.

Drucken von Dateien mit den Aufbereitungsfunktionen des EDTW (Menü `Optionen / Einstellungen / Drucken` (S. 114)):

a) Klicken Sie mit der rechten Maustaste auf eine Datei und wählen Sie die Option "Drucken" aus. Voraussetzung ist, dass Sie in dem entsprechenden Dateityp den Vorgang `print`, wie oben beschrieben, definiert haben.

b) Klicken Sie auf die Datei und ziehen Sie die Datei auf ein Druckersymbol auf Ihrem Desktop oder in einem Verzeichnis. Voraussetzung ist, dass Sie in dem entsprechenden Dateityp den Vorgang `printto`, wie oben beschrieben, definiert haben.

Aufruf über Programm FServer:

Mit dem BS2000-Kommando EDTW, als Teil des Softwareprodukts OPGCOM (Download: ftp://ftp1.opg.de/opgcom/Manopgcom.EXE), können Dateien vom BS2000 zu einem Windows-PC übertragen und in den EDT für Windows eingelesen werden. Die Daten werden im EBCDIC-Code zur Verfügung gestellt und können mit dem Kommando `WRITE` wieder ins BS2000 zurück übertragen werden. Das Windows-Programm FServer muss geladen sein (Download: http://www.opg.de/download/opgcom/setupfserver.exe).

Die Datenübertragung kann auch über den Actioncode EDTW des Programms CFS gestartet werden.

Weitere Informationen siehe Kommando `WRITE` (S. 270)

Aufruf über den Browser für vom URLServer gesendete Dateien:

Mit dem BS2000-Programm URLServer, als Teil des Softwareprodukts OPGCOM (Download: http://www.opg.de/produkte/frames/opgcom.pdf), können für den Browser BS2000-Dateien zur Verfügung gestellt werden, für die ein Update mit dem EDTW zulässig ist. Nach dem Download durch den Browser werden die Daten aufgrund der Verknüpfung der Dateitypen in den EDT für Windows eingelesen. Die Daten werden im EBCDIC-Code zur Verfügung gestellt und können mit dem Kommando `WRITE` wieder ins BS2000 zurück übertragen werden.

Weitere Informationen siehe Kommando `WRITE` (S. 270)

Aufruf durch Doppelklick auf das EDT-Icon oder aus einer Shell

In der Karteikarte "Verknüpfung" (erreichbar mit der rechten Maustaste/Eigenschaften) des EDT-Icons bzw. in der Kommandozeile einer Shell (DOS-Box) oder in einem Shell-Script können folgende Parameter angegeben werden:

EDTW.EXE [*datei datei ...*| `-STDIN`] [`-c` *cmdfile*] [`-e`] [`-l`] [`-p` *inifile*] [`-m` [*edt-group*]] [`-LEng`|`-LDeu`] [`-i` *procfile* [*par1 .... parn*]]

Beim Aufruf des Programms können Sie bis zu 32 Dateien angeben, die dann in die Arbeitsbereiche 0 bis 31 eingelesen werden. Die Schalter und Dateinamen können in beliebiger Reihenfolge angegeben werden mit Ausnahme des Schalter "-i", der immer als letzter Parameter erscheinen muss.

Zusätzliche Angaben zu einem Schalter, z.B. `-r` *rlen* oder `-p` *inifile* können durch Blank getrennt oder ohne Zwischenraum angegeben werde, also:

```
-cfile -r10000 -s16 -pfile -mgroup -Leng -Ldeu -ifile
```
oder
```
-c file -r 10000 -s 16 -p file -m group -L eng -L deu -i file
```

Falls Sie das Programm ohne Parameter aufrufen, wird nach dem Laden das leere Datenfenster des Arbeitsbereichs 0 angezeigt. Sie können dann z.B. eine neue Datei erstellen oder eine zu bearbeitende Datei mit dem Kommando `READ` einlesen.

datei In einen Arbeitsbereich des EDT wird die angegebene Datei geladen. Sie können bis zu 32 Dateien, durch Leerzeichen getrennt, angegeben werden. Enthält ein Dateinamen Leerzeichen, so ist er in Hochkommas einzuschließen. Es kann auch eine entfernte Datei mit allen notwendigen Parametern angegeben werden, z.B. "host1 r=datei1"

`-STDIN` Die Daten werden von der Systemdatei STDIN gelesen. Das Einlesen von STDIN ist nur möglich, wenn beim Laden eine STDIN - Datei mittels des Pipe-Zeichens zugewiesen wird. In der Regel wird diese Variante nur bei EDT-Prozeduren zum Einsatz kommen. Das Einlesen der STDIN-Datei kann auch mit dem Kommando `READ` (S. **220**) erfolgen.

-i *procfile par1 par2.... parn* oder

-i *procfile (par1,par2 .... parn)*

Prozedurdatei. In der Prozedurdatei können alle EDT-Kommandos einschließlich der Kommandos der Prozedursprache angegeben werden. Die Prozedurdatei wird sofort nach dem Laden des EDT ausgeführt.

Ist kein Pfadname angegeben, wird die Prozedurdatei in folgenden Pfaden gesucht:

1. aktuelles Verzeichnis
2. Installationsverzeichnis von EDTW.EXE
3. alle Pfade der Umgebungsvariablen PATH

Hinter dem Namen der Prozedurdatei können maximal 32 Parameter, jeweils durch Leerzeichen getrennt bzw. in Klammern und durch Kommas getrennt (wie Kommando `INPUT` (S. **206**)), für die Prozedurdatei angegeben werden. Enthält ein Parameter Leerstellen, so ist er in Hochkommas einzuschließen. Die Parameter müssen in der `PARAMS`-Anweisung (erste Zeile in der Prozedurdatei) definiert werden. Ein Beispiel für eine Prozedurdatei folgt weiter unten.

Enthält der Name der Prozedurdatei oder ein Parameter Leerzeichen, so ist er in Hochkommas einzuschließen (z.B. `EDTW.EXE datei1 -i'Eigene Dateien\Test1' par1 par2 'Parameter mit Leerzeichen'`).

Enthält die Prozedurdatei im Namen die Endung `.edt`, so kann die Prozedur auch im Explorer mit Doppelklick gestartet werden.

-c*cmdfile* Stapeldatei mit gültigen EDT-Kommandos, die nach dem Aufruf von EDT ausgeführt wird. Bevor die Kommandos ausgeführt werden, werden ev. angegebene Dateien eingelesen, dann wird ein neuer Arbeitsbereich erzeugt. Gegebenenfalls muß mit dem Kommando 0 bis 32 der zutreffende Arbeitsbereich aktiviert werden. Werden in der Kommandozeile Dateinamen angegeben, werden diese Dateien vor der Verarbeitung eingelesen. Kommandos der Prozedursprache sind hier nicht zulässig; diese können nur in Input- oder Prozedurdateien angegeben werden (siehe Schalter -i bzw. Kommando INPUT).

Erscheint im Laufe der Abarbeitung der Kommandos ein Fenster mit einer Rückfrage, müssen Sie diese mit einer gültigen Eingabe beantworten. Ein Beispiel für eine Kommandodatei folgt weiter unten.

Dieser Prozedurmodus wird nur noch aus Kompatibilitätsgründen unterstützt. Es sollte statt dessen der Schalter -i verwendet werden.

-e Beim Starten des EDT wird kein leeres Datenfenster erzeugt.

-l Beim Starten des EDT wird das Lizenzfenster nicht ausgegeben.

-LEng|-LDeu Das Programm kann wahlweise in englischer oder deutscher Sprache (Menüs, Dialogboxen, Meldungen usw.) gestartet werden. Fehlt dieser Parameter, wird beim ersten Start die Sprache von der Windows-Einstellung übernommen und in der Datei EDTW.INI gespeichert. Bei den folgenden Aufrufen wird die Einstellung der INI-Datei verwendet, die über das Menü Optionen/Verschiedene Optionen (S. 127) geändert werden kann.

-p *inifile* Alle Optionen, die mit den Kommandos der Menüzeile eingestellt werden können, werden in der Datei EDTW.INI im persönlichen Datenverzeichnis %APPDATA%\OPG gespeichert. Mit diesem Schalter kann eine andere INI-Datei zugewiesen werden. Wird eine nicht vorhandene Datei angegeben, so wird die INI-Datei beim Beenden des EDT erzeugt.

Insgesamt können max. 3 INI-Dateien in folgenden Prioritäten verarbeitet werden:

1. Datei (beliebiger Name), die mit dem Schalter -p zugewiesen ist.
2. EDTW.INI im persönlichen Datenverzeichnis %APPDATA%\OPG, bzw. falls die Variable APPDATA nicht vorhanden ist, im Windows-Systemverzeichnis C:\WINDOWS oder C:\WINNT (wird automatisch verwendet, falls vorhanden)
3. EDTW.INI auf dem Ladeverzeichnis (wird automatisch verwendet, falls vorhanden).

Die INI-Dateien werden pro Option in der o. g. Reihenfolge durchsucht. Sind die gleichen Parameter in mehreren INI-Dateien enthalten, gelten die Parameter aus der zuerst gelesenen Datei, d.h. die INI-Datei, die mit dem Schalter -p zugewiesen wurde, hat Vorrang vor der EDTW.INI aus dem APPDATA-Verzeichnis und die EDTW.INI auf dem APPDATA-Verzeichnis hat Vorrang vor der EDTW.INI auf dem Ladeverzeichnis. Dieser Mechanismus gilt nicht für die Filetransfer-Profile. Auch Filetransfer-Profile mit dem gleichen Namen in mehreren INI-Dateien können verarbeitet werden, weil in der Auswahlbox des Menüs Extras/Filetransfer (S. 136) zusätzlich der Name der INI-Datei für die Identifizierung des Filetransfer-Profils verwendet wird.

Falls die Einstellung "Optionen beim Programm-Ende sichern" (S. 131) aktiv ist, werden beim Beenden des EDT alle Einstellungen in die EDTW.INI des APPDATA-Verzeichnisses bzw. falls vorhanden in die EDTW.INI, die mit dem Schalter -p zugewiesen wurde, gespeichert. Ist nur die EDTW.INI im Ladeverzeichnis vorhanden, wird eine neue EDTW.INI im APPDATA-Verzeichnis erzeugt.

-m [*edt-group*] Mehrere Aufrufe des EDT in einem gemeinsamen Fenster verarbeiten

Das Programm EDT wird nur einmal bzw. pro EDT-Gruppe nur einmal geladen. Wird beim zweiten oder weiteren Aufruf ein Dateiname angegeben, so wird diese Datei in den nächsten freien Arbeitsbereich des bereits geladenen EDT eingelesen. Durch die wahlweise Angabe einer EDT-Gruppe kann z.B. ein EDT für alle Sourcen und ein zweiter EDT für alle Testdateien benutzt werden. Als *edt-group* kann ein 1-16 Stellen langer alphanumerischer Wert angegeben werden.

Erfolgt der Aufruf des EDT ohne die Option −m, sei es von der Shell aus, über einen Doppelklick auf das EDT-Icon oder vom Explorer, wird das Programm immer neu geladen.

Aufruf des EDT aus einer BAT-Datei mit dem Kommando START

Es ist zu beachten, dass nach dem Start eines Windows-Programms sofort der nächste Befehl der BAT-Datei ausgeführt wird. Wenn dies vermieden werden soll, so sollte das DOS-Kommando `START` mit dem Parameter `/WAIT` verwendet werden. Der Parameter `/MIN` bewirkt, dass das Programm EDT nur mit einem minimierten Icon gestartet wird.

Beispiel:

```
start /wait /min c:\programme\opg\edtw\edtw.exe dat1 -iproc.dat
```

Beispiele:

```
edtw test.dat -iproc.edt 95.01.01 10:00 'Prozedur beendet'
```

Die Datei test.dat wird in den Arbeitsbereich 0 des EDT eingelesen. Sofort danach werden die in der Datei proc.dat stehenden Anweisungen ausgeführt. Die Parameter "95.01.01" "10:00" und "Prozedur beendet" werden als &date, &time und &t an die Prozedur übergeben.

Inhalt der Datei test.dat:

```
99.01.01 22:00
90.03.20 09:00
94.10.12 11:00
```

Inhalt der Datei proc.edt:

```
@params &date,&time,&t   Definition Parameter
@proc1                   Zu Arbeitsbereich 1 wechseln
@@if !:1-8: < '&date' goto del    Zeileninhalt kleiner als 95.01.01 ?
@@if !:10-15: < '&time' goto del  Zeileninhalt kleiner als 10:00 ?
@@goto end               unbedingter Sprung zu Sprungziel end
@@:del                   Sprungziel del
@@del !                  aktuelle Zeile löschen
@@:end                   Sprungziel end
@end                     Zu Arbeitsbereich 0 wechseln
@do1,!=%,$,1             Aufruf Prozedur in Arbeitsbereich 1
@w o                     Datei zurückschreiben ohne Rückfrage
@msgbox '&t'             Message-Box ausgeben.
@h                       EDT wird beendet (ohne @h würde in den
                         Dialogmodus umgeschaltet)
```

```
edtw test.dat -ckommando.txt
```

Die Datei test.dat wird in den Arbeitsbereich 0 des EDT eingelesen. Danach wird ein neuer Arbeitsbereich erzeugt und die in der Datei kommando.txt stehenden Anweisungen ausgeführt.

Inhalt der Datei kommando.txt:

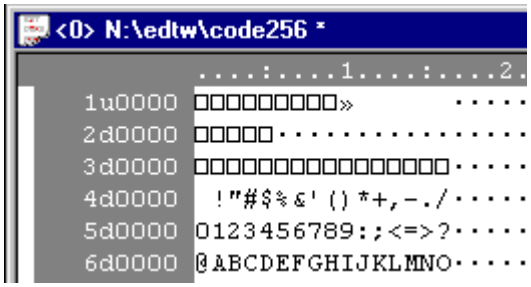
`0`	Umschalten auf Arbeitsbereich 0
`on&ca'112465't'992465'`	Zeichenfolgen '112465' durch '992465' ersetzen
`on&:1-1:f'A'd`	Zeilen löschen, die in Spalte 1 'A' enthalten
`on&f'neu:'copyto(1)`	Zeilen mit String 'neu' in Arbeitsbereich 1 kopieren
`1`	Zu Arbeitsbereich 1 wechseln
`write'daten.neu'o`	Arbeitsbereichs 1 in die Datei daten.neu schreiben
`0`	Zu Arbeitsbereich 0 wechseln
`write o`	Arbeitsbereich 0 in Datei test.dat schreiben (Overwrite)
`halt`	EDT beenden

Dateistruktur

Satzstrukturierte Dateien mit Satzende-Kennzeichen

Im EDT können Sie alle Binär-Dateien sowie ASCII- und ANSI-Dateien mit unbegrenzter Satzlänge bearbeiten. Die Sätze können mit dem Zeichen CR+LF (MS-DOS-Format: Carriage Return + Line Feed = `X'0D0A'`) oder nur LF (UNIX-Format) abschließen. Leersätze, die nur aus dem Zeichen CR+LF (`X'0D0A'`) bzw. LF (`X'0A'`) bestehen, sind zulässig.

Enthält eine Datei sowohl X'0A' als auch X'0D0A' als Satzende-Kennzeichen, wird beim Einlesen eine Dialogbox ausgegeben, in der gewählt werden kann, welches Format für den Arbeitsbereich gelten soll (MS-DOS / Unix / gemischtes Format). Wird das gemischte Format gewählt, so wird bei jeder Zeile des Arbeitsbereichs in der Zeilennummer statt des Punktes das Satzformat dargestellt:



d	MS-DOS/WINDOWS (`X'0D0A'`)
u	Unix (`X'0A'`)
*	Satz ohne Satzende-Kennzeichen

Mit dem Kommando `ERS` (S. **191**) kann das Format jeden Satzes oder für die ganze Datei geändert werden.

Das Satzende-Kennzeichen eines Satzes kann auch mit dem Kommando `E` der Markierungsspalte (S. **175**) erzeugt oder gelöscht werden.

Mit dem Kommandos `ON&FIND DOS/UNIX/NO` (S. **213**) können Sätze mit bestimmten Satzende-Kennzeichen gesucht werden.

Beim Zurückschreiben des Arbeitsbereiches in die Datei werden die Daten in dem Format geschrieben, das beim Einlesen festgestellt wurde. Beim "gemischten Format" bedeutet dies, dass jeder Satz mit dem Satzende-Kennzeichen, das beim Lesen festgestellt wurde, zurückgeschrieben wird. Beim Kommando `WRITE` (S. **261**) kann mit der Option D (MS-DOS) oder X (Unix) explizit ein anderes Format gewählt werden.

Binärdateien

Sätze mit einer Länge über 32.767 Bytes bzw. Dateien, die keine erkennbare Satzstruktur besitzen (z.B. EXE-Files), werden in Segmente von 70 Bytes aufgeteilt und beim Zurückschreiben in die Datei wieder verkettet. Mit der Option `RECORD` zum Kommando `READ`

(S. 220) kann auch das Record-Format erzwungen werden.

Fremdes Satzformat

Dateien mit einem Satzformat aus einem anderen Betriebssystem (z.B. UNIX oder BS2000 feste oder variable Satzlänge) können mit dem Kommando REFORMAT (S. 234) verarbeitet werden. Die Datei wird damit in das ANSI-Format umgewandelt. Sie können auch Dateien in EBCDIC- oder ASCII-Codierung einlesen und in das interne ANSI-Format umwandeln. Siehe Kommando CODE (S. 182).

Unterschiede und Erweiterungen zum BS2000-EDT

- Verarbeitung von UNICODE-Dateien in den Codierungen UTF-8 und UTF-16 (Big-Endian oder Little-Endian). Die Anzeige aller Arbeitsbereiche, unabhängig von der Codierung, erfolgt immer in UNICODE. UNICODE-Arbeitsbereiche können in andere Codes konvertiert werden oder mit anders codierten Arbeitsbereichen verglichen werden. Bei der hexadezimalen Anzeige können wahlweise 4 Halbbytes oder nur die 2 rechtsbündigen Halbbytes angezeigt werden. Daten, die in ISO-8859, EBCDIC oder in einem beliebigen Benutzercode codiert sind, können unverändert im Originalcode editiert werden.
- Es können Sätze bis zu einer Satzlänge von 32.767 Bytes verarbeitet werden.
- Mit dem Kommando UNDO bzw. der Schaltfläche  können vorhergehende Aktionen rückgängig gemacht werden, unabhängig davon, ob sie über ein Kommando oder die Markierungsspalte ausgelöst wurden oder ob es sich um Änderungen im Datenfeld handelt.
- EDT kann über Parameter so eingestellt werden, dass jeweils nach Ablauf einer vorgegebenen Anzahl von Sekunden automatisch alle Arbeitsbereiche des EDT gesichert werden (Autosave).
- Im Open-Modus können Dateien bis zu 1.024 GB eingelesen werden. Der Arbeitsspeicher wird dabei nur für Verwaltungs-Informationen der Daten verwendet. Die eigentlichen Daten werden bei Bedarf direkt von der Originaldatei gelesen. Neue bzw. geänderte Daten werden in einer temporären Datei gespeichert.
- In einen Arbeitsbereich können beliebig viele Dateien eingelesen und gemeinsam editiert werden (siehe Kommando READ (S. 224)) und Menü Datei / Öffnen mehrfach (S. 58), z.B. Austausch eines Strings in allen Primärprogrammen. Anschließend können mit dem Kommando REWR (S. 237) alle Dateien in die Ursprungsdateien oder wahlweise auch in neue Dateien zurückgeschrieben werden. Mit dem Kommando SHOW DIR wird ein Inhaltsverzeichnis aller Dateien eines Arbeitsbereichs erzeugt.
- In allen Strings zu den EDT-Kommandos können spezielle EDT-System-Variablen (S. 344) verwendet werden, mit denen z.B. der aktuelle Dateiname, der Pfad, das Datum oder die Uhrzeit als String oder Teil eines Strings angegeben werden können. Bei allen String-Parametern können mehrere Zeichenfolgen verknüpft werden (S. 341).
- Mit dem Kommando VIEW (S. 260) können mehrere Sichten auf einen Arbeitsbereich aktiviert werden, z.B. wird in der VIEW 0 der Beginn eines Primärprogramms mit der Parameterbeschreibung angezeigt, in VIEW 1 der Funktionsteil und in VIEW 2 wird der Datenteil angezeigt.
- Integrierter Filetransfer: Filetransfer zu anderen Hosts (BS2000, z/OS, Unix, Windows) ohne FT-Software auf dem PC. Auf dem anderen Host muss openFT bzw. FTP vorhanden sein. Im Menü Extras/Filtransfer können FT-Profile definiert werden, die in der EDTW.INI gespeichert werden. Der Filetransfer wird mit den Kommandos READ (S. 226) und WRITE (S. 264) gestartet. Bei diesen Kommandos und bei den Kommandos FILE (S. 192) und FSTAT (S. 197) kann jeweils statt des Dateinamens der FT-Profilname und der BS2000-Dateiname angegeben werden.
- Filetransfer zum BS2000-Host über das Programm FILESEND oder URLServer aus dem Programmpaket OPGCOM. Der Filetransfer wird vom Host initiiert mit dem Kommando EDTW bzw. Actioncode EDTW des Programms CFS oder die Daten werden über den Browser eingelesen. Die Daten können mit dem EDT-Kommando WRITE (S. 270) zurückgeschrieben werden. Im Gegensatz zu allen anderen FT-Methoden können auch **ISAM-Dateien, PAM-Dateien und delta-gespeicherte LMS-Elemente** editiert werden. BS2000-Zip-Dlemente können nur gelesen werden.
- Bei den Kommandos ON ..... ist zusätzlich der Parameter Q (Query) möglich. Diese Option bewirkt für jede Zeile eine Abfrage an den Benutzer, ob die Aktion (Change, Delete, Print, Copy) durchgeführt werden soll. Anstelle ON..FIND kann auch die Option ON..SEARCH (S. 212) mit der CFS-Syntax (beliebig viele Suchbegriffe mit "und/oder" verknüpft) angegeben werden.
- EDT kann über eine Parameterdatei so eingestellt werden, dass er genauso wie im BS2000 reagiert (Standard). Zusätzlich können besondere "PC-Features" genutzt werden (siehe Kommando EDIT WORD (S. 190)).

- Beim Einfügen von Zeichen in eine Zeile gehen die über den rechten Bildschirmrand hinaus geschobenen Zeichen nicht verloren.
- Weiterverarbeitung der mit dem Kommando FSTAT (S. 197) erzeugten Dateienliste.
- Mit dem Kommando FSTAT (S. 197) kann auch eine Liste von Elementen eines ZIP-Archivs erzeugt werden.
- Das Kommando SETF (S. 246) kann auch in Prozeduren aufgerufen werden. Damit ist es möglich, während des Prozedurablaufs bestimmte Daten anzuzeigen.
- Die Daten können in ASCII, ANSI, EBCDIC oder UNICODE editiert und konvertiert werden (siehe Menü Ansicht/Code-Anzeige (S. 94) und Kommando CODE (S. 181)). Die Konvertierung erfolgt mit dem Menü Funktionen/Code-Umsetzung (S. 83) bzw. dem Kommando CODE (S. 181).
- Verschlüsselung von EDT·Prozedurdateien (S. 357).
- Neben den Integer-Variablen stehen auch Float-Variablen (S. 337) für Gleitpunkt-Berechnungen zur Verfügung.
- Es werden im LOWER OFF-Modus auch Kleinbuchstaben wahlweise am Bildschirm angezeigt. Im BS2000 werden im LOWER OFF-Modus Kleinbuchstaben als Schmierzeichen dargestellt.
- Falls beim Einlesen, Neunumerieren oder Kopieren die maximale Zeilennummer überschritten wird, gehen im Gegensatz zum BS2000 keine Sätze verloren. Der Arbeitsbereich wird automatisch neu numeriert.
- Es stehen 32 Arbeitsbereiche (0…31) und je 100 Line-, Integer-, Float- und String-Variablen zur Verfügung. In jeder String-Variablen können bis zu 32.767 Zeichen gespeichert werden. Bei Integer-Variablen sind Zahlen von -2^{63} bis $2^{63}-1$ (64Bit=8.388.607 Terrabyte) zulässig.

Markierungsspalte:

- Auch bei Verwendung der Markierungen C oder M können die Markierungen A, B oder O mehrfach angegeben werden. Nach Ausführung der Funktion für diese Maske werden die Markierungen C und M gelöscht.
- Mit der Markierung F kann eine Zeile markiert werden. Die Zeilennummer wird dann hervorgehoben und die Zeile wird so behandelt, als wäre sie bei einem ON..FIND-Kommando gefunden worden. Mit der Markierung N kann dieser Status wieder zurückgesetzt werden.
- Mit der Markierung E können Zeilen zu einer Zeile mit unbegrenzter Länge verkettet werden.
- Mit der Markierung S (Split) können Zeilen aufgeteilt werden. Im BS2000 bewirkt die Markierung S die Positionierung des Datenfensters.
- In einem Arbeitsbereich mit einer Dateienliste, die mit dem Kommando FSTAT erzeugt wurde, kann mit der Markierung 1 - 9 die Datei in den jeweiligen Arbeitsbereich eingelesen werden. Handelt es sich bei der Zeile um ein Unterverzeichnis oder ein ZIP-Archiv, wird automatisch das FSTAT-Kommando aufgerufen und das Ergebnis in den entsprechenden Arbeitsbereich übertragen.

zusätzliche EDT-Kommandos:

- Kommando BEEP (S. 178) zur Ausgabe eines akustischen Signals.
- Kommando CHDIR/CD (S. 179): Wechseln Arbeitsverzeichnis.
- Kommando CAT (S. 178) und CUT (S. 186) zum Verketten bzw. Aufteilen von Stringvariablen.
- Kommando CODE (S. 181) zur Code-Konvertierung und Code-Darstellung.
- Kommando COPYFILE (S. 186) zum Kopieren von Dateien.
- Kommando DELETE All VARIABLES (S. 188) zum Löschen aller Variablen.
- Kommando DELETE MULTIPLE (S. 187): Löschen gleiche Zeilen bzw. Zeilen mit gleichem Inhalt in bestimmten Spalten.

- Kommando DIALOGBOX (S. 275): Ausgabe der Dialogbox ÖFFNEN oder SAVE AS zur Anforderung eines Dateinamens.
- Kommando MKDIR (S. 209) zum Erzeugen von Verzeichnissen.
- Kommando MOVEFILE (S. 210) zum Umbenennen/Verschieben von Dateien.
- Kommando ON..SEARCH (S. 212): Ähnliche Funktionalität wie ON..FIND aber es können Suchbegriffe mit der CFS-Syntax (beliebig viele Suchbegriffe mit "und/oder" verknüpft) angegeben werden.
- Kommando RMDIR (S. 238) zum Löschen von Verzeichnissen.
- Kommando S (S. 212): Suchen im Arbeitsbereich ab der ersten angezeigten Zeile mit der CFS-Syntax (beliebig viele Suchbegriffe mit "und/oder" verknüpft).
- Kommando SETENV (S. 245) zum Setzen und Anzeigen von Umgebungsvariablen.
- Kommando SHOWNIL (S. 247): Darstellung des NIL-Zeichens in den Daten als Punkt oder Blank. Diese Option ist insbesondere zur Bearbeitung von Unicode-Daten gedacht.
- Kommando SORT (S. 248): Mehrere Sortierbegriffe möglich. Option REVERSE zum Kommando (Sortieren der Zeilen in einem Arbeitsbereich in umgekehrter Reihenfolge).
- Kommando STRIP (S. 251): Entfernen Leerstellen und Tabulatorzeichen.
- Kommando STT/TTS (S. 251): Leerstellen in Tabulatorzeichen umwandeln und umgekehrt (Spaces to Tab bzw. Tab to Spaces).
- Kommando WAIT (S. 261): Warten von n Sekunden.
- Kommando WINDEF (S. 301): Zeile für Dialogbox definieren.
- Kommando WINGEN (S. 307): Erweiterte Dialogbox erzeugen und ausgeben.
- Kommando WINOUT (S. 302): Benutzerdefinierte Dialogbox ausgeben.
- Kommando WINSIZE (S. 303) zum Ändern der Arbeitsbereichs-Fenster.

Kommandos für Prozeduren:

- Kommentare (S. 274) rechts neben EDT·Kommandos in Prozedurdateien.
- Bei IF- und GOTO-Anweisungen kann als Sprungziel eine Zeilennummer oder ein Name angegeben werden. Der Name kann auch in einer Stringvariablen stehen. Diese beiden Kommandos können auch in der Ebene 0 einer INPUT-Prozedur verwendet werden.
- Kommando IF (S. 279): Bei allen Varianten ist als True-Action die Angabe GOTO RETURN oder ":*text*" zulässig. Bei IF .TRUE./.FALSE./.EMPTY kann zusätzlich ein Arbeitsbereich angegeben werden.
- Zusätzliche Varianten des Kommandos IF (S. 280):

  Kommando IF *line-var* = SELECTED: Prüfen, ob eine Zeile markiert ist.
  Kommando IF *line-var* = EXIST: Prüfen, ob eine Zeile existiert.
  Kommando IF WINGEN *n* = EXIST: Prüfen, ob ein WINGEN-Fenster noch existiert.

- Kommando INPUT (S. 206): Kommentare können in INPUT-Prozeduren am Ende jeder Prozedurzeile, getrennt durch doppelte Strichpunkte ";;" angegeben werden (S. 285). Die geschachtelte Nutzung von INPUT-Prozeduren wird unterstützt. Dadurch entfällt in vielen Fällen der dynamische Aufbauen von Prozedurebenen.

- Zusätzliche Varianten des Kommandos SET:

  mehrere Datums- und Zeitvarianten (S. 299),
  SET-Kommandos für Float-Variablen (S. 292),
  SET-Kommandos SET #S*n*=C#I*nd* (S. 293) und SET #L*n*=C#I*n* (S. 298) zum Aufbereiten von Zahlen mit zusätzlichen Optionen (Tausender-Punkte, variable Länge ohne führende Nullen, Zahlen > 2 GB) usw.),
  Mit dem Kommando SET (S. 290) können alle vier Grundrechenarten ausgeführt werden (BS2000 nur Plus und Minus).
  SET *intvar* = DAY (S. 292): Nummer des Wochentages.
  SET *intvar* = RECORDS (S. 292): Anzahl der Sätze des aktuellen Arbeitsbereichs.
  SET CONVX/CONVC *string* (S. 297): Anzahl der Sätze des aktuellen Arbeitsbereichs.

2 Tastenbelegung

Funktionstasten

F1	Aktiviert das Hilfe-System. Es wird das Thema zu dem Feld angezeigt, in dem sich der Cursor befindet.
<Shift> F1	Der Mauszeiger wechselt zu einem Pfeil mit Fragezeichen. Sobald Sie an irgendeiner Stelle im EDT-Fenster mit der linken Maustaste klicken, werden zu diesem Thema die Hilfe-Informationen angezeigt. Sie können auf die Menüzeile, auf Icons, auf Buttons oder sonstige Felder (überschreibbare oder nicht überschreibbar) klicken.
F2	Gibt den ganzen Bildschirm zum Überschreiben frei. Wirkt wie die Taste F2 im BS2000.
F3	Positioniert auf die nächste Zeile, die mit dem Kommando ON...FIND bzw. Markierung F markiert wurde. Entspricht der Taste F3 im BS2000.
F4	Positioniert rückwärts auf die vorhergehende mit dem Kommando ON...FIND bzw. Markierung F markierte Zeile. Entspricht der Eingabe -F3 im BS2000.
<Strg> F4	Aktuelles Arbeitsbereichs-Fenster schließen. Falls sich im Arbeitsbereich noch nicht gesicherte Daten befinden, erfolgt eine Rückfrage, ob die Daten gesichert werden sollen.
<Alt> F4	EDT beenden. Falls sich in einem Arbeitsbereich noch nicht gesicherte Daten befinden, erfolgt für jeden Arbeitsbereich eine Rückfrage, ob die Daten gesichert werden sollen.
F5	Aktiviert das Kommandogedächtnis und zeigt das letzte Kommando an (rückwärts blättern im Kommandogedächtnis). Bei wiederholter Betätigung der Taste wird das vorletzte, vorvorletzte Kommando angezeigt. Ausführliche Beschreibung siehe Kapitel 13 (S. 351).
F6	Zeigt das nächste (zeitlich spätere) Kommando im Kommandogedächtnis (vorwärts blättern). Ausführliche Beschreibung siehe Kapitel 13 (S. 351).
F7	Führt die in der programmierten Einzeltaste SINGLE_PK festgelegte Tastenkombination aus. Beschreibung siehe Kapitel 15, Programmierbare Tasten (S. 355).
F8	Tastenkombination der programmierten MULTI_PK Taste abrufen. Beschreibung siehe Kapitel 15, Programmierbare Tasten (S. 355).
F9	Positioniert auf die erste Trefferzeile (Kommando ON...FIND siehe S. 210) auf der nächsten Bildschirmseite.

Shortcuts

<Strg> "A"	Alles markieren (S. 79)
<Strg> "C"	Markierte Daten in Zwischenablage kopieren (S. 78)
<Strg> "F"	Find: Zeichenfolge suchen (S. 67)

<Strg> "H"	Datei anhängen an Daten des aktuellen Arbeitsbereichs (S. 59)
<Strg> "K"	Suchen und Kopieren der Trefferzeilen (S. 75)
<Strg> "L"	Suchen und Löschen der Trefferzeilen bzw. des Suchbegriffs (S. 77)
<Strg> "N"	Neue Datei öffnen (S. 57)
<Strg> "O"	Bestehende Datei öffnen (S. 57)
<Strg> "P"	Drucken (S. 64)
<Strg> "R"	Replace: Zeichenfolge ersetzen (S. 72)
<Strg> "S"	Datei sichern (S. 61)
<Strg> "T"	Den mit dem Kommando TABS (S. 253) definierten Hardware-Tabulator ein- bzw. ausschalten (gleiche Wirkung wie TABS ON und TABS OFF).
<Strg> "U"	Daten ab dem Cursor mit den Daten aus Zwischenablage überschreiben (S. 80)
<Strg> "V"	Daten aus der Zwischenablage ab dem Cursor einfügen (S. 80)
<Strg> "5"	Wenn sich der Cursor vor bzw. auf einer öffnenden oder schließenden Klammer (rund "()", eckig "[]" oder geschweift "{}") befindet, wird die dazugehörende schließende bzw. öffnende Klammer gesucht. Die gesamten Daten von der öffnenden bis zur schließenden Klammer werden markiert. Alle in diesem Bereich enthaltenen Klammern der gleichen Art werden mit einer FIND-Markierung versehen. Diese Funktion ist auch im Kontextmenü der rechten Maustaste enthalten.
<Alt> <Back>	Undo - macht letzte Aktion rückgängig (S. 66)
<Shift> <Del>	Ausschneiden (S. 79)
<Strg> <Einfg>	Markierte Daten in Zwischenablage kopieren (S. 78)
<Shift> <Einfg>	Daten aus der Zwischenablage ab dem Cursor einfügen (S. 80)
<Strg> "X"	Ausschneiden (S. 79)
<Strg> "Y"	Redo - macht letzten Undo rückgängig (S. 67)
<Strg> "Z"	Undo - macht letzte Aktion rückgängig (S. 66)
<Shift> <Strg> "Z"	Dialogbox mit den bisherigen Aktionen zur Auswahl für Undo (S. 66)
<Strg> <Cursor_down>	Unabhängig von der Cursor-Position wird das Bildschirmfenster um eine Zeile nach unten verschoben (wie Kommando "+1").
<Shift> <Cursor_down>	Markieren des Textes in Richtung Dateiende für Editier-Funktionen (Kopieren, Löschen, Verschieben, Ausschneiden).
<Strg> <Cursor_up>	Unabhängig von der Cursor-Position wird das Bildschirmfenster um eine Zeile nach oben verschoben (wie Kommando "-1").
<Shift> <Cursor_up>	Markieren des Textes in Richtung Dateianfang für Editier-Funktionen (Kopieren, Löschen, Verschieben, Ausschneiden).
<Strg> <Ende>	Verschiebt den angezeigten Ausschnitt an das Ende des Arbeitsbereichs (wie Kommando "++").
<Alt> <Entf>	Löscht ab der Cursor-Position alle Zeichen bis Zeilenende. Steht der Cursor in der ersten Spalte, so werden alle Zeichen der Zeile gelöscht. Die Zeile bleibt jedoch als Leerzeile erhalten.
<Strg> <Entf>	Löscht die ganze Zeile. Die Zeile bleibt als Leerzeile erhalten.
<Strg> <Page_down>	Positioniert den Cursor auf das letzte Feld am Bildschirm.

<Strg> <Page_up>	Positioniert den Cursor auf das erste Feld am Bildschirm (Zeile 1, Spalte 1).
<Strg> <Pos1>	Positioniert den Bildschirmausschnitt auf den Anfang des Arbeitsbereichs (wie Kommando "--").
<Strg> <Tab_right>	Aktivieren nächstes EDT-Datenfenster in der Reihenfolge der Arbeitsbereichs-Nummer.

Sonstige Tasten

<Back>	Taste "Backspace" löscht von rechts nach links. Befindet sich der Cursor im EDIT WORD Modus am Anfang einer Zeile, so bewirkt die Taste eine Verkettung mit der vorhergehenden Zeile.
<Cursor_up>	Taste "Cursor nach oben", verschiebt den Cursor eine Zeile nach oben.
<Strg> <Cursor_up>	Unabhängig von der Cursor-Position wird das Bildschirmfenster um eine Zeile nach oben verschoben (wie Kommando "-1").
<Shift> <Cursor_up>	Markieren des Textes in Richtung Dateianfang für Editier-Funktionen (Kopieren, Löschen, Verschieben, Ausschneiden).
<Cursor_down>	Taste "Cursor nach unten", verschiebt den Cursor um eine Zeile nach unten.
<Strg> <Cursor_down>	Unabhängig von der Cursor-Position wird das Bildschirmfenster um eine Zeile nach unten verschoben (wie Kommando "+1").
<Shift> <Cursor_down>	Markieren des Textes in Richtung Dateiende für Editier-Funktionen (Kopieren, Löschen, Verschieben, Ausschneiden).
<Cursor_left>	Taste "Cursor nach links", verschiebt den Cursor um ein Zeichen nach links.
<Strg> <Cursor_left>	Verschiebt das gesamte Bildschirmfenster um eine Spalte nach links (wie Kommando "<1").
<Shift> <Cursor_left>	Markieren des Textes links vom Cursor für Editier-Funktionen (Kopieren, Löschen, Verschieben, Ausschneiden).
<Cursor_right>	Taste "Cursor nach rechts", verschiebt den Cursor um ein Zeichen nach rechts.
<Strg> <Cursor_right>	Verschiebt das gesamte Bildschirmfenster um eine Spalte nach rechts (wie Kommando ">1").
<Shift> <Cursor_right>	Markieren des Textes rechts vom Cursor für Editier-Funktionen (Kopieren, Löschen, Verschieben, Ausschneiden).
<Einfg>	Taste Einfg oder Ins. Schaltet bei der Daten- und Kommandoeingabe zwischen "Überschreibe-Modus" und "Einfügemodus" um. Der Modus wird in der Statuszeile rechts mit der Kennzeichnung "UEB" bzw. "EINF" angezeigt.
<Ende>	Taste Ende oder End. Verschiebt den Cursor an die Position nach dem letzten Zeichen in der Zeile (Zeile im Datenfenster oder in der Kommandozeile).
<Strg> <Ende>	Verschiebt den angezeigten Ausschnitt an das Ende des Arbeitsbereichs (wie Kommando "++").

<Enter>	Wirkung in der Kommandozeile: Das Kommando wird ausgeführt. Wirkung im Datenfenster mit der Option EDIT WORD OFF: Im Überschreib-Modus werden die Daten geändert. Der Cursor wird in die Kommandozeile positioniert. Wirkung im Datenfenster mit der Option EDIT WORD ON: Der Cursor wird auf die nächste Zeile positioniert. Befindet sich der Cursor am Beginn oder Ende einer Zeile, so wird eine neue Zeile eingefügt. Befindet sich der Cursor innerhalb einer Zeile, so wird die Zeile aufgeteilt (Split). Wirkung in der Markierungsspalte: Die Zeilen-Kommandos werden ausgeführt. Der Cursor wird in die Kommandozeile positioniert.
<Entf>	Taste Entf oder Del. Entfernt das Zeichen an der Cursor-Position. Wirkung im Datenfenster mit der Option EDIT WORD ON: Befindet sich der Cursor am Zeilenende (nach dem letzten Zeichen), so wird das unsichtbare Zeilenende-Kennzeichen gelöscht und die Zeile mit der nachfolgenden Zeile verbunden (Join).
<Alt> <Entf>	Löscht ab der Cursor-Position alle Zeichen bis Zeilenende. Steht der Cursor in der ersten Spalte, so werden alle Zeichen der Zeile gelöscht. Die Zeile bleibt jedoch als Leerzeile erhalten.
<Strg> <Entf>	Löscht die ganze Zeile. Die Zeile bleibt als Leerzeile erhalten.
<Esc>	Taste Esc. Aktion abbrechen, z.B. Hilfe-Ausgabe. Steht eine Eingabe in der Kommandozeile, so wird diese gelöscht. Wirkung im Datenfenster: Positioniert den Cursor in die Kommandozeile.
<Page_up>	Taste PgUp oder "Bild nach oben". Blättert im Arbeitsbereich in Richtung Dateianfang, wie Kommando "-"
<Strg> <Page_up>	Positioniert den Cursor auf das erste Feld am Bildschirm (Zeile 1, Spalte 1).
<Page_down>	Taste PgDn oder "Bild nach unten". Blättert im Arbeitsbereich in Richtung Dateiende, wie Kommando "+".
<Strg> <Page_down>	Positioniert den Cursor auf das letzte Feld am Bildschirm.
<Pos1>	Taste Pos1 oder Home. Verschiebt den Cursor auf die erste Spalte in der aktuellen Zeile. Befindet sich der Cursor am Beginn des Kommandofeldes oder in der Markierungsspalte einer Datenzeile bzw. am Beginn einer Zeile, so verschiebt die Taste Pos1 den Cursor an den Anfang des Bildschirms, d.h. in Zeile 1 und Spalte 1.
<Strg> <Pos1>	Positioniert den Bildschirmausschnitt auf den Anfang des Arbeitsbereichs (wie Kommando "--").
<Tab_right>	Taste "Tabulator rechts". Verschiebt den Cursor in das nächste Eingabefeld.
<Shift_Tab_right>	Taste "Tabulator links" oder "Umschalten Tabulator rechts". Verschiebt den Cursor in das vorhergehende Eingabefeld.
<Strg> <Tab_right>	Aktivieren nächstes EDT-Datenfenster in der Reihenfolge der Arbeitsbereichs-Nummer.

3 Maus-Funktionen

Markieren

Insbesondere die Funktionen im Menü `Bearbeiten` und `Funktionen` sowie im Menü der rechten Maustaste beziehen sich auf markierte Datenbereiche im Arbeitsbereich. Diese Funktionen können natürlich auch mit den Shortcut-Tasten (z.B. `<Strg> C` zum Kopieren usw.) und den Schaltflächen in der Schaltflächenzeile ausgelöst werden.

Zusätzlich können mit allen EDT-Kommandos, die mit Bereichsangaben arbeiten, auch markierte Datenbereiche bearbeitet werden. Als Bereichszeichen ist in diesem Fall das Zeichen "|" anzugeben. Das Zeichen kann mit dem Menü `Optionen / Einstellungen / Sonderzeichen` definiert werden.

Im Editierfenster können Sie in den folgenden Bereichen Daten markieren:

im Datenbereich: ganze Zeilen, Teilzeilen oder Rechteck;
in der Spalte mit der Zeilennummer: ganze Zeilen;
im Zeilenlineal: Ein Spaltenbereich über alle Sätze.

Der markierte Text wird in einer anderen Farbe dargestellt. Die Farbe können Sie im Menü `Optionen/Farben...` einstellen.

Die Markierung bleibt so lange bestehen, bis ein Mausklick auf eine Position außerhalb der Markierung erfolgt oder bis eine anderer Bereich markiert wird (Ausnahme Markierung mit der `<Strg>`-Taste). Wenn ein Mausklick im Kommandofeld erfolgt, wird die Markierung nicht gelöscht, damit die Markierung in Kommandos als Bereich angegeben werden kann. Werden jedoch Daten in der Kommandozeile markiert, so wird die Markierung im Datenbereich gelöscht.

Daten können auch mit den Richtungstasten (`<Cursor_left>`, `<Cursor_right>`, `<Cursor_down>`, `<Cursor_up>`, `<PgDown>` und `<PgUp>`) und gedrückter `<Shift>`-Taste markiert werden.

Linke Maustaste ziehen

Durch Ziehen mit der gedrückten linken Maustaste wird der Text vom Beginn der Aktion bis zum Loslassen der linken Maustaste markiert. Sie können in jede Richtung ziehen.

<Alt> Linke Maustaste ziehen

Rechteck markieren: Diese Aktion ist nur im Datenbereich möglich. Die Cursor-Position zu Beginn der Aktion bestimmt die erste Ecke eines Rechtecks. Durch Ziehen mit der linken Maustaste und vorher gedrückter Taste `<Alt>` in vertikaler und horizontaler Richtung wird ein Rechteck markiert.

Diese Funktion kann auch mit dem Menü `Bearbeiten / Rechteck markieren` eingeschaltet werden.

<Shift> Linker Mausklick

Der Bereich von der bisherigen Cursorposition bis zur Position des aktuellen Mausklicks wird markiert.

<Strg> Linker Mausklick oder Linke Maustaste ziehen

Mehrere Bereiche markieren: Diese Aktion ist nur in der Zeilennummer zulässig. Durch Drücken der Taste <Strg> während des Markierens (Ziehen = mehrere Zeilen, Klick = eine Zeile) erreichen Sie, dass die bisherige Markierung an einer beliebigen Stelle erhalten bleibt. Sie können z.B. die Zeilen 1-10 und danach die Zeilen 100-200 usw. markieren.

Drag & Drop

Durch Klicken mit der linken Maustaste in einen markierten Bereich und Ziehen wird der markierte Bereich wahlweise verschoben oder kopiert. Die Funktion Drag & Drop kann wahlweise ganz ausgeschaltet oder nur nach Rückfrage zugelassen werden (siehe Menü Optionen/verschiedene Optionen (S. 105)).

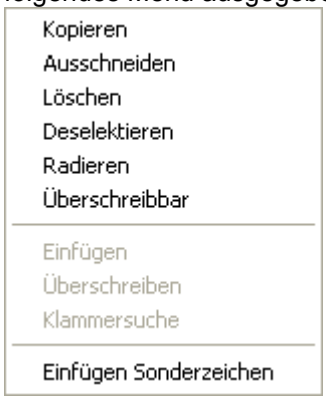
Linke Maustaste ziehen Verschieben: Der markierte Bereich wird im Sendebereich gelöscht und im Zielbereich eingefügt.

Linke Maustaste ziehen + <Strg>

Kopieren: Wenn vor dem Loslassen der linken Maustaste die Taste <Strg> gedrückt wird, werden die markierten Daten im Sendebereich nicht gelöscht.

Kontext-Menü mit der rechten Maustaste

Durch Betätigen der rechten Maustaste wird an der aktuellen Cursorposition folgendes Menü ausgegeben:



In Abhängigkeit des aktuellen Status sind evtl. bestimmte Funktionen deaktiviert, wie z. B. "Einfügen" und "Überschreiben" und "Klammersuche" im obigen Beispiel.

Klammersuche:

Wenn sich der Cursor vor bzw. auf einer öffnenden oder schließenden Klammer (rund "()", eckig "[]" oder geschweift "{}") befindet, wird die dazugehörende schließende bzw. öffnende Klammer gesucht. Die gesamten Daten von der öffnenden bis zur schließenden Klammer werden markiert. Alle in diesem Bereich enthaltenen Klammern der gleichen Art werden mit einer FIND-Markierung versehen. Diese Funktion ist auch über die Tastenkombination <Strg> "5" (S. 47) erreichbar.

Einfügen Sonderzeichen:

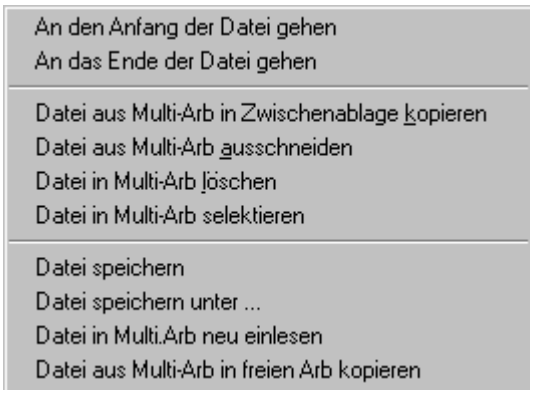
Es wird eine Dialogbox mit allen verfügbaren Zeichen des aktuellen Zeichensatzes ausgegeben (S. 91).

Hinweis:

Werden in einen markierten Bereich Daten aus der Zwischenablage eingefügt, werden vor dem Einfügen die markierten Daten, wie in WINDOWS-Programmen üblich, gelöscht. Ist dies nicht gewünscht, kann über eine Option im Menü `Optionen/verschiedene Optionen` (S. 127) das Löschen der markierten Daten ausgeschaltet werden.

Untermenü bei Multi-Arbeitsbereichen

Falls es sich um einen Multi-Arbeitsbereich handelt (mehrere Dateien in einem Arbeitsbereich, siehe Kommando `READ` (S. 224)), wird ein Folgemenü angezeigt, das Funktionen für die aktuelle Datei anbietet. Als aktuelle Datei gilt die Datei, zu der die Zeile gehört, in der sich der Cursor befindet.



An den Anfang der Datei gehen

Die erste Zeile der aktuellen Datei wird als erste Zeile des Arbeitsbereichs angezeigt.

An das Ende der Datei gehen

Die letzte Zeile der aktuellen Datei wird als erste Zeile des Arbeitsbereichs angezeigt.

Datei aus Multi-Arb in Zwischenablage kopieren

Alle Datenzeilen der aktuellen Datei werden in die Zwischenablage kopiert.

Datei aus Multi-Arb ausschneiden

Alle Datenzeilen der aktuellen Datei werden gelöscht und in die Zwischenablage übertragen.

Datei in Multi-Arb löschen

Alle Datenzeilen der aktuellen Datei werden gelöscht.

Datei in Multi-Arb selektieren

Alle Datenzeilen der aktuellen Datei werden selektiert, d.h. farblich markiert und für weitere Aktionen markiert, wie z.B. Kopieren, Überschreiben usw.

Datei speichern

Alle Datenzeilen der aktuellen Datei werden in die Originaldatei geschrieben. Es erfolgt keine Rückfrage, ob die Datei überschrieben werden soll.

Datei speichern unter ...

Alle Datenzeilen der aktuellen Datei werden in eine Datei geschrieben. Der Name der Datei kann über eine Dialogbox ausgewählt, bzw. eingegeben werden.

Datei in Multi-Arb neu einlesen

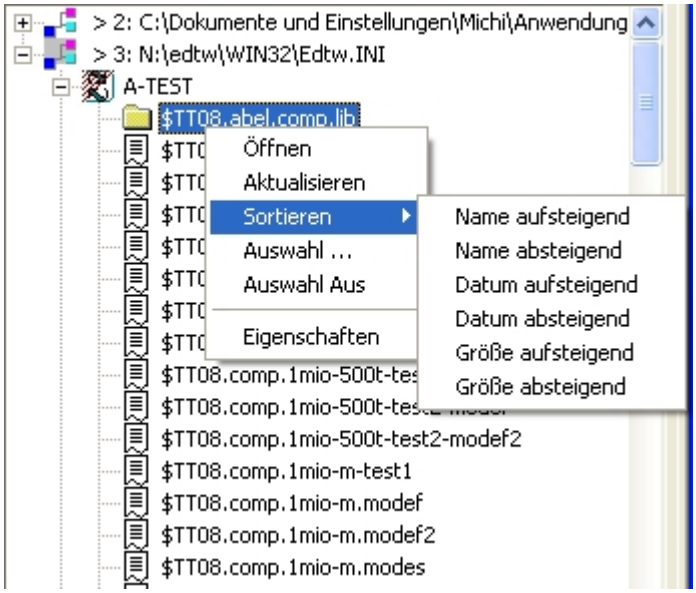
Alle Datenzeilen der aktuellen Datei werden gelöscht. Anschließend wird die Datei an die gleiche Stelle neu eingelesen.

Datei aus Multi-Arb in freien Arb kopieren

Alle Datenzeilen der aktuellen Datei werden in den nächsten freien Arbeitsbereich kopiert. Dieser Arbeitsbereich erhält den Namen der Originaldatei.

Untermenü im Baumfenster

Falls es sich um ein Verzeichnis oder ein Profil im Baumfenster (S. 100) handelt, wird ein Menü mit den Optionen zum Öffnen, Aktualisieren, Dateiauswahl, Sortieren und Anzeigen bzw. Ändern der Eigenschaften angezeigt:



Öffnen

Das markierte Profile bzw. das markierte Verzeichnis wird geöffnet und die Elemente werden angezeigt.

Aktualisieren

Das markierte Profile bzw. das markierte Verzeichnis wird neu eingelesen.

Sortieren

Die Elemente eines Profils bzw. Verzeichnisses werden nach Name, Datum oder Größe aufsteigend oder absteigend sortiert.

Auswahl ....

Es wird eine Dialogbox (S. 55) ausgegeben, in der ein Suchbegriff eingegeben werden kann. Wahlweise können die Elemente gefiltert werden, d.h. es werden nur die Elemente angezeigt, die die Suchbedingungen erfüllen oder es werden weiterhin alle Elemente angezeigt und die Treffer werden farblich markiert.

Auswahl Aus

Es werden alle Elemente angezeigt. Soweit Elemente farblich markiert waren, wird diese Markierung gelöscht.

Eigenschaften

Die Eigenschaften des Profils bzw. des Verzeichnisses werden angezeigt.

Auswahl von Tree-Elementen

Filter

Es werden nur die Elemente angezeigt, deren Dateinamen die Bedingungen des Suchbegriffs enthalten. Das Icon des Profils bzw. des Verzeichnisses wird als Hinweis auf den Filter geändert.

Suchen

Die gefundenen Treffer werden mit einem geänderten Icon angezeigt (grün). Mit der Taste F3 wird auf den nächsten Treffer und mit der Taste F4 auf den vorhergehenden Treffer positioniert.

Suchbegriff

Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*****) miteinander verknüpft sind. Für jedes Suchargument kann eine Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe S. 346.

Die Klein / Großschreibung wird nicht beachtet (intern wird immer `V'...'` verwendet). Falls die Klein / Großschreibung berücksichtigt werden soll, muss `L'...'` angegeben werden.

Doppelklick linke Maustaste

Durch einen Doppelklick mit der linken Maustaste können folgende Funktionen durchgeführt werden:

Zeilennummer Die Zeile wird als überschreibbar gekennzeichnet (wie Markierung X in der Markierungsspalte).

Datenbereich Das Wort, in dem sich der Cursor befindet, wird markiert.

Doppelklick rechte Maustaste

Arbeitsbereich 1 - 31 Durch einen Doppelklick mit der rechten Maustaste auf die Zeilennummer oder den Datenbereich wird diese Zeile als erste Zeile im Arbeitsbereich angezeigt.

Protokollbereich Die Kommandos `ON ...PRINT ... und PRINT` übertragen die entsprechenden Zeilen in den Protokollbereich (Arbeitsbereich 32). Durch einen Doppelklick mit der rechten Maustaste auf die Zeilennummer oder den Datenbereich wird der Ursprungsbereich der angeklickten Zeile aktiviert und auf diese Zeile positioniert.

Intelli Mouse-Funktionen

Mit dem Rad der IntelliMouse können Sie einfach und bequem navigieren. Mit dem Stellrad können Sie den Arbeitsbereich vertikal und horizontal verschieben, ohne dass Sie die Maus auf die Scrollbars positionieren müssen. Folgende Funktionen sind vorgesehen:

Rad-Impuls alleine	Drei Zeilen auf/ab
Rad + Ctrl	Eine Seite auf/ab
Rad + Shift	Drei Spalten links/rechts
Rad + Ctrl + Shift	Ein Achtel der Seitenbreite links/rechts

4 Menüzeile

Über die Menüzeile können alle Funktionen des EDT erreicht werden. Sowohl alle Aktionen, die über die Kommandos in der Kommandozeile möglich sind als auch alle sonstigen Funktionen und Einstellungen, die zusätzlich angeboten werden, können in der Menüzeile ausgewählt werden. Die wichtigsten Funktionen können zusätzlich über die Schaltflächenzeile aktiviert werden.

Menü Datei

Neu Erzeugt eine neue Datei.

Eine bereits existierende Datei wird mit der Option Öffnen in einen neuen Arbeitsbereich eingelesen.

Shortcuts

Symbol:
Tasten: <STRG>+N

Öffnen Eine bereits existierende Datei wird in einen neuen Arbeitsbereich eingelesen. Die Datei kann in einer Dialogbox (S. 60) ausgewählt werden. Eine Datei kann auch mehrmals eingelesen werden. Benutzen Sie das Menü Fenster zum Wechseln zwischen den verschiedenen Fenstern Es können auch mehrere Dateien, max. 32, durch Benutzung der Taste <Strg> bzw. <Shift> markiert werden. In diesem Fall wird jede Datei in einen eigenen Arbeitsbereich eingelesen.

Mehrere Dateien können wie folgt als ausgewählt markiert werden:

- Auswählen von mehreren Dateien, die in der Liste nicht hintereinander stehen: Erste Datei mit einem linken Mausklick markieren + Taste <Strg> + linker Mausklick, um eine zusätzliche Datei zu markieren.
- Auswählen von mehreren Dateien, die in der Liste hintereinander stehen: Erste Datei mit einem linken Mausklick markieren + letzte Datei mit <Shift> + linker Mausklick markieren.

Neue Dateien können mit der Option Neu erzeugt werden.

Shortcuts

Symbol:
Tasten: `<Strg>`+O

Öffnen mehrfach

In der Dialogbox `Öffnen` können beliebig viele Dateien ausgewählt werden, die anschließend in ein Fenster eingelesen werden. Mehrere Dateien können wie folgt als ausgewählt markiert werden:

- Auswählen von mehreren Dateien, die in der Liste nicht hintereinander stehen: Erste Datei mit einem linken Mausklick markieren + Taste <Strg> + linker Mausklick, um eine zusätzliche Datei zu markieren.
- Auswählen von mehreren Dateien, die in der Liste hintereinander stehen: Erste Datei mit einem linken Mausklick markieren + letzte Datei mit <Shift> + linker Mausklick markieren.

Die Gesamtlänge der ausgewählten Dateinamen darf max. 32 KB betragen. Wird diese Grenze erreicht, können Sie entweder den Dialog mehrmals aufrufen und die zusätzlich entstandenen Arbeitsbereiche in den ersten Arbeitsbereich kopieren oder Sie können das Kommando "READ mehrfach" (S. 224) benutzen, beim dem es keine Begrenzung bezüglich der Anzahl der Dateien gibt. Sie können das Kommando READ auch mehrmals in einem Arbeitsbereich aufrufen.

Vor dem Beginn einer neuen Datei wird eine Zeile mit dem Dateinamen, der Größe der Datei und dem Datum der letzten Änderung eingefügt. Diese Zeile beginnt mit dem String "$$$***>>>>>>>>>> " (3 x $, 3 x * und 10 x >). Nach der letzten Zeile einer Datei wird ein Endezeile eingefügt, der mit dem String "<<<<<<<<<<***$$$" beginnt und ebenfalls die Statusangaben der Datei enthält.

Mit dem Menübefehl `Extras / Liste der Dateien im REWRITE-Arbeitsbereich` (S. 136) kann eine Liste aller im REWRITE-Arbeitsbereich enthaltenen Dateien angezeigt und zu ausgewählten Dateien positioniert werden.

Alle im Arbeitsbereich enthaltenen Dateien können gemeinsam geändert werden, z.B. mit dem Kommando ON...CHANGE, und danach mit dem Kommando REWRITE in die Ursprungsdateien oder in neue Einzeldateien in einem anderen Verzeichnis geschrieben werden.

Hinweis:
Die vom EDT erzeugte erste und letzte Zeile, die zur Identifizierung der enthaltenen Dateien dienen, dürfen nicht gelöscht oder geändert werden, da dies bei einer Folgeverarbeitung mit dem Kommando REWRITE zu Fehlern führen kann.

Anhängen

Eine bereits existierende Datei wird an das Ende des aktuellen Arbeitsbereichs angehängt. Die Datei kann in einer Dialogbox (S. **60**) ausgewählt werden. Es können auch mehrere Dateien durch Benutzung der Taste <`Strg`> bzw. <`Shift`> markiert werden. In diesem Fall werden alle Dateien hintereinander wird an das Ende des aktuellen Arbeitsbereichs Arbeitsbereich angehängt bzw. in den leeren Arbeitsbereich eingelesen. Im Gegensatz zur Funktion `Datei/Öffnen mehrfach` (S. **58**) werden zwischen den Einzeldateien keine Trennzeilen eingefügt.

Mehrere Dateien können wie folgt als ausgewählt markiert werden:

a) Auswählen von mehreren Dateien, die in der Liste nicht hintereinander stehen:
b) Erste Datei mit einem linken Mausklick markieren + Taste <`Strg`>
c) mit linker Maustaste zusätzliche Datei markieren.

Auswählen von mehreren Dateien, die in der Liste hintereinander stehen:

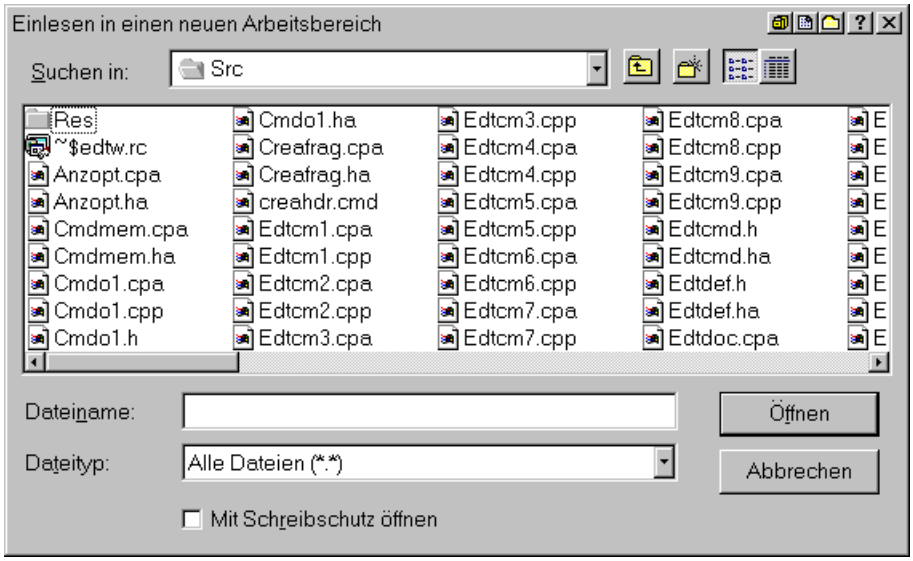
a) Erste Datei mit einem linken Mausklick markieren
b) letzte Datei mit <`Shift`> + linker Mausklick markieren.

Neue Dateien können mit der Option Neu erzeugt werden.

Shortcuts:

Symbol:
Tasten: <`Strg`>+H

Dialogbox für Datei Einlesen und Datei Anhängen



Suchen in

Zeigt eine Liste der zur Verfügung stehenden Ordner und Dateien an. Wo sich der aktuelle Ordner in der Hierarchie auf Ihrem Computer befindet, wird angezeigt, wenn Sie auf den Abwärtspfeil klicken. Der Inhalt eines Ordners wird angezeigt, wenn Sie darauf doppelklicken.
Das Feld unten zeigt die Ordner und Dateien im ausgewählten Pfad an. Sie können auch auf einen Ordner oder eine Datei doppelklicken, um den Ordner oder die Datei zu öffnen.
Sie öffnen einen Ordner, der sich eine Ebene weiter oben in der Hierarchie befindet, indem Sie auf die entsprechende Schaltfläche in der Symbolleiste klicken.

Listbox

Um ein in diesem Feld aufgelistetes Dokument zu öffnen, doppelklicken Sie auf den Dokumentnamen. Um ein Dokument in einem anderen Ordner zu öffnen, doppelklicken Sie auf den Namen des Ordners.

Dateiname

Um das Dokument anzugeben, das Sie öffnen oder suchen möchten, geben Sie im Feld Dateiname den Namen des Dokuments ein.

Dateityp

Gibt den Typ der Datei an, die Sie öffnen. Die Liste enthält alle zur Verfügung stehenden Dateitypen, die das Programm erkennt. Die Liste der Dateitypen kann im Menü `Optionen Filter` (S. 125) modifiziert werden.

Mit Schreibschutz öffnen

Gibt an, dass Sie die Möglichkeit haben, die Datei zu lesen, aber keine Änderungen daran vornehmen können.

Schließen Es werden alle Fenster geschlossen. Falls eine Datei noch nicht gesichert ist, wird eine Dialogbox angezeigt, in der die Sicherung ausgewählt werden kann. Das Schließen eines Fensters ohne Sichern führt zum Verlust aller Daten seit dem letzten Zurückschreiben. Vor dem Schließen einer neuen Datei ohne Namen wird die Dialogbox Sichern als angezeigt, in der Sie den Dateinamen bestimmen können.

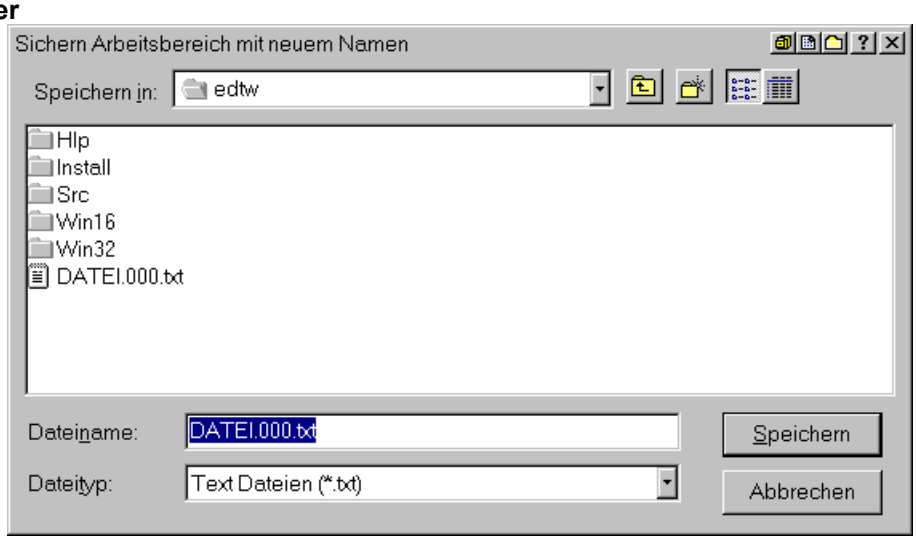
Sie können das Dokument auch durch einen Doppelklick auf das Close-Icon im Dateifenster schließen:

Speichern Der Arbeitsbereich wird unter dem aktuellen Dateinamen und Verzeichnis zurückgeschrieben. Bei der erstmaligen Sicherung einer neuen Datei wird die Dialogbox Speichern unter angezeigt. Falls Sie den Dateinamen oder das Verzeichnis ändern wollen, wählen Sie das Kommando `Speichern unter`.

Shortcuts

Symbol:
Tasten: `<Strg>`+S

Speichern unter



Suchen in

Zeigt eine Liste der zur Verfügung stehenden Ordner und Dateien an. Wo sich der aktuelle Ordner in der Hierarchie auf Ihrem Computer befindet, wird angezeigt, wenn Sie auf den Abwärtspfeil klicken. Der Inhalt eines Ordners wird angezeigt, wenn Sie darauf doppelklicken.
Das Feld unten zeigt die Ordner und Dateien im ausgewählten Pfad an. Sie können auch auf einen Ordner oder eine Datei doppelklicken, um den Ordner oder die Datei zu öffnen.
Sie öffnen einen Ordner, der sich eine Ebene weiter oben in der Hierarchie befindet, indem Sie auf die entsprechende Schaltfläche in der Symbolleiste klicken.

Listbox

Zeigt eine Liste der Ordner und Dateien im ausgewählten Pfad an. Der Inhalt eines Ordners wird angezeigt, wenn Sie darauf doppelklicken.
Sie können auch das Feld Suchen in verwenden, um die Ordnerhierarchie anzuzeigen.
Sie öffnen den Ordner, der sich eine Ebene höher in der Hierarchie befindet, indem Sie auf die entsprechende Schaltfläche in der Symbolleiste klicken.

Dateiname

Stellt Ihnen einen Bereich zur Eingabe des Dateinamens zur Verfügung.
Sie können das Zeichen * als Platzhalterzeichen verwenden. Sie können z.B. *.* eingeben, um eine Liste aller Dateien anzuzeigen.
Sie können auch den vollständigen Pfad einer Datei angeben. Sie können z.B. `C:\dokument\brief.doc` eingeben.

Dateityp

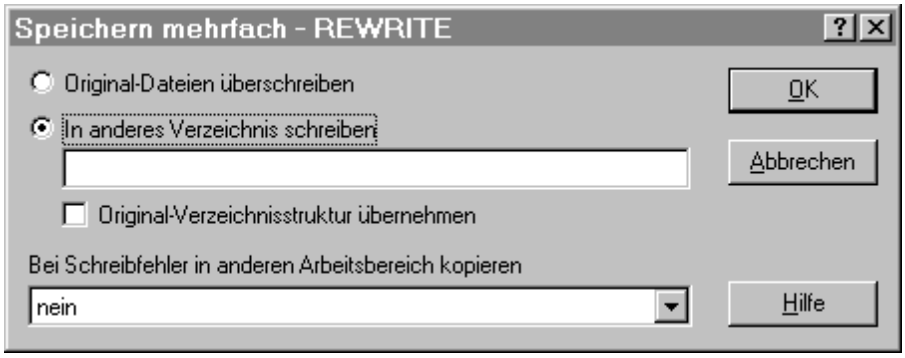
Gibt den Typ der Datei an, die Sie speichern. Die Liste enthält alle zur Verfügung stehenden Dateitypen, die das Programm erkennt. Die Liste der Dateitypen kann im Menü `Optionen Filter` (S. 125) modifiziert werden.

Netzwerk...

Mit diesem Button können Sie eine Verbindung zu einer Netzwerkfreigabe herstellen.

Zum Speichern unter dem ursprünglichen Namen und Verzeichnis können Sie die Option Sichern verwenden.

Speichern mehrfach



Wurden in einen Arbeitsbereich mehrere Dateien mit dem Kommando `READ` *rng* (*n*) bzw. über das Menü-Kommando `Öffnen mehrfach` (siehe auch Kommando `FSTAT` (S. **197**) bzw. `READ'..*..'` (S. **224**) und `REWRITE` (S. **237**)) eingelesen, werden alle Dateien zurückgeschrieben. Nach Ausführung des Kommandos wird ein Protokoll ausgegeben, das für jede Datei den vollständigen alten und evtl. neuen Dateinamen und die Anzahl der Sätze enthält.

Die Originaldateien werden ohne Warnung überschrieben. Es werden nur die Dateien zurückgeschrieben, die tatsächlich geändert wurden. Vor dem Zurückschreiben wird geprüft, ob die Originaldatei noch das gleiche Datum und die gleiche Uhrzeit hat wie beim Einlesen.

in anderes Verzeichnis schreiben

Ist eine Datei bereits vorhanden, erfolgt eine Rückfrage, ob die Datei überschrieben werden soll. Mit dieser Option werden alle Dateien in das angegebene Verzeichnis geschrieben, unabhängig von einer Modifizierung.

Original-Verzeichnisstruktur übernehmen

Der volle Pfad (ohne Laufwerksbezeichnung) wird an die Pfadangabe angehängt, so dass die ursprüngliche Verzeichnisstruktur erhalten bleibt.

Beispiel:

Feld "in anderes Verzeichnis schreiben" enthält `C:\NEU`

Der Arbeitsbereich enthält die Dateien
`C:\TEST1\TEST` und
`C:\TEST2\TEST`

Diese Dateien werden gespeichert unter:
`C:\NEU\TEST1\TEST` und
`C:\NEU\TEST2\TEST`

Bei Schreibfehler in anderen Arbeitsbereich kopieren

Falls beim Schreiben von Dateien Fehler auftreten, werden die Daten der fehlerhaften Dateien einschließlich des Beginn- und Endesatzes in den angegebenen Arbeitsbereich geschrieben, d.h. es wird ein neuer REWRITE-Arbeitsbereich erzeugt. Nach Beheben der Fehler (z.B. Netzlaufwerk nicht erreichbar oder Schreibschutz), kann in diesem Arbeitsbereich eine neue REWRITE-Aktion gestartet werden. Der aktuelle Arbeitsbereich bleibt unverändert.

Alles speichern

Alle geöffneten Dateien werden gespeichert.

Löschen.. Löschen von Dateien. Der Name bzw. die Namen der zu löschenden Dateien werden in einer Dialogbox angefordert. Mehrere Dateien können wie folgt als ausgewählt markiert werden:

- Auswählen von mehreren Dateien, die in der Liste nicht hintereinander stehen: Erste Datei mit einem linken Mausklick markieren +
  Taste <Strg> + linker Mausklick, um eine zusätzliche Datei zu markieren.
- Auswählen von mehreren Dateien, die in der Liste hintereinander stehen: Erste Datei mit einem linken Mausklick markieren +
  letzte Datei mit <Shift> + linker Mausklick markieren.

Falls mehrere Dateien ausgewählt sind, erfolgt eine Rückfrage, ob das Löschen jeder Datei bestätigt werden soll.

Drucken ... Der markierte Bereich, eine Auswahl von Seiten oder der ganze Arbeitsbereich wird ausgedruckt. In einer Dialogbox kann der Drucker ausgewählt und die Anzahl der Exemplare eingegeben werden.

Die Optionen für die Druckaufbereitung können im Menü `Optionen / Einstellungen / Drucken` (S. **114**) eingestellt werden. Die Schrift für den Ausdruck kann im Menü `Optionen / Einstellungen / Schrift Drucken` (S. **115**) ausgewählt werden.

Druckbildvorschau...

Die Daten werden wie beim Drucken aufbereitet und angezeigt. Es können wahlweise eine oder zwei Seiten nebeneinander angezeigt werden.

Die Optionen für die Druckaufbereitung können im Menü `Optionen / Einstellungen / Drucken` (S. **114**) eingestellt werden. Die Schrift für den Ausdruck kann im Menü `Optionen / Einstellungen / Schrift Drucken` (S. **115**) ausgewählt werden.

Druckereinrichtung...

In einer Dialogbox können folgende Optionen geändert werden:

Name: Auswahl des Druckers.
Orientierung: Hochformat oder Querformat.
Größe: Papierformat, z.B. A4.
Quelle: Papierfach, von dem das Papier eingezogen werden soll.
Eigenschaften: Weitere Einstellungen für den Drucker.

1, 2, 3, 4 Klicken Sie mit der Maus auf den Eintrag mit der gewünschten Datei, die Sie einlesen wollen oder geben Sie die entsprechende Zahl ein. In der Liste der zuletzt benutzten Dateien sind auch die entfernten Dateien mit allen notwendigen Parametern für den Filetransfer enthalten. Die Anzahl der max. Einträge kann im Menü `Optionen / Einstellungen / Nächster Start` (S. **127**) eingestellt werden.

Weitere Dateien..

Es wird eine Liste mit weiteren Dateien angezeigt, die bisher mit EDT bearbeitet wurden. Klicken Sie mit der Maus auf den Eintrag mit der gewünschten Datei.

Die Dateien sind in der Reihenfolge der letztmaligen Benutzung sortiert. Mit dem Button `Sortiert` können die Dateien nach dem Namen sortiert bzw. nachher wieder in die ursprüngliche Reihenfolge umsortiert werden.

Ende (Menü Datei)

Ende

EDT für Windows wird beendet. Sie können EDT auch durch Doppelklick auf das Icon im linken oberen Eck des Fensters beenden. Falls Arbeitsbereiche noch nicht gesichert sind, fordert EDT für Windows Sie auf, die Arbeitsbereiche zurückzuschreiben.

Shortcuts
Mouse: Doppelklick auf das Applikation Icon

Tasten: ALT+F4

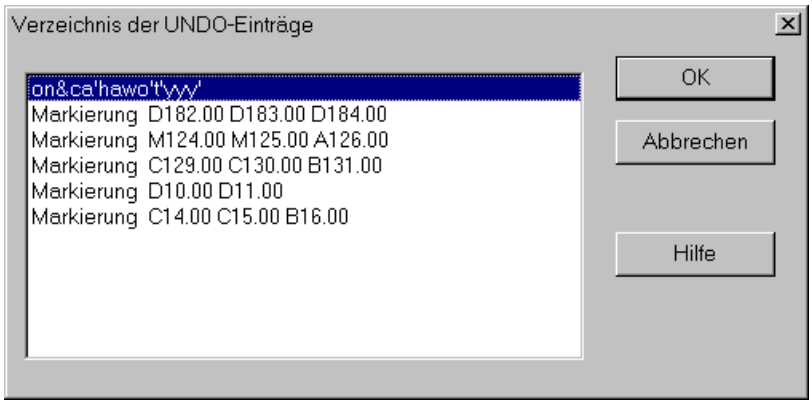
Menü Bearbeiten

Rückgängig Die letzte Aktion wird rückgängig gemacht. Die UNDO-Funktion kann auch mit dem Kommando `UNDO` (S. 254) aufgerufen werden. Mit dem Kommando `UNDO ?` wird eine Dialogbox (S. 66) mit den bisherigen Aktionen ausgegeben.

Shortcuts:

Symbol:
Tasten: `<Strg>`+Z oder
`<Alt> + <Backspace>`

Dialogbox UNDO



In der Listbox werden alle bisher durchgeführten Aktionen angezeigt. Wenn Sie einen Eintrag auswählen und mit dem OK-Button bestätigen, werden alle Aktionen vom ersten Eintrag (letzte ausgeführte Aktion) bis zur ausgewählten Aktion rückgängig gemacht. Wenn Sie z.B. den dritten Eintrag auswählen, werden die Aktionen von Zeile 1 bis Zeile 3 rückgängig gemacht.

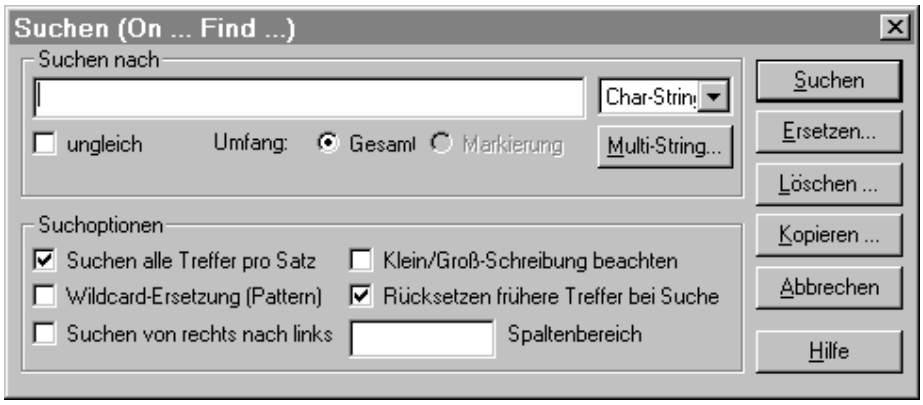
Wiederherstellen

Vorhergehende UNDO-Aktion wieder rückgängig machen.

Shortcuts:

Symbol:
Tasten: <Strg>+Y

Suchen



Die Suche erfolgt in dem markierten Bereich bzw. wenn nichts markiert ist, im gesamten Arbeitsbereich.

Shortcuts:

Symbole: Dialogbox Suchen anzeigen.

Positionieren auf nächsten Treffer.

Positionieren auf vorhergehenden Treffer.

Treffermarkierung deselektieren.

Tasten: <F3> Positionieren auf nächsten Treffer.

<F4> Positionieren auf vorhergehenden Treffer.

<Strg> F Dialogbox Suchen anzeigen.

Suchen

Es werden alle Treffer markiert. Der erste Treffer wird entsprechend der Einstellung im Menü `Optionen / Suchen` (S. 110) als n-ter Satz im Datenfenster angezeigt. Zum Positionieren auf den nächsten bzw. vorhergehenden Treffer können Sie die Symbole  und.  bzw. die Tasten `<F3>` und `<F4>` verwenden. Zum Löschen aller Markierungen können Sie das Symbol  oder das Kommando `DMA` verwenden.

Ersetzen...

Die Dialogbox `Suchen und Ersetzen` (S. 72) wird angezeigt. Hier können die weiteren Parameter zum Ersetzen von Daten angegeben werden.

Löschen ...

Die Dialogbox `Suchen und Löschen` (S. 77) wird angezeigt. Hier können die weiteren Parameter zum Löschen von Zeilen oder Suchbegriffen angegeben werden.

Kopieren ...

Die Dialogbox `Suchen und Kopieren` (S. 75) wird angezeigt. Hier können die weiteren Parameter zum Kopieren der Trefferzeilen angegeben werden.

ungleich

Die Suchbedingung ist erfüllt, wenn der Suchbegriff nicht vorkommt. Beim "`Suchen und Löschen`" ist diese Option nur zulässig, wenn der ganze Satz gelöscht wird.

Diese Option gilt ebenfalls nicht für den Suchtyp "Multi-String" und "Suchdatei", weil bei diesen beiden Varianten die Not-Option innerhalb des Multi-Strings für jede Einzelbedingung angegeben werden kann.

Umfang

Gesamt: Der Suchbegriff wird im gesamten Arbeitsbereich gesucht.

Markierung: Der Suchbegriff wird nur im markierten Bereich gesucht. Diese Option ist nur aktiv, falls eine Markierung vorhanden ist.

Suchen nach

Je nach Suchtyp (siehe rechtsstehendes Auswahlfeld) können folgende Suchbegriffe eingetragen werden:

Char-String Die Suchzeichenfolge ist ohne Hochkommas einzugeben. Evtl. vorkommende Hochkommas oder Anführungszeichen werden als Suchstring interpretiert und sind nicht zu verdoppeln wie beim ON-Kommando.

Hexa-String Hexa-Zeichenfolge ohne das einleitende Zeichen "X" und ohne Hochkommas.

Multi-String Der Suchbegriff besteht aus mehreren Such-Zeichenfolgen, die mit und/oder verknüpft werden. Für jeden Suchbegriff kann ein Vergleichsoperator (< / > / = / !=) und ein Spaltenbereich angegeben werden. Der Suchbegriff kann entweder über eine weitere Dialogbox (S. 71) erzeugt werden, die mit dem Button `Multi-String` erreicht wird oder direkt hier eingegeben werden (ausführliche Beschreibung siehe S. 346). Die Funktionalität entspricht den Kommandos `ON ... SEARCH ...`.

Suchdatei Hier kann eine Datei angegeben werden, in der die Suchbegriffe gespeichert sind. Dateiaufbau siehe S. 348. Die Funktionalität entspricht den Kommandos `ON ... SEARCH` (*filename*) ...

Satzlänge Suchen alle Zeilen mit der angegebenen Länge. Folgende Angaben sind möglich:.

nnn | =*nnn* | >*nnn* | <*nnn* | *nnn-mmm*

nnn Satzlänge bzw. "Satzlänge von".

mmm "Satzlänge bis".

Diese Option ist nur für die Funktion "Suchen" zulässig. Kopieren, Löschen und Ersetzen ist nicht möglich.

Für jeden der vier Suchtypen wird der jeweils letzte Suchbegriff gespeichert und bei der nächsten Auswahl wieder angezeigt.

Auswahlfeld für den Suchtyp

In diesem Feld kann der Typ des Suchbegriffs ausgewählt werden.

Char-String Suchen nach einer Zeichenfolge.

Hexa-String Suchen nach einem Hexadezimalen String.

Multi-String Suchen nach mehreren Zeichenfolgen.

Suchdatei Suchen nach mehreren Zeichenfolgen, die in einer Datei gespeichert sind.

Satzlänge Suchen nach Zeilenlänge.

Leere Sätze Suchen nach leeren Zeilen. Dieser Suchtyp ist nur für die Aktion "`Suchen und Löschen`" zulässig.

Treffer Suchen nach Zeilen, die durch eine vorherige Suche-Aktion als Treffer markiert wurden. Dieser Suchtyp ist nur für die Aktionen "`Suchen und Löschen`" sowie "`Suchen und Kopieren`" zulässig. Dadurch ist es möglich, zuerst mit, evtl. mehreren, Suche-Aktionen Zeilen zu markieren und dann alle markierten Suchen und Kopieren bzw. Suchen und Löschen zu bearbeiten. Diese Option entspricht der Angabe `MARK` des Kommandos `ON`.

Die Dialogbox `Multi-String` (S. 71) wird ausgegeben. Hier können mehrere Suchbegriffe eingegeben werden.

Suchoptionen

Die Suchoptionen "Klein/Groß-Schreibung beachten" und "Rücksetzen frühere Treffer" werden beim ersten Aufruf aus den Einstellungen der Menüzeile `Optionen / Suchen` (S. 110) übernommen. Evtl. Änderungen der Suchoptionen bleiben bis zum nächsten Aufruf der Dialogbox erhalten, werden aber beim Beenden des Programms nicht gespeichert.

Suchen alle Treffer im Satz

☑ Es werden alle gefundenen Such-Zeichenfolgen in einer Zeile als Treffer angezeigt. Diese Option entspricht der Angabe `ALL` im Kommando `ON` (S. **210**).

☐ Es wird nur der erste Treffer in einer Zeile angezeigt.

Wildcard-Ersetzung (Pattern)

☑ Musterzeichen im Suchbegriff ersetzen. Diese Option entspricht der Angabe `PATTERN` im Kommando `ON` (S. **210**). Es gibt zwei Musterzeichen:

* ersetzt eine beliebig lange, auch leere Zeichenfolge;
/ ersetzt genau ein Zeichen.

☐ Die Musterzeichen werden nicht ersetzt.

Suchen von rechts nach links

☑ Die Suche erfolgt pro Zeile von rechts nach links, d.h. von Zeilenende Richtung Zeilenanfang. Diese Option entspricht der Angabe "R" des Kommandos `ON ....`

☐ Die Suche erfolgt von Zeilenanfang in Richtung Zeilenende.

Klein- Großschreibung beachten

☑ Die Groß-/Kleinschreibung wird beachtet.

☐ Es wird unabhängig von der Groß-/Kleinschreibung nach der angegebenen Zeichenfolge gesucht. Der gleiche Effekt kann auch durch Angabe der Suchzeichenfolge in der Form V'*string*' erreicht werden.

Falls die Option `LOW OFF` (Kleinbuchstaben in Großbuchstaben umwandeln) aktiv ist, wird die Zeichenfolge auch ohne diese Option in Großbuchstaben umgewandelt.

Rücksetzen frühere Treffer bei Suchen

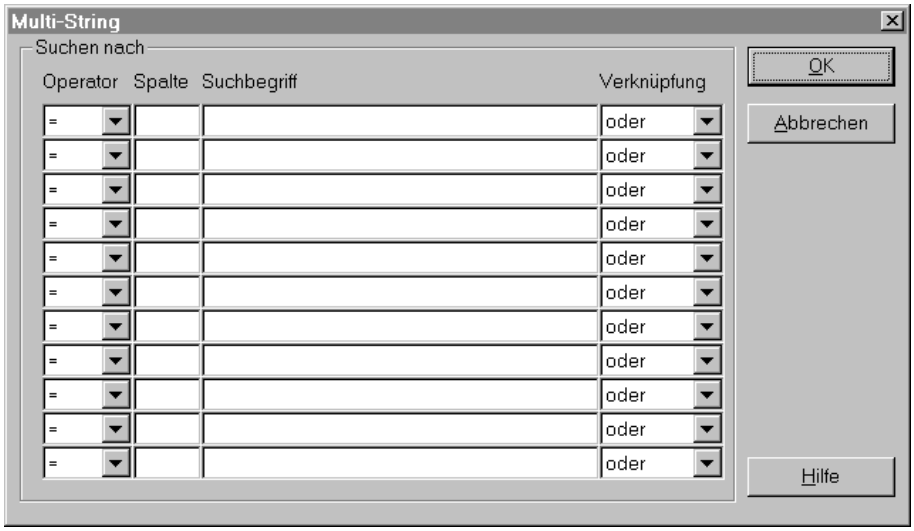
☑ Die Find-Markierungen werden automatisch vor jeder neuen Suche gelöscht. Siehe hierzu auch das Kommando `DMA` (S. **188**).

☐ Die Find-Markierungen bleiben erhalten.

Spaltenbereich

Durch Angabe eines bzw. mehrerer Spalten oder Spaltenbereiche kann die Suche auf bestimmte Spalten eingeschränkt werden. Spaltenbereiche werden in der Form *spalte1 - spalte2* angegeben. Mehrere Spalten werden durch Komma getrennt. Die Einschränkung auf bestimmte Spalten kann auch dadurch erreicht werden, dass vor Aufruf der Dialogbox nur die gewünschten Spalten markiert werden.

Multi-String



Suchen nach

Diese Variante des Suchbegriffs eröffnet die Möglichkeit, mehrere Suchbegriffe mit und/oder zu verknüpfen. Zu jedem Suchbegriff kann ein Spaltenbereich und ein Vergleichsoperand (</>/=/!=) angegeben werden.

Operator

Vergleichsoperator:

>	Suche nach einer Zeichenfolge > Suchbegriff
<	Suche nach einer Zeichenfolge < Suchbegriff
!= oder -	Suche nach einer Zeichenfolge ungleich Suchbegriff.
=	Suche nach einer Zeichenfolge = Suchbegriff.

Spalte

Spaltenbereich, in dem die gesuchte Zeichenfolge beginnen muss.

:col1-col2: Das erste Zeichen der gesuchten Zeichenfolge muss im Spaltenbereich zwischen *col1* und *col2* beginnen.

:col1: Die Zeichenfolge wird nur an der angegebenen Spalte *col1* gesucht und muss dort beginnen.

>:col1: | *<:col1:*
Die Zeichenfolge wird im Bereich ab Spalte *col1* bis Satzende (>:*col1*:) bzw. vom Satzanfang bis Spalte *col1* gesucht (<:*col1*:).

Standard: Suche in gesamten Spaltenbereich (von Spalte 1 bis Zeilenende) bzw. nur in den markierten Spalten.

Suchbegriff

Suchzeichenfolge: *string* | '*string*' | X'*string*'| V'*string*'

Die Suchzeichenfolge kann mit oder ohne Hochkommas eingegeben werden. Es kann auch eine Hexa-Zeichenfolge in der Form X'...' angegeben werden. Ebenfalls zulässig ist die Angabe von Anführungszeichen als spezielle Begrenzungszeichen. Ausführliche Beschreibung des Suchbegriffs siehe S. 236.

V'*string*' Die Zeichenfolge wird unabhängig von der Klein-/ Großschreibung gesucht.

Enthält *string* Hochkommas (') oder Anführungszeichen (''), so müssen diese verdoppelt werden ('' bzw. "").

Verknüpfung

oder Suche das vorausgegangene oder das nachfolgende Suchargument. Die Suchbedingung gilt als erfüllt, wenn zumindest eines der beiden Suchargumente enthalten ist.

und Suche das vorausgegangene und das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente enthalten sind. Die Reihenfolge der Suchargumente in der Zeile ist ohne Bedeutung.

und danach Suche das vorausgegangene und das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente enthalten sind. Die Suchargumente müssen in der gleichen Reihenfolge auftreten, wie im Suche-Kommando angegeben.

Hinweise:

Die Reihe der angegebenen Suchargumente und Verknüpfungsoperatoren wird linear abgearbeitet. Falls mehrere mit "und" bzw. "und danach" verknüpfte Suchargumente angegeben wurden und eines von ihnen nicht in der Zeile enthalten ist, so wird der Suchvorgang beendet bzw. beim nächsten, mit oder "," verknüpften Suchargument fortgesetzt.

Ersetzen

Diese Funktion ermöglicht nicht nur den üblichen Stringaustausch, sondern auch

- das Einfügen eines Strings vor oder nach dem Suchbegriff,
- das Ersetzen der Daten vor oder nach dem Suchbegriff,
- das Kopieren der gesuchten Zeilen in andere Arbeitsbereiche,
- das Löschen der gesuchten Zeilen,
- das Löschen des Suchbegriffs
- das Löschen der Daten vor oder nach dem Suchbegriff.

Dies entspricht der Funktionalität der Kommandos `ON...CHANGE ...`und `ON...INSERT ...` (S. 212). über die Button `Kopieren..` und `Löschen...` können auch die Funktionen der Kommandos `ON...FIND ... COPY...`, `ON ... DELETE` und `ON...FIND DELETE ...` erreicht werden. Die Suche erfolgt in dem markierten Bereich bzw. wenn nichts markiert ist, im gesamten Arbeitsbereich.

Shortcuts:

Tasten: `<Strg> R` Dialogbox Ersetzen anzeigen.

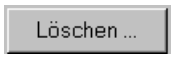
Eine ausführliche Beschreibung der Eingabemöglichkeiten zum Suchbegriff und zu den Suchoptionen finden Sie auf Seite S. 68.

Der erste Treffer wird entsprechend der Einstellung im Menü `Optionen / Suchen` (S. 110) als n-ter Satz im Datenfenster angezeigt. Zusätzlich wird am linken oberen Rand des Bildschirms folgende Dialogbox angezeigt:

Ja	Der aktuelle Suchbegriff wird ersetzt.
Nein	Die Trefferzeile wird nicht geändert Es wird zum nächsten Treffer positioniert.
Alle	Alle Trefferzeilen werden geändert.
Abbrechen	Die Aktion wird abgebrochen.

Alle gefundenen Zeichenfolgen werden ohne Rückfrage ersetzt.

Die Dialogbox `Suchen und Kopieren` (S. 97) wird angezeigt. Hier können die weiteren Parameter zum Kopieren der Trefferzeilen angegeben werden.



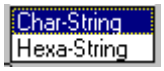
Die Dialogbox `Suchen und Löschen` (S. 100) wird angezeigt. Hier können die weiteren Parameter zum Löschen von Zeilen oder Suchbegriffen angegeben werden.

Ersetzen durch

Je nach String-Typ (siehe rechtsstehendes Auswahlfeld) können folgende Ersetzungs-Zeichenfolgen eingetragen werden:

Char-String	Die Ersetzungs-Zeichenfolge ist ohne Hochkommas einzugeben. Evtl. vorkommende Hochkommas oder Anführungszeichen werden als Ersetzungszeichen interpretiert und sind nicht zu verdoppeln wie beim ON-Kommando.
Hexa-String	Hexa-Zeichenfolge ohne das einleitende Zeichen "X" und ohne Hochkommas.

Auswahlfeld für den String-Typ



In diesem Feld kann der Typ der Ersetzungs-Zeichenfolge ausgewählt werden.

Char-String Suchen nach einer Zeichenfolge.

Hexa-String Suchen nach einem Hexadezimalen String.

Überschreiben des Suchbegriffs

Der Suchbegriff wird durch den Ersetzungsstring ersetzt.

Überschreiben vor dem Suchbegriff

Die Daten vor dem Suchbegriff in der Trefferzeile werden durch den Ersetzungsstring ersetzt.

Überschreiben nach dem Suchbegriff

Die Daten nach dem Suchbegriff in der Trefferzeile werden durch den Ersetzungsstring ersetzt.

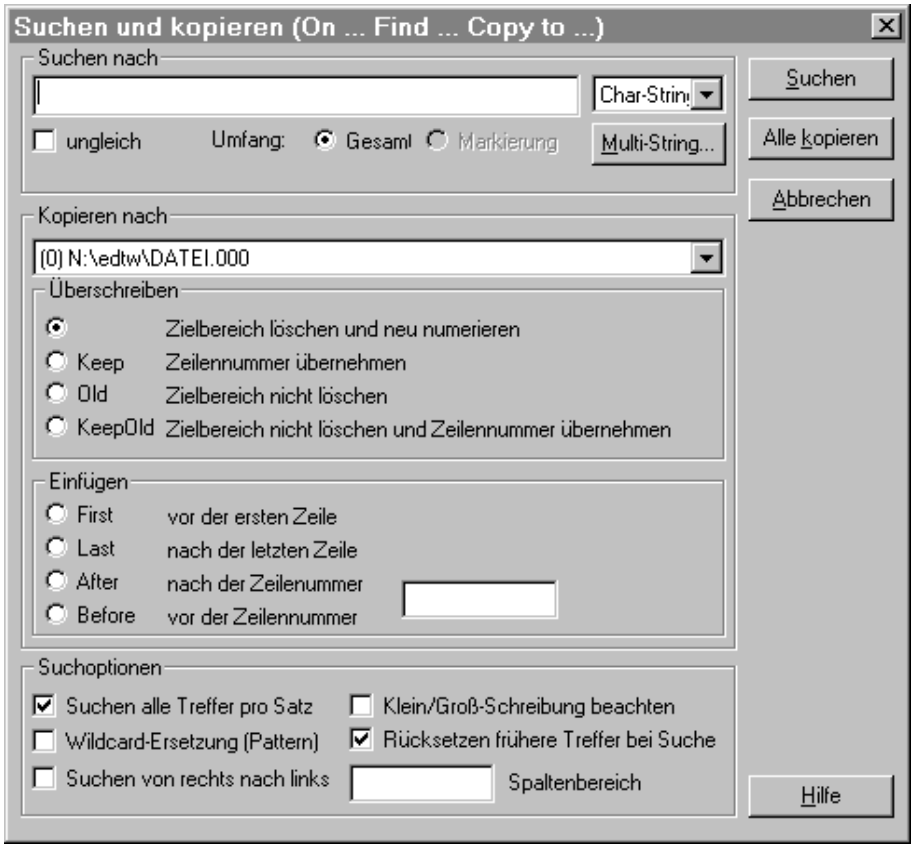
Einfügen vor dem Suchbegriff

Der Ersetzungsstring wird vor dem Suchbegriff eingefügt.

Einfügen nach dem Suchbegriff

Der Ersetzungsstring wird nach dem Suchbegriff eingefügt.

Suchen und Kopieren der Trefferzeilen



Mit dieser Funktion können Zeilen gesucht und in einen anderen Arbeitsbereich kopiert werden. Dies entspricht der Funktionalität des Kommandos `ON...COPY ...` (S. **213**). Die Suche erfolgt in dem markierten Bereich bzw. wenn nichts markiert ist, im gesamten Arbeitsbereich.

Shortcuts:

Tasten: `<Strg> K`

Eine ausführliche Beschreibung der Eingabemöglichkeiten zum Suchbegriff und zu den Suchoptionen finden Sie auf Seite S. **68**.

Der erste Treffer wird entsprechend der Einstellung im Menü `Optionen / Suchen` (S. **110**) als n-ter Satz im Datenfenster angezeigt. Zusätzlich wird am linken oberen Rand des Bildschirms folgende Dialogbox angezeigt:

Ja	Die aktuelle Trefferzeile wird kopiert.
Nein	Die Trefferzeile wird nicht kopiert. Es wird zum nächsten Treffer positioniert.
Alle	Alle Trefferzeilen werden kopiert.
Abbrechen	Die Aktion wird abgebrochen.

Alle gefundenen Zeichenfolgen werden ohne Rückfrage kopiert.

Die Dialogbox `Multi-String` (S. 91) wird ausgegeben. Hier können mehrere Suchbegriffe eingegeben werden.

Kopieren nach

Aus der Listbox kann der Zielarbeitsbereich 0 bis 31 ausgewählt werden. Falls in einem Arbeitsbereich bereits Daten gespeichert sind, wird auch der Dateiname zu der Arbeitsbereichs-Nummer angezeigt.

Zielbereich löschen und neu numerieren

Der Zielarbeitsbereich wird gelöscht. Die kopierten Trefferzeilen werden neu numeriert.

Zielbereich löschen und Zeilennummer übernehmen

Zeilennummer des Sendebereichs bleibt erhalten. Der Zielarbeitsbereich wird vor dem Übertragen gelöscht.

Zielbereich nicht löschen

Zeilennummer des Sendebereichs bleibt erhalten. Der Zielarbeitsbereich wird nicht gelöscht.

Zielbereich nicht löschen und Zeilennummer übernehmen

Zeilennummer des Sendebereichs bleibt erhalten. Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht.

Einfügen vor der ersten Zeile

Die Trefferzeilen werden vor der ersten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

Einfügen nach der letzten Zeile

Die Trefferzeilen werden nach der letzten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

Einfügen nach der Zeilennummer ...

Die Trefferzeilen werden nach der Zeile mit der Zeilennummer n im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

Einfügen vor der Zeilennummer ...

Die Trefferzeilen werden vor der Zeile mit der Zeilennummer n im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

Zeilennummer

Zeilennummer, nach bzw. vor der die Trefferzeilen im Zielarbeitsbereich eingefügt werden sollen (gilt nur für die Optionen "Einfügen vor der Zeilennummer" und "Einfügen nach der Zeilennummer").

Suchen und Löschen

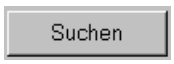
Es können entweder die ganzen Zeilen mit den Treffern oder nur der Suchbegriff bzw. die Daten vor oder nach dem Suchbegriff gelöscht werden.

Mit dieser Funktion können Zeilen oder Teile der Zeilen gelöscht werden. Dies entspricht der Funktionalität der Kommandos `ON...DELETE ... bzw. ON ... FIND DELETE ...` (S. **215**). Die Suche erfolgt im markierten Bereich bzw. wenn nichts markiert ist, im gesamten Arbeitsbereich.

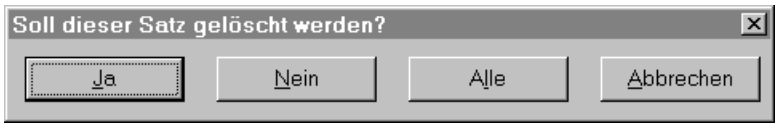
Shortcuts:

Tasten: `<Strg> L` Dialogbox Suchen und Löschen anzeigen.

Eine ausführliche Beschreibung der Eingabemöglichkeiten zum Suchbegriff und zu den Suchoptionen finden Sie auf Seite S. **68**.



Der erste Treffer wird entsprechend der Einstellung im Menü `Optionen / Suchen` (S. **110**) als n-ter Satz im Datenfenster angezeigt. Zusätzlich wird am linken oberen Rand des Bildschirms folgende Dialogbox angezeigt:



Ja	Die Löschaktion für die aktuelle Trefferzeile wird ausgeführt.
Nein	Es wird zum nächsten Treffer positioniert.
Alle	Die Löschaktionen werden für alle Trefferzeilen ausgeführt.
Abbrechen	Die Aktion wird abgebrochen.

Je nach eingestellter Löschoption werden alle gefundenen Zeichenfolgen bzw. Daten vor oder nach dem Suchbegriff oder die gefundenen Zeilen ohne Rückfrage gelöscht.

Die Dialogbox `Multi-String` (S. 91) wird ausgegeben. Hier können mehrere Suchbegriffe eingegeben werden.

Zeile mit Treffer

Löschen aller gefundenen Zeilen. Dieses Kommando entspricht dem Format 13 des ON-Kommandos im BS2000-EDT.

Löschen nur Suchbegriff

Der Suchbegriff wird in allen gefundenen Zeilen gelöscht. Der restliche Zeileninhalt bleibt erhalten. Dieses Kommando entspricht dem Format 10 des ON-Kommandos im BS2000-EDT.

Zeileninhalt vor dem Suchbegriff

Der Inhalt vor dem Suchbegriff wird in den Trefferzeilen gelöscht. Dieses Kommando entspricht dem Format 11 des ON-Kommandos im BS2000-EDT.

Zeileninhalt nach dem Suchbegriff

Der Inhalt nach dem Suchbegriff wird in den Trefferzeilen gelöscht. Dieses Kommando entspricht dem Format 12 des ON-Kommandos im BS2000-EDT.

Gehe zu ... In einer Dialogbox kann die gewünschte Zeilennummer eingegeben werden. Diese Zeile wird dann als erste Zeile im Datenfenster angezeigt. Mit dem Kommando # kann ebenfalls auf eine Zeilennummer positioniert werden.

Kopieren Die markierten Daten überschreiben den bisherigen Inhalt der Zwischenablage.

Shortcuts

Symbol:
Tasten: `<Strg>`+C

Ausschneiden Der markierte Bereich wird gelöscht und in die Zwischenablage kopiert. Diese Option ist nur aktiv, falls Daten markiert sind.

Die ausgeschnittenen Daten überschreiben den bisherigen Inhalt der Zwischenablage.

Shortcuts:

Symbol: 

Tasten: `<Strg>`+X

Löschen Der markierte Bereich wird gelöscht. Diese Option ist nur aktiv, falls Daten markiert sind.

Shortcuts:
Tasten: `<Entf>`

Markierung deselektieren

Deselektieren Die Markierung wird aufgehoben.

Shortcuts:
Tasten: `<Esc>`

Radieren Der markierte Bereich wird mit Leerstellen überschrieben. Die Funktion steht auch über das Menü "Rechte Maustaste" zur Verfügung.

Shortcuts:
Tasten: `<Strg> + <Leerstelle>`

Alles Markieren

Alle Daten des Arbeitsbereichs werden markiert.

Shortcuts:
Tasten: `<Strg>` + A

Rechteck markieren

Beim Markieren mit der Maus über mehrere Zeilen wird ein Rechteck markiert. Diese Einstellung gilt solange, bis die Option wieder deaktiviert wird. Dieser Markier-Modus kann auch temporär (für eine Markierung) eingeschaltet werden, indem vor dem Ziehen mit der Maus die Taste `<Alt>` gedrückt wird.

Treffer deselektieren

Die Treffermarkierung nach dem Kommando ON& .... bzw. der Suche über die Dialogbox des Menüs `Bearbeiten Suchen` wird aufgehoben (wie Kommando DMA).

Shortcuts

Symbol: 

Überschreibbar

Die markierten Zeilen werden zum Überschreiben freigegeben (wie Markierung X). Diese Funktion ist auch über das Menü der rechten Maustaste oder durch Doppelklick erreichbar.

Einfügen

Der Inhalt der Zwischenablage wird an der Einfügemarke eingefügt. Markierte Daten werden überschrieben. Diese Option ist nur verfügbar, falls in der Zwischenablage Daten enthalten sind.

Werden in einen markierten Bereich Daten aus der Zwischenablage eingefügt, werden vor dem Einfügen die markierten Daten, wie in WINDOWS-Programmen üblich, gelöscht. Ist dies nicht gewünscht, kann über eine Option im Menü `Optionen / verschiedene Optionen` (S. 127) das Löschen der markierten Daten ausgeschaltet werden.

Shortcuts

Symbol: 

Tasten: `<Strg>`+V

Überschreiben

Überschreibt die Daten ab der Einfügemarke mit dem Inhalt der Zwischenablage. Falls ein ganzer Satz bzw. ganze Sätze durch Mausklick in der Markierungsspalte markiert wurden und die Sätze in der Zwischenablage kürzer sind, wird der Rest des alten Satzes gelöscht.

Shortcuts

Tasten: `<Strg>`+U

Menü Funktionen

Sortieren

Die Funktionalität entspricht dem Kommando `SORT` (S. **248**)

Umfang

Hier kann entweder der gesamte Arbeitsbereich oder nur der markierte Zeilenbereich angegeben werden. Falls keine Markierung vorhanden ist, ist nur die Option "Gesamt" zulässig.

Sortierweise

aufsteigende Sortierreihenfolge **oder**

absteigende Sortierreihenfolge **oder**

Umdrehen in der Reihenfolge: Alle Zeilen im Arbeitsbereich werden in umgekehrter Reihenfolge sortiert, d.h. die letzte Zeile ist die erste Zeile, die vorletzte Zeile ist die 2. Zeile usw.

Standard: aufsteigend

Bei aufsteigender oder absteigender Sortierreihenfolge kann die Option "Klein-Großschreibung beachten" aktiviert oder deaktiviert werden.

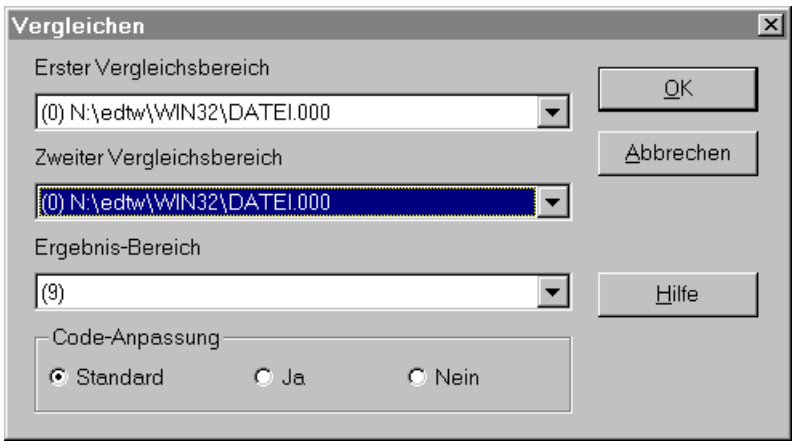
Hinweis:

Falls Sie Zeilen mit dem gleichen Inhalt bzw. Zeilen, die nur in bestimmten Spalten den gleichen Inhalt enthalten, löschen wollen, können Sie das Kommando `DELETE ... MULTIPLE` verwenden.

Spaltenbereich

cl1-cl2 Als Sortierkriterium wird der Inhalt der Spalten *cl1-cl2* benutzt.

Vergleichen



Die Funktionalität entspricht dem Kommando COMP (S. 184)

Arbeitsbereiche

Aus den Listboxen können die zu vergleichenden Arbeitsbereiche und der Ergebnis-Arbeitsbereich ausgewählt werden. Als Ergebnis-Arbeitsbereich ist der Arbeitsbereich 9 vorgegeben.

Code-Anpassung

Ja — Falls die zu vergleichenden Arbeitsbereiche unterschiedlich codiert sind (ANSI/ASCII/EBCDIC/UNICODE), werden die Daten des zweiten Arbeitsbereichs vor dem Vergleich in die Code-Variante des ersten Arbeitsbereichs übersetzt. Die Übersetzung erfolgt nur intern, die Daten des zweiten Arbeitsbereichs bleiben unverändert. Die Daten des Ergebnisbereichs werden in ANSI bzw. UNICODE codiert.

Nein — Falls die zu vergleichenden Arbeitsbereiche unterschiedlich codiert sind (ANSI/ASCII/EBCDIC), werden die Daten ohne Code-Angleichung verglichen. In den Ergebnis-Bereich werden die Daten ebenfalls in der Original-Codierung übertragen.

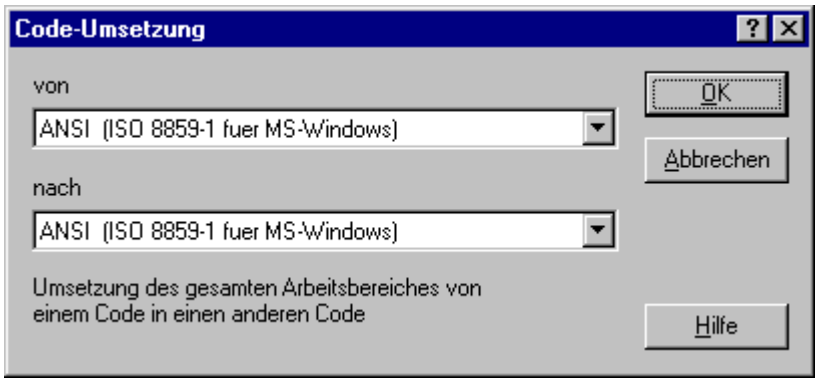
Standard — Textdateien: Vor dem Vergleich erfolgt eine Code-Anpassung (Option Ja)
Binärdateien: Es erfolgt keine Code-Anpassung (Option Nein).

Der Translate-Modus kann auch mit dem Kommando PAR TRANS=YES|NO (S. 218) eingestellt werden.

Klein-/Großbuchstaben

Wählen Sie die Konvertierungsart und den Bereich "Gesamt" oder "Markierung" aus. Es werden auch alle Umlaute und sprachabhängige Sonderzeichen, wie z. B. â, á à usw. umgesetzt. Die Umsetzung erfolgt in Abhängigkeit von den Definitionen in der Datei CODEPAGE.TXT (siehe Kapitel 17 Code-Tabellen (S. 359)). Die Funktionalität entspricht den Kommandos LOWER (S. 208) und UPPER (S. 258).

Code-Umsetzung



Wählen Sie aus der ersten Liste die vorliegende Codierung und aus der zweiten Liste die gewünschte neue Codierung. Die Code-Konvertierung und die Einstellung des Editier-Codes kann auch mit dem Kommando `CODE` (S. **181**) erfolgen. Weitere Informationen siehe Kapitel 17 Code-Tabellen (S. **359**) und 18 UNICODE-Unterstützung (S. **375**).

Hinweise:

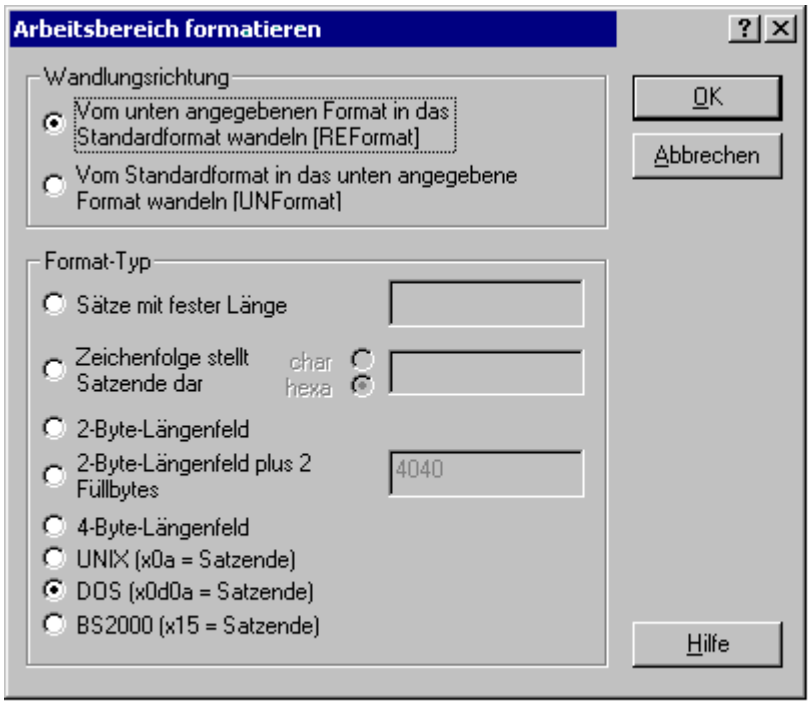
Code für den aktuellen Arbeitsbereich einstellen

Über das Menü `Ansicht / Anzeige-Code` (S. **94**) kann eine andere Code-Variante zum Editieren und Anzeigen des aktuellen Arbeitsbereichs ausgewählt werden kann. Es ist zu beachten, dass dabei nicht die bestehenden Daten des aktuellen Arbeitsbereichs konvertiert werden, sondern nur der richtige Editier-Code ausgewählt wird, d.h. wenn es sich z.B. um eine EBCDIC-Datei handelt, wählen Sie den Editier-Code EBCDIC aus, damit die abdruckbaren Zeichen richtig dargestellt werden und Datenänderungen im EBCDIC-Code durchgeführt werden.

Standardcode für neue Arbeitsbereiche ändern

Über das Menü `Optionen / Code/Shell` (S. **117**) kann die Code-Variante für neue Arbeitsbereiche bestimmt werden.

Formatieren



EDT verarbeitet standardmäßig satzstrukturierte ANSI-Dateien, wobei jede Zeile mit CR/LF abgeschlossen ist. Enthält eine Datei keine Zeilenende-Kennzeichen, so kann sie trotzdem bearbeitet werden. EDT liest dann die Datei in Stücken zu je 64 Bytes pro Zeile ein. Solche Zeilen erkennt man daran, dass anstelle des Punktes in der Zeilennummer ein Stern angezeigt wird. Die Datei ist dann im sog. Binär-Modus eingelesen. Dieser Modus kann auch für eine satzstrukturierte Datei beim Kommando READ durch die Option `B` nach dem Dateinamen erzwungen werden.

Satzstrukturierte Dateien aus anderen Systemen, wie z.B. UNIX oder BS2000, haben einen anderen Aufbau. Um auch solche Dateien im Record-Modus bearbeiten zu können, müssen sie neu formatiert werden. Die Daten können dann wie eine ASCII- oder ANSI-Datei satzweise bearbeitet werden. Die Daten können vor dem Zurückschreiben in die Datei entweder wieder in das ursprüngliche Format umgewandelt werden oder im ASCII- bzw. ANSI-Format gespeichert werden.

Die Funktionalität entspricht den Kommando `REFORMAT` (S. **234**) und `UNFORMAT` (S. **255**).

Wandlungsrichtung

Die Wandlungsrichtung bestimmt die Art der Konvertierung (Fremdes Format --> ANSI oder ANSI --> fremdes Format). Liegen die Daten in einem fremden Format vor, sind die Daten in das ANSI-Format umzuwandeln. Sollen die Daten wieder in das fremde Format bzw. erstmals in das fremde Format konvertiert werden, ist die andere Richtung auszuwählen.

Sätze fester Länge

Die Daten des gesamten Arbeitsbereichs werden in Sätze mit der Länge n aufgeteilt. Der letzte Satz kann kürzer sein, falls die Länge des gesamten Arbeitsbereichs kein Vielfaches der angegebenen Satzlänge ist.

Datei mit anderem Satz-Trennzeichen

Die Zeichenfolge kann als Character-String oder als Hexadezimal-String angegeben werden. Der Arbeitsbereich wird nach der Zeichenfolge durchsucht und in Sätze aufgeteilt bzw. beim Erstellen des Formats wird nach jeder Zeile das Satz-Trennzeichen eingefügt. Falls ein Satz länger ist, als die maximale Satzlänge (32.767), wird er in 64 Byte langen Teilsätzen dargestellt (zu erkennen an dem Stern in der Zeilennummer).

Sätze mit Längenfeld

2-Byte-Längenfeld

Die Datei ist so organisiert, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Ein Satz mit einem Längenfeld von X'0005' enthält z.B. noch drei Nutzbytes. Sätze dieses Formats werden von verschiedenen Textsystemen verwendet (DCA-Format). Enthält das Satzlängenfeld einen Wert < 2, wird die Konvertierung abgebrochen und der gesamte Arbeitsbereich wieder auf den Binärmodus zurückgesetzt.

2-Byte-Längenfeld plus 2 Füllbytes

Die Datei ist so organisiert, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Danach folgen 2 Füllbytes (Standard = x'4040'). Sollten die Füllbytes nicht den Wert X'4040' enthalten, kann der abweichende Wert im nebenstehenden Feld eingetragen werden. Enthält das Satzlängenfeld einen Wert < 4, wird die Konvertierung abgebrochen und der gesamte Arbeitsbereich wieder auf den Binärmodus zurückgesetzt.

4-Byte-Längenfeld

Length-Field - Länge 4. Die Datei ist so organisiert, dass in den ersten vier Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Dieses Format wird z.B. in BS2000-ZIP-Archiven benutzt, die nicht im kompatiblen Format erzeugt wurden.

Die Datei ist so organisiert, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Ein Satz mit einem Längenfeld von X'0005' enthält z.B. noch drei Nutzbytes. Sätze dieses Formats werden von verschiedenen Textsystemen verwendet (DCA-Format). Enthält das Satzlängenfeld einen Wert kleiner 2, wird die Konvertierung abgebrochen und der gesamte Arbeitsbereich wieder auf den Binärmodus zurückgesetzt. Enthält die Satzlänge einen Wert größer als die maximale Satzlänge (32.767), wird dieser Satz weiter im Binärmodus angezeigt. Alle anderen Sätze werden konvertiert.

UNIX

Der aktuelle Arbeitsbereich enthält eine Datei im UNIX-Format mit dem Satz-Trennzeichen X'0A' bzw. wird in dieses Format konvertiert.

DOS

Der aktuelle Arbeitsbereich enthält eine Datei im MS-DOS-Format mit den Satz-Trennzeichen X'0D0A' bzw. wird in dieses Format konvertiert.

BS2000

Der aktuelle Arbeitsbereich enthält eine Datei im POSIX-Format des BS2000 mit den Satz-Trennzeichen X'15' bzw. wird in dieses Format konvertiert.

Prefix/Suffix

Button Prefix

In allen Zeilen des markierten bzw. des gesamten Bereichs wird am Zeilenanfang die Zeichenfolge eingefügt.

Button Suffix

In allen Zeilen des markierten bzw. des gesamten Bereichs wird am Zeilenende die Zeichenfolge eingefügt.

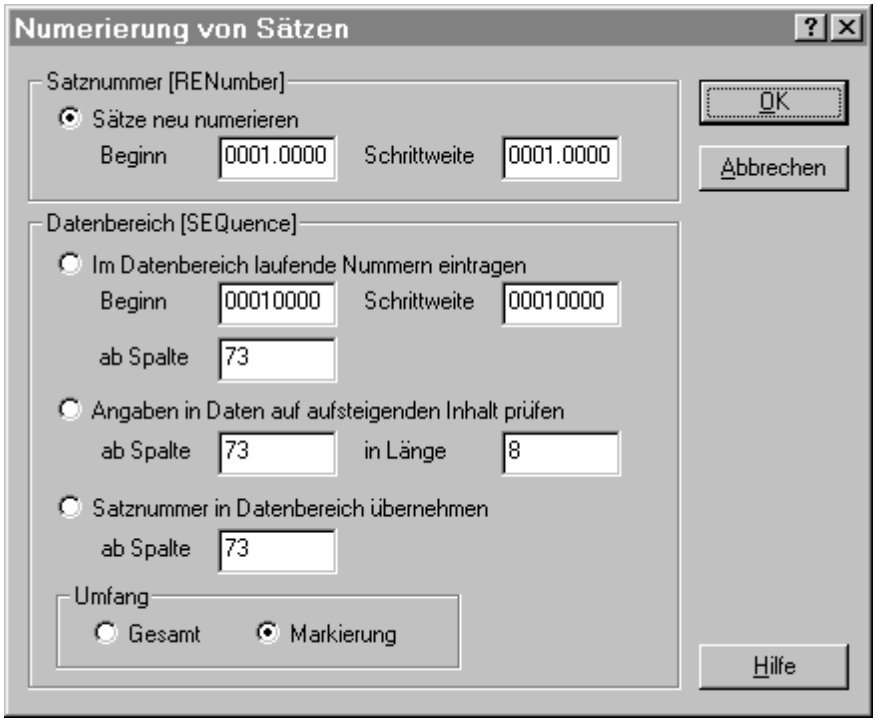
Umfang

Die Aktion wird für alle Zeilen oder für die markierten Zeilen des aktuellen Arbeitsbereichs ausgeführt.

Zeichenfolge

Zeichenfolge, die am Satzanfang bzw. am Satzende eingefügt werden soll. Die Zeichenfolge kann wahlweise im Character- oder Hexadezimalformat eingegeben werden.

Numerieren



Mit dieser Funktion können Sie entweder die Zeilen eines Arbeitsbereichs neu numerieren (wie Kommando RENUMBER siehe S. **236**) oder die Zeilennummer bzw. eine aufsteigende Nummer in die Daten schreiben (wie Kommando SEQUENCE siehe S. **245**).

Satznummer

Die Zeilen im aktuellen Arbeitsbereich werden neu numeriert. Im Feld Beginn ist die Zeilennummer der ersten Zeile und im Feld Schrittweite der Erhöhungswert einzutragen.

Wird bei der Neunumerierung der Maximalwert von 9999.9999 überschritten, gehen keine Zeilen verloren! Lediglich die Numerierung kann nicht mehr korrekt angezeigt werden.

In Datenbereich laufende Nummer eintragen

In jede Zeile des Zeilenbereichs wird an einer bestimmten Spalte eine Zahl geschrieben. Diese Zahlen bilden eine aufsteigende Folge.

ab Spalte

Spalte, in der die erste Ziffer stehen soll (Standard 73).

Beginn

Ganze Dezimalzahl für die erste Zeile (Standard = 00000100), maximal 8 Stellen. Die Zahlen, die in die folgenden Zeilen geschrieben werden, haben die gleiche Stellenanzahl. Fehlt der Wert, schreibt der EDT in die erste Zeile die Zahl 00000100.

Schrittweite

Schrittweite zur Bildung der folgenden Zeilennummern
Standard = 100

Angaben in Daten auf aufsteigenden Inhalt prüfen

Der Inhalt des Spaltenbereichs wird auf aufsteigende Reihenfolge geprüft. Alle Zeilen, in denen der Spalteninhalt gleich oder kleiner der vorhergehenden Zeile ist, werden im Protokollbereich ausgegeben. Alle falschen Zeilen werden mit der FIND-Markierung versehen, d.h. neben der farblichen Hervorhebung kann mit der Taste F3 bzw. mit der Schaltfläche  auf den nächsten falschen Satz positioniert werden.

Im Feld "`ab Spalte`" ist die Spalte anzugeben, in der das erste zu überprüfende Zeichen steht (Standard 73). Im Feld "`in Länge`" ist die Länge des Numerierungs-Strings anzugeben.

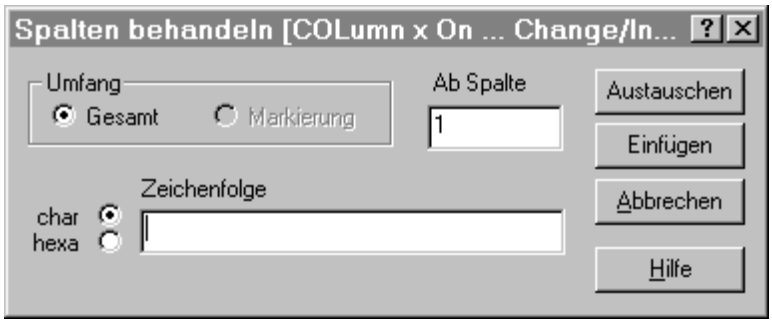
Satznummer in Datenbereich übernehmen

In jede Zeile wird die dazugehörige Zeilen-Nummer geschrieben. Die Zeilennummer wird als 8-stellige Zahl ohne Dezimalpunkt eingefügt. Als Spalte ist die Spalte anzugeben, in der die erste Ziffer stehen soll (Standard 73)

Umfang

Die Aktion wird für alle Zeilen oder für die markierten Zeilen des aktuellen Arbeitsbereichs ausgeführt.

Spalten einfügen/austauschen



Mit dieser Funktion können Sie Zeichenfolgen ab einer bestimmten Spalte einfügen oder austauschen (wie Kommando `COLUMN,` siehe S. 183).

Umfang

Die Aktion wird für alle Zeilen oder für die markierten Zeilen des aktuellen Arbeitsbereichs ausgeführt.

Ab Spalte

Spalte, ab der die Zeichenfolge ausgetauscht oder eingefügt wird.

Zeichenfolge

Die Zeichenfolge kann im Character- oder im Hexadezimalformat eingegeben werden.

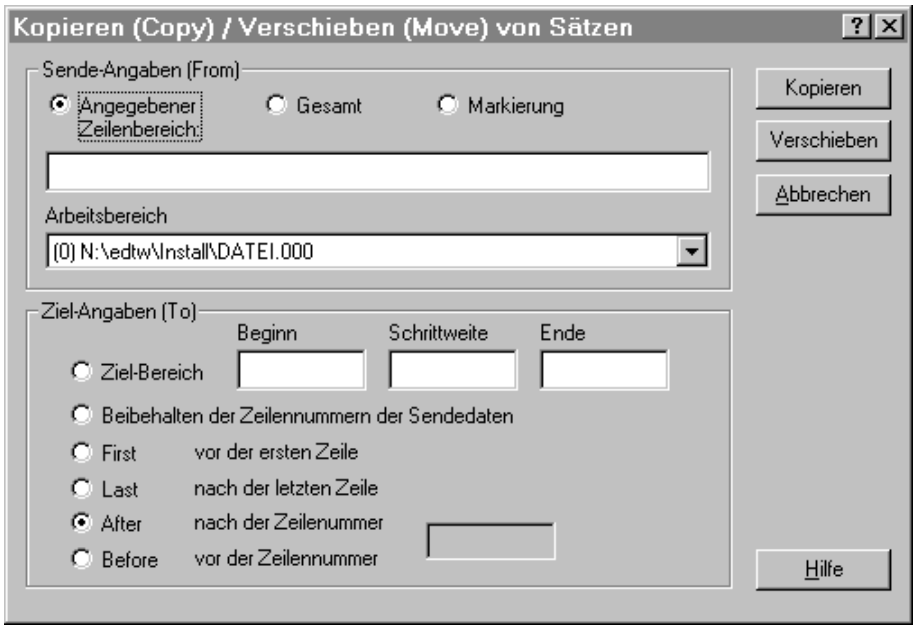
Button Einfügen

Die Zeichenfolge wird ab der angegebenen Spalte eingefügt. Falls die Zeile weniger Spalten enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht.

Button Austauschen

Die Daten werden ab der angegebenen Spalte durch die Zeichenfolge ersetzt. Falls die Zeile weniger Spalten enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht.

Kopieren/Verschieben



Mit dieser Funktion können Sie Daten aus einem anderen oder dem aktuellen Arbeitsbereich in den aktuellen Arbeitsbereich kopieren bzw. verschieben (wie Kommandos `COPY` (S. **185**) und `MOVE` (S. **209**)). Bitte achten Sie darauf, dass Sie sich vor Aufruf dieser Funktion im Ziel-Arbeitsbereich befinden.

Gesamt / Markierung

Die Aktion wird für alle Zeilen oder für die markierten Zeilen des aktuellen Arbeitsbereichs ausgeführt.

Zeilenbereich

Eine oder mehrere Zeilen bzw. Zeilenbereiche, durch Kommas getrennt, z.B. 1,5-8,20-50,100.

Arbeitsbereich

Wählen Sie aus der Listbox den Arbeitsbereich aus, in dem sich die Daten befinden, die kopiert bzw. verschoben werden sollen.

Zielbereich

Die Daten werden ab der angegebenen Zeile (Feld Beginn), in der angegebenen Schrittweite bis zur angegebenen letzten Zeilennummer kopiert bzw. verschoben. Sollen mehr Zeilen kopiert bzw. verschoben werden als laut Zielangabe möglich, werden die restlichen Zeilen nicht mehr verarbeitet. Eventuell vorhandene Daten mit der gleichen Zeilennummer werden überschrieben.

Beibehalten der Zeilennummer des Sendebereichs

Die Zeilennummern des Sendebereichs bleiben im Zielarbeitsbereich erhalten. Eventuell vorhandene Daten mit der gleichen Zeilennummer werden überschrieben.

First

Kopiert bzw. verschiebt die Daten vor die erste Zeile. Es werden keine Zeilen des Zielbereichs überschrieben.

Last

Kopiert bzw. verschiebt die Daten hinter die letzte Zeile. Es werden keine Zeilen des Zielbereichs überschrieben.

After / Before

Kopiert bzw. verschiebt die Daten vor bzw. hinter die Zeile mit der angegebenen Nummer. Es werden keine Zeilen des Zielbereichs überschrieben.

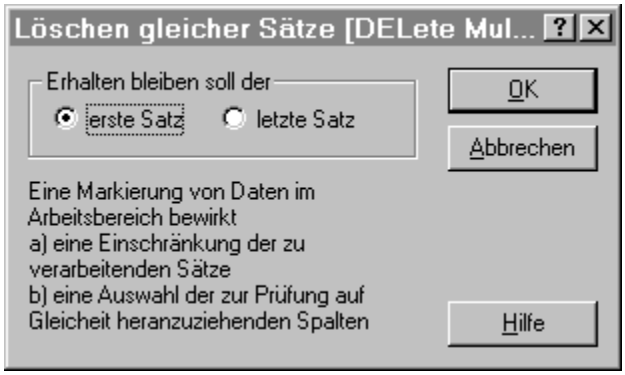
Button Kopieren

Die Daten werden kopiert.

Button Verschieben

Die Daten werden kopiert, anschließend werden die Ursprungsdaten gelöscht.

Löschen gleiche Sätze



In einem sortierten Arbeitsbereich werden alle Zeilen gleichen Inhalts bis auf die erste bzw. letzte Zeile gelöscht. Der Vergleich kann auf bestimmte Spalten eingeschränkt werden. (S. 187)

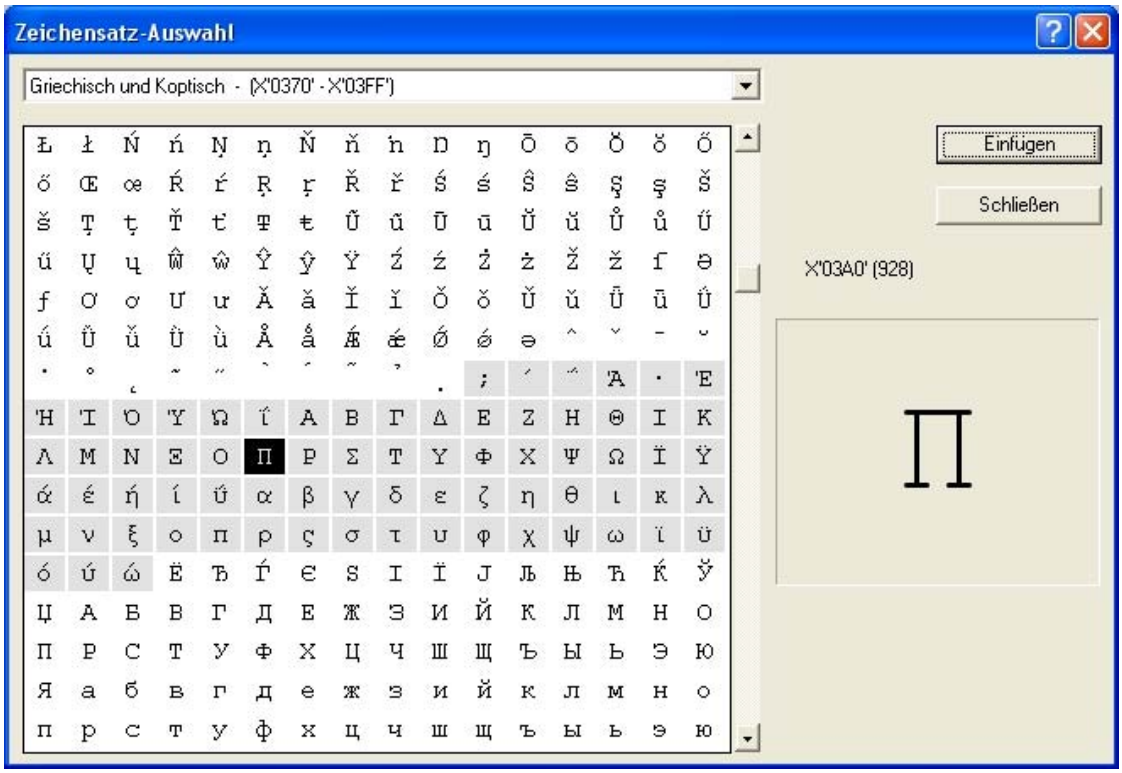
Erhalten bleiben soll der erste Satz

Im angegebenen Bereich werden alle gleichen Zeilen bis auf die letzte Zeile gelöscht (z.B. Zeilen 3 bis 5 sind gleich: es werden die Zeilen 3 und 4 gelöscht). Ist ein Spaltenbereich markiert, werden nur die angegebenen Spalten verglichen.

Erhalten bleiben soll der letzte Satz

Im angegebenen Bereich werden alle gleichen Zeilen bis auf die erste Zeile gelöscht (z.B. Zeilen 3 bis 5 sind gleich: es werden die Zeilen 4 und 5 gelöscht). Ist ein Spaltenbereich markiert, werden nur die angegebenen Spalten verglichen

Sonderzeichen einfügen



Beim Klick auf den Button "Einfügen" wird das ausgewählte Sonderzeichen an der aktuellen Cursorposition im Datenfenster oder in der Kommandozeile eingefügt und der Cursor um ein Zeichen weiter positioniert.

Es werden nur die gültigen Zeichen des aktuellen Zeichensatzes angezeigt. Über die Combobox kann eine Gruppe von Zeichen aus dem UNICODE-Zeichenvorrat ausgewählt werden. Die Zeichen der aktuellen Gruppe werden mit grauem Hintergrund dargestellt.

Menü Ansicht

Hexadezimal Die Daten werden hexadezimal angezeigt. Auch bei eingeschaltetem Long-Modus wird nur eine Zeile pro Datensatz angezeigt. Das Unter-Menü erlaubt die Möglichkeit, eine zwei- oder vierzeilige hexadezimale Ansicht zu aktivieren, wobei letztere nur bei einem Arbeitsbereich im Code UNICODE möglich ist.

Änderungen können sowohl in der Character-Zeile als auch in den Hexa-Zeilen durchgeführt werden. Es wird jeder Tastendruck sofort verarbeitet. Diese Option entspricht dem Kommando `HEXA ON`.

Shortcuts:

Symbol: 

Zeilenumbruch

Zeilen, die länger als die Bildschirmbreite sind, werden in der nächsten Bildschirmzeile fortgesetzt. Diese Darstellungsform wirkt nur bei ausgeschalteter Hexa-Darstellung. Diese Option entspricht dem Kommando `EDIT LONG.`

Shortcuts:

Symbol: 

Zeilenlineal Das Zeilenlineal am oberen Fensterrand wird ein- bzw. ausgeblendet. Diese Option entspricht dem Kommando `SCALE ON`.

Shortcuts:

Symbol: 

Kleinbuchstaben anzeigen

Diese Option gilt für den Fall, dass die Option `Klein- / Großschreibung` ausgeschaltet ist. d.h. Kleinbuchstaben werden bei der Eingabe in Großbuchstaben umgewandelt. Im BS2000-EDT werden in diesem Modus (Kommando LOW OFF) die Kleinbuchstaben als Schmierzeichen dargestellt.

Falls diese Option aktiviert ist, werden im `LOW OFF`-Modus zwar neu erfaßte Kleinbuchstaben in Großbuchstaben umgesetzt, jedoch werden bereits vorhandene Kleinbuchstaben als solche angezeigt. Der Modus kann auch mit dem Kommando `LOWER OFF,DISP=ON` oder `SL` eingeschaltet werden.

Falls diese Option nicht aktiviert ist, werden im `LOW OFF`-Modus die bestehenden Kleinbuchstaben, wie im BS2000 als Schmierzeichen angezeigt. Der Modus kann auch mit dem Kommando `LOWER OFF,DISP=OFF` oder `SLO` eingeschaltet werden.

Shortcuts:

Symbol: 

Statuszeile Die Statuszeile (S. **172**) unterhalb der Kommandozeile wird ein- bzw. ausgeblendet. Die Statuszeile enthält Hinweistexte zu den aktiven Symbolen sowie Informationen über die Cursorposition, die aktuelle Zeilennummer und den Einfügemodus.

Haupt-Toolbar

Die Schaltflächenzeile mit den Icons für die wichtigsten Aktionen, wie z.B. Einlesen, Sichern usw., wird eingeblendet bzw. ausgeblendet.

Beschreibung der Schaltflächen siehe S. 163

Ansicht-Toolbar

Die Schaltflächenzeile mit den Icons für die wichtigsten Ansicht-Optionen, wie z.B. Hexadezimale Anzeige, Zeilenumbruch usw., wird eingeblendet bzw. ausgeblendet.

Beschreibung der Schaltflächen siehe S. 163.

User-Toolbar

Die Schaltflächenzeile mit den Icons für die selbst definierten Verarbeitungen (z.B. Aufruf einer EDTW Input-Prozedur für die aktuelle Arbeitsebene oder Aufruf eines externen Programms) wird eingeblendet bzw. ausgeblendet.

Die Erstellung eigener Schaltflächen kann über Extras -> User-Toolbar vorgenommen werden (S. 157).

Scrollbar senkrecht

Senkrechten Scrollbar am rechten Fensterrand ein- bzw. ausblenden.

Scrollbar waagrecht

Waagrechten Scrollbar am unteren Fensterrand ein- bzw. ausblenden.

Code-Anzeige

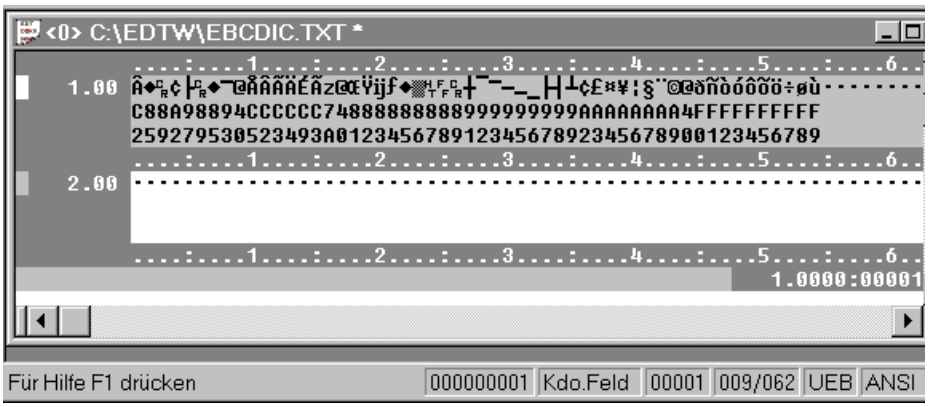
Code-Darstellung für den aktuellen Arbeitsbereich

Falls im Arbeitsbereich vom Standard-Code abweichende Daten stehen, kann hier der entsprechende Editier-Code ausgewählt werden. Dies bewirkt, dass die Darstellung der abdruckbaren Zeichen umgestellt wird und dass alle Datenänderungen im ausgewählten Code durchgeführt werden. Im Hexa-Modus (Kommando HEX ON) wird z.B. im EBCDIC-Code das Zeichen "A" mit "C1" wie im BS2000 dargestellt. Die aktuelle Code-Einstellung wird in der Statuszeile angezeigt.

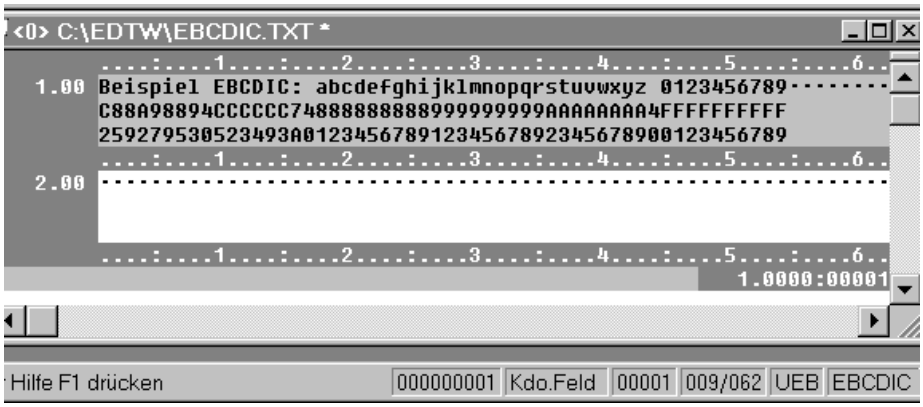
Eine Änderung des Editier-Codes bewirkt keine Umcodierung der bereits bestehenden Daten. Die bestehenden Daten können mit dem Kommando `CODE` (S. 181) bzw. über das Menü `Funktionen / Code-Umsetzung` (S. 83) konvertiert werden.

Beispiel:

Im Arbeitsbereich 0 befindet sich eine EBCDIC-Datei. Es ist die Codierung ANSI eingeschaltet. Die Characterzeile ist nicht lesbar. Zeichen, die über die Tastatur eingegeben werden oder Daten, die durch Kommandos geändert werden, werden in ANSI erstellt.



Im Arbeitsbereich 0 befindet sich eine EBCDIC-Datei. Es ist die Codierung EBCDIC eingeschaltet. Die Characterzeile ist wie im BS2000 lesbar. Zeichen, die über die Tastatur eingegeben werden oder Daten, die durch Kommandos geändert werden, werden in EBCDIC erstellt.



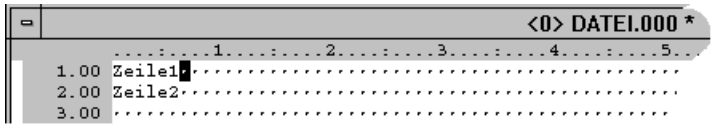
Zeilennummer

Im Datenfenster kann die Zeilennummer (S. **172**) ausgeblendet oder 6-stellig bzw. 8-stellig angezeigt werden.

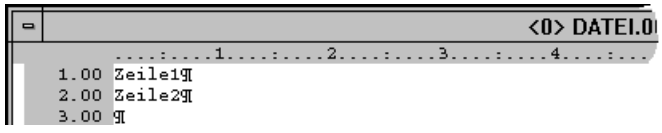
Satzende-Darstellung

Aus Das Satzende wird durch Leerstellen angezeigt.

Nilzeichen Das Satzende wird durch NIL-Zeichen angezeigt.



Endezeichen Das Satzende wird durch ein Satz-Trennzeichen angezeigt.



Cursordarstellung

Die Cursorform kann für den Einfüge-Modus und den Überschreibe-Modus unterschiedlich eingestellt werden.

Einf-Voll	TEST	Cursor Voll mit Invers-Darstellung im Einfüge-Modus
Einf-Senkrecht	TEST	Cursor senkrecht im Einfüge-Modus
Einf-Waagrecht	TEST	Cursor waagrecht im Einfüge-Modus
Ueb-Voll	TEST	Cursor Voll mit Invers-Darstellung im Überschreibe-Modus
Ueb-Senkrecht	TEST	Cursor senkrecht im Überschreibe-Modus
Ueb-Waagrecht	TEST	Cursor waagrecht im Überschreibe-Modus

Syntax-Highlighting ein/ausschalten

Aktiv Syntax-Highlighting mit der zuletzt eingestellten Sprache des aktuellen Arbeitsbereichs einschalten.

keine Sprache ausgewählt Sprache zurücksetzen.

Syntax-Highlighting für eine bestimmte Sprache einschalten:

COB Cobol für BS2000 Mainframes

ASS Assembler /390 für BS2000 Mainframes

SDF SDF-Prozeduren für BS2000 Mainframes

CPP C/C++ für BS2000 Mainframes

Das Syntax-Highlighting kann auch mit der Schaltfläche in der Ansicht-Toolbar (S. **164**) oder mit dem Kommando `SHL` (S. **247**) aktiviert oder de

Siehe auch die Beschreibung der Optionen zum Syntax-Highlighting (S. **149**).

Aktivieren der zusätzlichen Fensterelemente

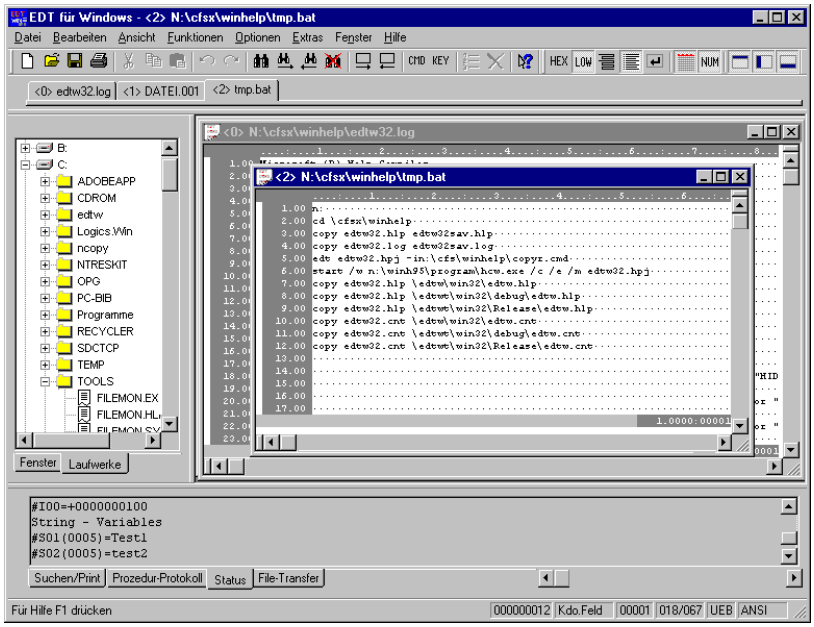
Das EDT-Hauptfenster (S. **97**) kann in 4 verschiedene Bereiche aufgeteilt werden:

Oberer Teil: Karteikarten/Reiter (S. **98**) der Arbeitsbereiche
Linker Teil: Baumfenster (S. **98**) mit einer Liste der Laufwerke/Dateien, der FT-Profile oder der Arbeitsbereiche.
Rechter Teil: Datenfenster mit EDT-Arbeitsbereichen
Unterer Teil: Protokollfenster (S. **100**) für PRINT / STATUS / Fehlermeldungen.

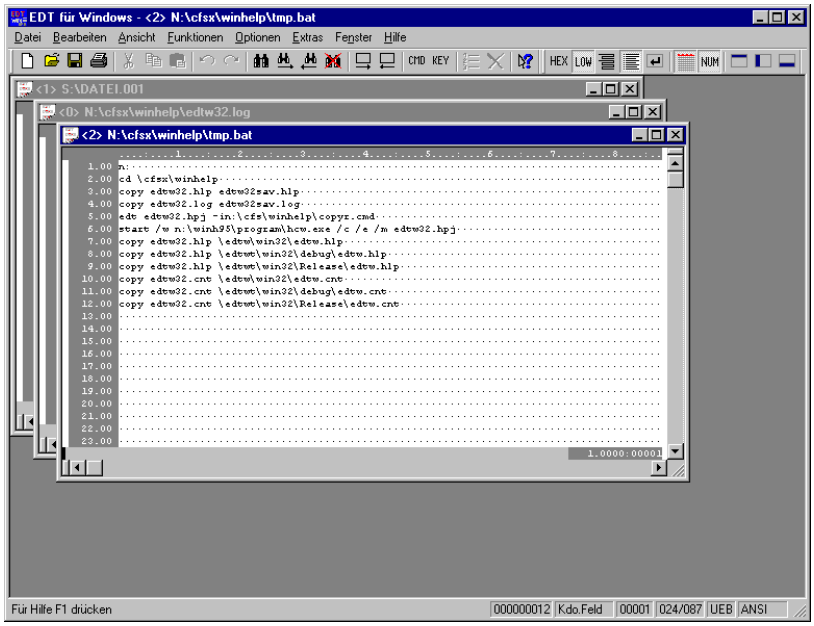
Die Größe aller Elemente kann durch Ziehen mit der Maus am Rand beliebig angepaßt werden.

Die einzelnen Teilfenster können auch mit den folgenden Schaltflächen aktiviert bzw. deaktiviert werden:

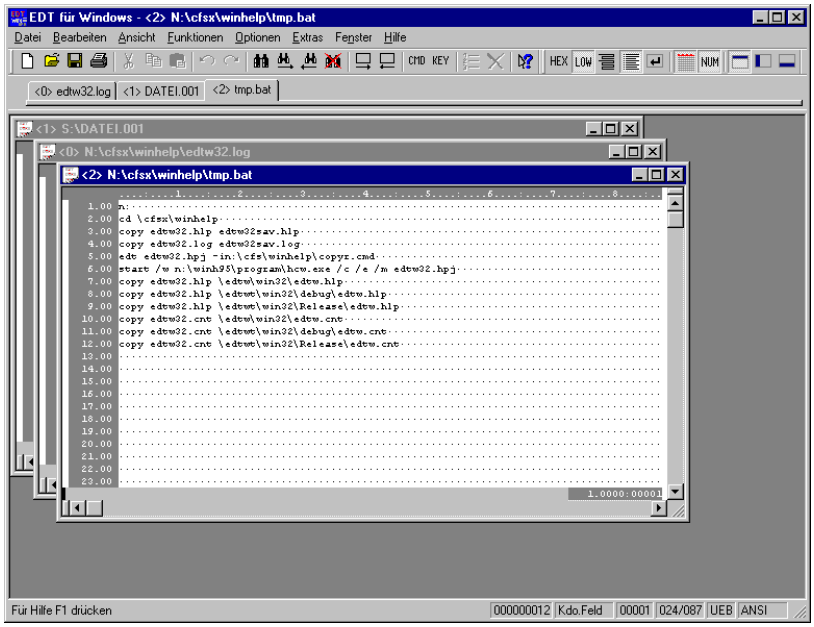
Alle Elemente



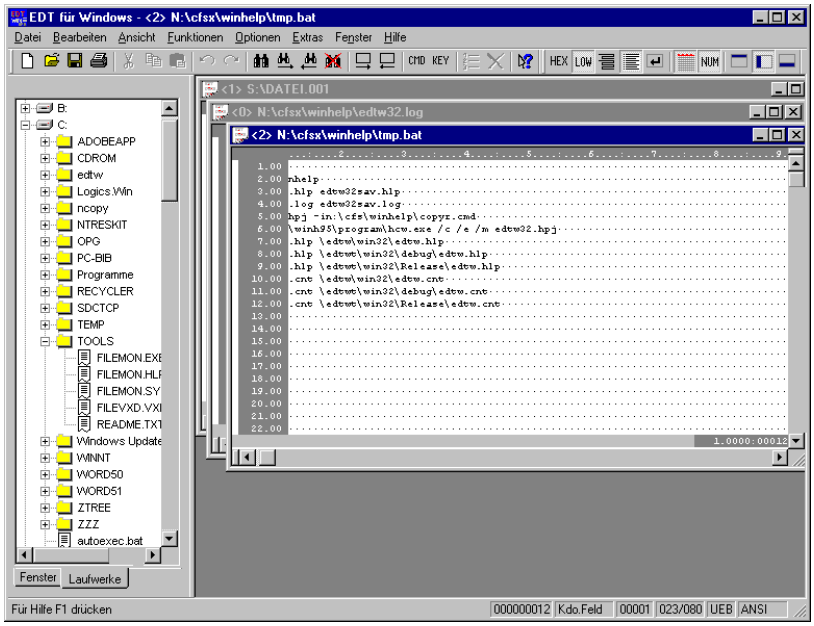
nur Datenfenster



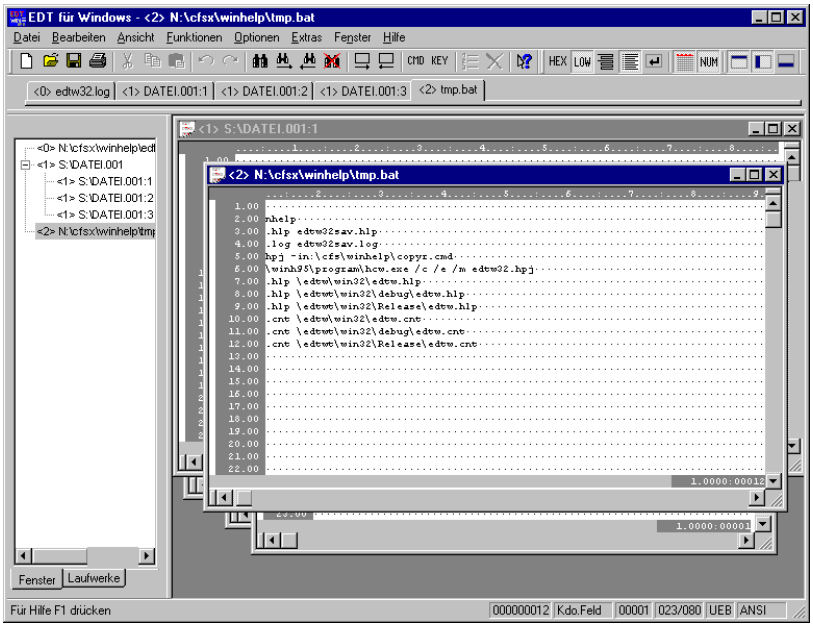
Datenfenster mit Karteikarten



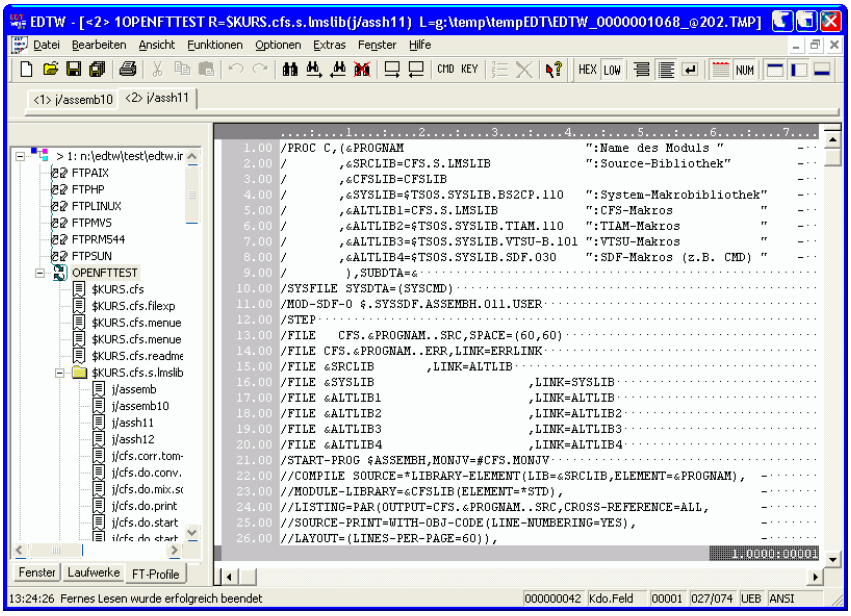
Datenfenster und Baumfenster mit Dateienliste



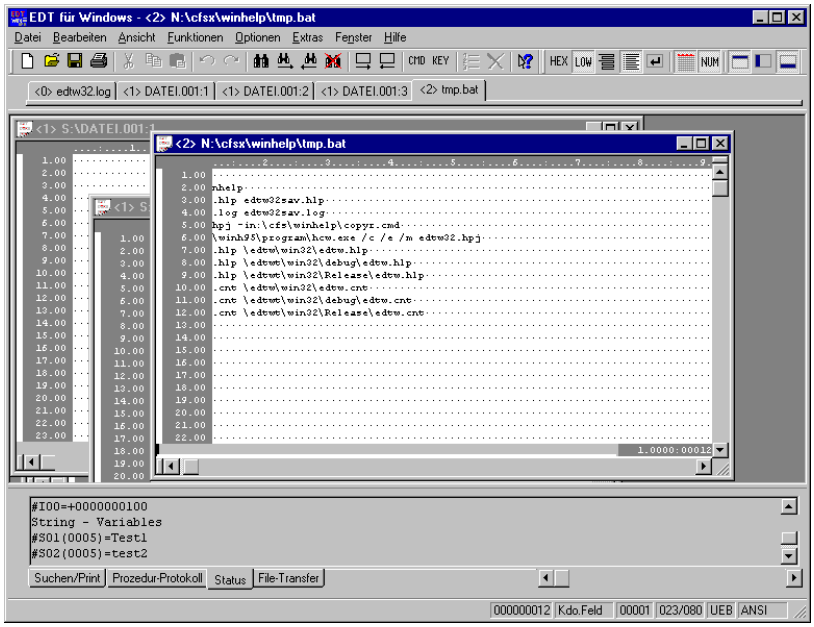
Datenfenster und Baumfenster mit Liste der Arbeitsbereiche



Datenfenster und Baumfenster mit Liste der Filetransfer-Profile



Datenfenster und Protokollfenster



Reiter offene Dateien

Über dem Fenster für die Arbeitsbereiche werden "Reiter" (S. **98**) angezeigt, mit deren Hilfe durch Anklicken der jeweilige Arbeitsbereich aktiviert werden kann.

Weitere Optionen für die Darstellung des Hauptfensters siehe S. **96**.

Shortcuts:

Symbol: 

Baumfenster Links neben dem Fenster für die Arbeitsbereiche wird ein Fenster (S. **98**) angezeigt, in dem entweder eine Liste aller Dateien (wie Explorer), eine Liste mit den geöffneten Arbeitsbereichen oder eine Liste der Filetransfer-Profile (S. **140**) und fernen Dateien in Baumstruktur dargestellt wird. Mit den Schaltflächen "Fenster", "FT-Profile" und "Laufwerke" am untersten Rand des Baumfensters kann zwischen den drei Ansichten umgeschaltet werden.

Shortcuts:

Symbol: 

Liste der lokalen Laufwerke / Dateien:

Die lokalen Laufwerke, Verzeichnisse und Dateien werden wie im Explorer in einer Baumstruktur dargestellt. Durch Klick auf das Symbol "+" wird ein Laufwerk / Verzeichnis geöffnet und die darin enthaltenen Verzeichnisse / Dateien angezeigt. Durch Doppelklick auf eine Datei werden die Daten in den nächsten freien Arbeitsbereich eingelesen. Ein ZIP-Archiv wird dabei wie ein Verzeichnis behandelt.

Die Liste der lokalen Laufwerke wird beim Laden des EDTW aktualisiert. Durch einen Doppelklick auf den Button "Laufwerke" kann auch noch später die Liste aktualisiert werden.

Liste der FT-Profile:

Durch einen Doppelklick auf den Profilnamen wird das Kommando FSTAT mit den Einstellungen des Profils ausgeführt und die Dateien bzw. Verzeichnisse in die Baumstruktur eingehängt.

Folgende Objekte werden dabei als Verzeichnis dargestellt:

- PLAM-Bibliotheken bei einem openFT-Profil
- PLAM-Elemente mit mehreren Versionen bei einem openFT-Profil, wobei die aktuelle Version als Datei angezeigt wird und die restlichen Versionen als Verzeichnis mit Zusatz *old, z.B. `s/test/*old`
- PO-Archive bei einem FTP_MVS-Profil
- Verzeichnisse bei einem FTP_Unix, FTP_POSIX oder FTP_OMVS-Profil

Bei einem Doppelklick auf ein Verzeichnis wird ein Folge-FSTAT aufgerufen und die entsprechenden Objekte in die Baumstruktur eingehängt.

Durch Doppelklick auf eine Datei bzw. ein PLAM-Element oder ein Element eines PO-Archivs werden die Daten in den nächsten freien Arbeitsbereich eingelesen.

Bei einem Klick mit der rechten Maustaste wird ein Menü mit den Punkten "Öffnen" und "Eigenschaften" ausgegeben.

Als Eigenschaften wird bei den Profilen die Dialogbox für die Änderung von Profilen angezeigt. Bei Dateien und Verzeichnissen werden alle verfügbaren Informationen des fernen Systems angezeigt.

Liste der Arbeitsbereiche:

Durch einfachen Mausklick wird der Arbeitsbereich bzw. das entsprechende Viewfenster als aktives Fenster angezeigt.

Weitere Optionen für die Darstellung des Hauptfensters siehe S. **96**.

Ausgabefenster

Im unteren Bereich des Hauptfensters wird ein Fenster (S. **100**) dargestellt, in dem alle Protokollausgaben angezeigt werden. Im Ausgabefenster werden vier Kategorien von Ausgaben unterschieden:

Suchen/Print Alle Ausgaben der Kommandos `ON...PRINT` und `PRINT`
Prozedur-Protokoll: Alle Meldungen aus Prozedurabläufen.
Status: Ausgabe des Kommandos `STATUS`
Filetransfer: Alle Ausgaben des Filetransfers (Kommandos `FSTAT`, `READ` und `WRITE`)

Es wird automatisch die Kategorie angezeigt, in der zuletzt Meldungen ausgegeben wurden. Zusätzlich kann mit den Schaltflächen am unteren Rand des Ausgabefensters die entsprechende Kategorie ausgewählt werden. Weitere Optionen für die Darstellung des Hauptfensters siehe S. **96**.

Shortcuts:

Symbol: 

Menü Optionen

Vollbildmodus

Die Markierungsspalte und das Datenfenster des aktuellen Datenbereichs werden permanent überschreibbar/nicht überschreibbar. Der EDIT FULL-Modus wird automatisch mit der Option WORD Modus im Menü Optionen eingeschaltet. Diese Option entspricht dem Kommando EDIT FULL (S. 189).

Shortcuts:

Symbol: 

Word-Modus In diesem Modus reagiert der EDT auf Tasteneingaben ähnlich wie die andere Editier- und Textverarbeitungsprogramme (z.B. WINWORD). Im Einzelnen bewirkt das Kommando folgendes:

- Falls sich der Cursor im Datenfenster befindet, bewirkt die Taste <Enter>, dass eine neue Zeile begonnen wird. Befindet sich der Cursor am Beginn oder am Ende einer Zeile, so bewirkt dies das Einfügen einer neuen Zeile. Befindet sich der Cursor innerhalb der Zeile, so bewirkt dies das Aufteilen der Zeile an dieser Stelle (Split). Eine Ausführung eines Kommandos erfolgt nur, wenn sich der Cursor bei <Enter> in der Kommandozeile befindet.
- Mit der Taste <Esc> gelangt man vom Datenfenster zur Kommandozeile.
- Befindet sich der Cursor am Ende einer Zeile, so wird mit der Taste <Entf> die aktuelle Zeile mit der nächsten Zeile verkettet.
- Befindet sich der Cursor in der ersten Spalte, so wird mit der Taste <Back> die aktuelle Zeile mit der vorhergehenden verkettet.
- Der EDIT FULL-Modus wird eingeschaltet, d.h. das Editierfenster ist immer überschreibbar (Markierung X oder Taste F2 ist nicht notwendig).

Ist der WORD-Modus ausgeschaltet, verhält sich der EDT wie im BS2000:

- Die Taste <Enter> bewirkt im Datenfenster, dass die eingegebenen Daten in den Speicher übernommen werden. Der Cursor wird in die Kommandozeile positioniert. In die nächste Zeile des Datenfensters gelangt man nur mit der Taste <Tab_Right> oder <Cursor_down>.
- Mit der Taste <Esc> wird das Programm beendet.
- Der EDIT FULL-Modus wird ausgeschaltet, fall die Option Vollbildmodus im Menü Optionen nicht aktiv ist. Die Option entspricht dem Kommando EDIT WORD.

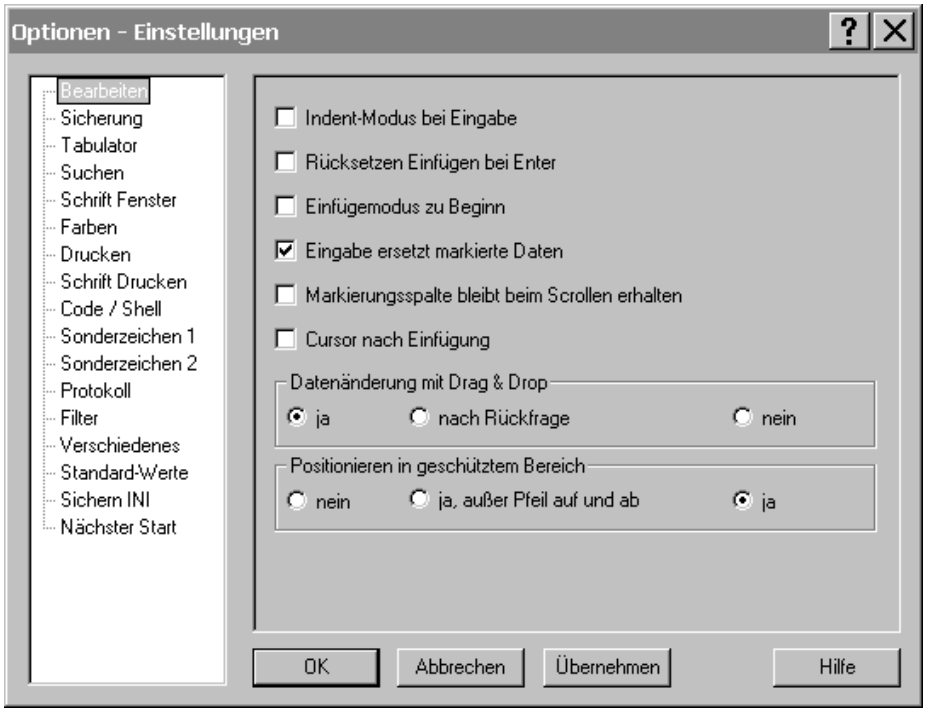
Shortcuts:

Symbol: 

Klein- / Großschreibung

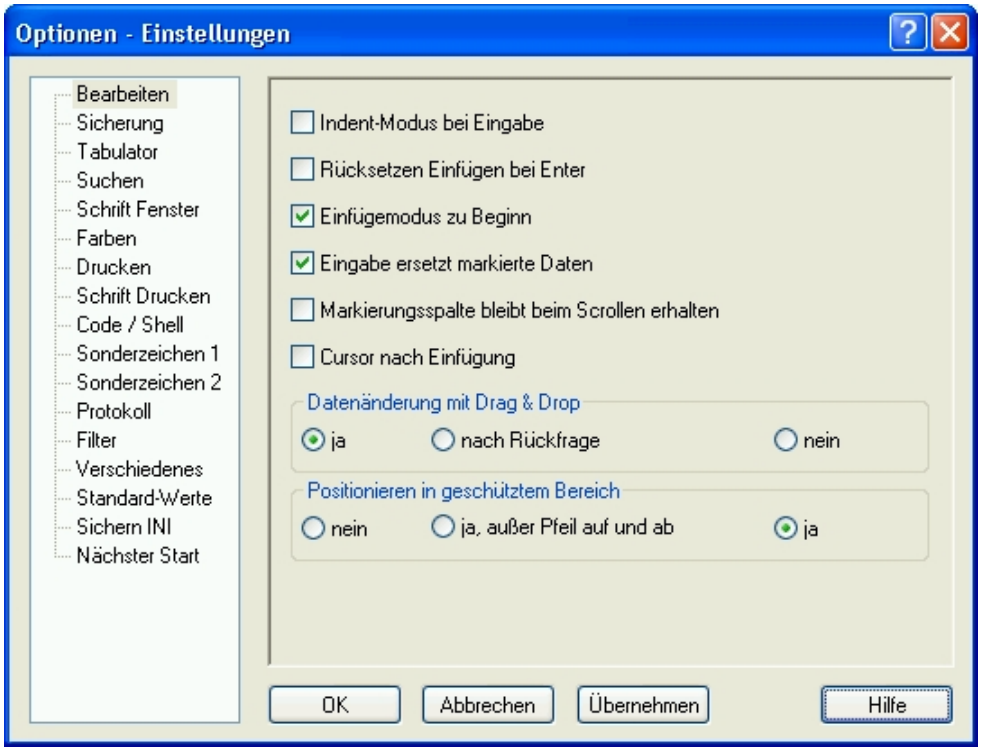
Wenn die Option nicht aktiv ist, werden neu eingegebene Kleinbuchstaben (a…z, ä, ö, ü, ß) in Großbuchstaben umgewandelt. Bereits bestehende Kleinbuchstaben werden weiterhin als solche angezeigt. Diese Option entspricht dem Kommando LOW OFF.

Einstellungen



Im linken Teil des Fensters können die einzelnen Kategorien ausgewählt werden, die dann im rechten Teil des Fensters angezeigt werden. Alle geänderten Optionen der verschiedenen Kategorien werden erst nach dem Anklicken des Button "OK" oder "Übernehmen" gültig.

Bearbeiten



Indent-Modus bei Eingabe

Cursorposition in neuer Zeile definieren.

☑ Wird mit der Taste <Enter> im Modus EDIT WORD (S. 190) in eine neue Zeile positioniert, so steht der Cursor in der Spalte, in der die vorhergehende Zeile beginnt. Dies ist z.B. beim Editieren von Primärprogrammen hilfreich.

☐ In einer neuen Zeile wird der Cursor immer auf Spalte 1 positioniert.

Die Einstellung ist auch mit dem Kommando EDIT INDENT (S. 189) möglich.

Rücksetzen Einfügen bei Enter

☑ Der Insert-Modus wird bei Betätigen der Taste <Enter> auf den Overwrite-Modus zurückgesetzt.

☐ Der Insert-Modus gilt solange, bis er durch Betätigen der Taste <Einfg> zurückgesetzt wird.

Der Modus wird in der Maske unten rechts durch die Zeichen EINF bzw. UEB angezeigt.

Einfügemodus zu Beginn

☑ Nach dem Programmstart ist der Einfüge-Modus eingeschaltet.

☐ Nach dem Programmstart ist der Überschreibe-Modus eingeschaltet.

Eingabe ersetzt markierte Daten

☑ Wenn der Cursor in einem markierten Bereich steht und Daten eingefügt werden, entweder mit der Tastatur oder mit der Funktion Einfügen des Menüs Rechte Maustaste bzw. Bearbeiten, werden die markierten Daten gelöscht und anschließend die neuen Daten in Abhängigkeit des Einfügemodus eingefügt bzw. überschrieben.

☐ Die markierten Daten werden nicht gelöscht. Die Daten werden ab der Cursorposition in Abhängigkeit des Einfügemodus eingefügt bzw. überschrieben.

Markierungsspalte bleibt beim Scrollen erhalten

☐ Die Anweisungen in der Markierungsspalte werden nur mit der Taste <Enter> übernommen. Die Anweisungen in der Markierungsspalte werden gespeichert und aus der Markierungsspalte gelöscht. Sobald die Zeile mit Eingaben in der Markierungsspalte durch Positionieren mit dem Scrollbar oder mit den entsprechenden Tasten aus dem Datenfenster verschwindet, ohne dass vorher die Enter-Taste betätigt wurde, wird die Markierung gelöscht und somit ignoriert. Diese Option entspricht der Arbeitsweise des BS2000- und MS-DOS-EDT. Es werden nur die im Dateienfenster sichtbaren Markierungen bearbeitet.

☑ Die Anweisungen in der Markierungsspalte (S. 175) werden ausgeführt, wenn die Taste <Enter> betätigt wird. Dabei werden alle Anweisungen in der Markierungsspalte berücksichtigt, die bisher im ganzen Arbeitsbereich eingegeben wurden. Verschwindet eine Zeile mit Eingaben in der Markierungsspalte durch Positionieren mit dem Scrollbar oder mit den entsprechenden Tasten aus dem Datenfenster, ohne dass vorher die Enter-Taste betätigt wurde, bleiben die Markierungsanweisungen erhalten und werden bei Betätigung der Enter-Taste ausgewertet. Diese Option führt zu einer vom BS2000- und MS-DOS-EDT abweichenden Arbeitsweise. Dies kann dazu führen, dass versehentlich eingegebene und nicht gelöschte Markierungsanweisungen ausgeführt werden, die nicht gewollt waren. Bei der Bearbeitung von großen Dateien kann außerdem eine Verzögerung bei der Bearbeitung auftreten, weil bei jedem ENTER der ganze Arbeitsbereich nach Markierungs-Anweisungen durchsucht werden muss.

Cursor nach Einfügung

☑ Nach dem Einfügen von Daten aus der Zwischenablage (Funktion Einfügen oder Überschreiben) wird der Cursor, wie im WINDOWS üblich, auf die Spalte nach den eingefügten Daten positioniert.

☐ Nach dem Einfügen von Daten aus der Zwischenablage (Funktion Einfügen oder Überschreiben) wird der Cursor auf die erste Spalte der eingefügten Daten positioniert. Diese Option kann z.B. vorteilhaft sein, wenn in einem zusammenhängenden Zeilenbereich immer ab der gleichen Spalte Daten eingefügt werden sollen. Der Cursor muss dann nicht nach jeder Einfügung wieder in die richtige Spalte positioniert werden.

Datenänderung mit Drag & Drop

ja Datenänderung mit Drag & Drop (markierten Text durch Ziehen mit der Maus verschieben und kopieren) ist zulässig.

nach Rückfrage:

Bevor die Drag & Drop-Aktion ausgeführt wird, erfolgt eine Rückfrage, ob die Aktion wirklich durchgeführt werden soll. Dadurch werden versehentliche Datenänderungen vermieden.

nein Datenänderung mit Drag & Drop ist nicht zulässig.

Positionieren Cursor in geschütztem Bereich

Es kann eingestellt werden, ob der Cursor im Datenbereich in nicht überschreibbare Zeilen positioniert werden soll:

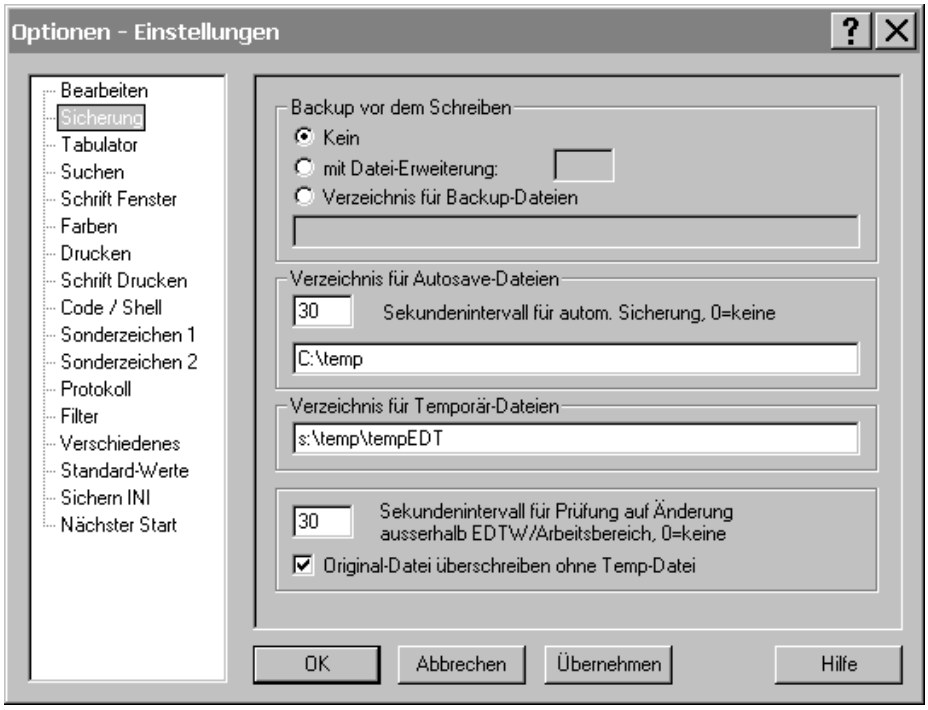
nein Wird der Cursor zu einer nicht überschreibbaren Zeile bewegt, wird der Cursor in die Markierungsspalte positioniert (wie im BS2000).

ja, außer Pfeil auf und ab

Grundsätzlich kann der Cursor auch in nicht überschreibbare Zeilen positioniert werden außer mit den Tasten <Cursor_up> und <Cursor_down>.

ja Der Cursor kann immer in nicht überschreibbare Zeilen positioniert werden.

Datei-Sicherung



Backup vor dem Schreiben

Vor dem Überschreiben einer Datei wird wahlweise eine Kopie der Originaldatei im gleichen Verzeichnis oder in einem speziellen Backup-Verzeichnis erstellt.

Datei-Erweiterung für Backup-Dateien

Namenserweiterung (Suffix) für die Sicherungskopie. Die Sicherungskopie wird vor dem Schreiben einer Datei durch Kopieren der bisherigen Datei erstellt. Ist der Parameter kürzer als drei Stellen, so wird die Namenserweiterung gebildet, indem die fehlenden Stellen vom Original-Dateinamen angehängt werden. Alternativ kann die Sicherungskopie auch in einem speziellen Verzeichnis erzeugt werden. In diesem Fall ist der Parameter `Backup-Verzeichnis` anzugeben.

Beispiel:

```
TEST.DAT   TEST.DAT  Original-Dateinamen
@          OLD       Datei-Erweiterung
TEST.@DA   TEST.OLD  Dateinamen Sicherungskopie
```

Verzeichnis für Backup-Dateien

Dateiverzeichnis, in dem die Sicherungskopien von geänderten Dateien abgelegt werden. Dieser Parameter bewirkt, dass vor jedem Zurückschreiben einer Datei der alte Inhalt der Datei unter dem Original-Dateinamen im angegebenen Verzeichnis gesichert wird. Statt eines absoluten Pfades kann auch eine Umgebungsvariable in der Form %*Variable*% angegeben werden. Der Name des Pfades wird dann beim Laden des EDT ermittelt.

Zeitintervall für automatische Sicherung

Nach Ablauf von n Sekunden wird der Inhalt des Arbeitsbereichs automatisch in eine temporäre Datei gesichert, falls die Daten geändert wurden. Enthält dieses Feld den Wert 0, erfolgt keine automatische Sicherung. Bei regulärem Programm-Ende werden die Sicherungsdateien gelöscht, während bei einem Programmabbruch, z.B. wegen Systemabsturz, diese Dateien erhalten bleiben. Wird das Programm EDT nach einer irregulären Beendigung erneut geladen, so werden evtl. vorhandene Sicherungsdateien erkannt und automatisch eingelesen.

In der Titelzeile des Datenfensters wird der Originalname mit dem Zusatz "[wiederhergestellt]" angezeigt. Die Daten können mit dem Kommando `WRITE` ohne weitere Parameter oder mit der Schaltfläche  gesichert werden. Das gilt auch für entfernte Dateien, die ursprünglich mit dem EDT-Filetransfer eingelesen wurden.

Name der Sicherungsdatei: `EDTW_`*pid_arb_pc*`.ASV` (*pid* = Prozeß-ID, *arb* = Arbeitsbereich, *pc* = Prozessorname). Zusätzlich wird eine Datei `EDTW_`*pid_arb_pc*`.ASN` erzeugt, in der das Erstellungsdatum, der Original-Dateinamen und weitere Informationen von entfernten Dateien gespeichert werden.

Verzeichnis für Autosave-Dateien

Name des Verzeichnisses für die Autosave-Dateien. Diese Dateien werden nach der regulären Beendigung des Programms gelöscht. Es kann auch ein Netzwerk-Pfad angegeben werden, da der Prozessorname im Dateinamen der Sicherungsdatei enthalten ist. Statt eines absoluten Pfades kann auch eine Umgebungsvariable in der Form %*Variable*% angegeben werden. Der Name des Pfades wird dann beim Laden des EDT ermittelt.

Standard: `%TMP%`

Tempfile-Verzeichnis

Name des Verzeichnisses für temporäre Dateien. Temporäre Dateien werden z.B. während der Ausführung des Kommandos `LIST` und entfernter `READ`, angelegt.

Statt eines absoluten Pfades kann auch eine Umgebungsvariable in der Form %*Variable*% angegeben werden. Der Name des Pfades wird dann beim Laden des EDT ermittelt. Ist das Feld leer, werden die Umgebungsvariablen `TMP` bzw. `TEMP` benutzt. Falls diese Variablen nicht vorhanden sind bzw. eine angegebene Variable leer ist, werden die Verzeichnisse `C:\TMP` oder `C:\TEMP` benutzt. Falls diese Verzeichnisse nicht vorhanden sind, wird der Verzeichnisname angefordert, sobald die erste temporäre Datei erzeugt wird.

Standard: `%TMP%`

Zeitintervall für Prüfung auf Änderung außerhalb des Arbeitsbereichs

Nach Ablauf von n Sekunden wird geprüft, ob in der Zwischenzeit die eingelesene Datei außerhalb des EDT bzw. außerhalb des Arbeitsbereichs geändert wurde. Die Prüfung wird ebenfalls durchgeführt, wenn der Arbeitsbereich gewechselt wird. Enthält dieses Feld den Wert 0, erfolgt keine Prüfung.

Hat sich eine Datei geändert (Datum und Uhrzeit neuer als der Stand zum Zeitpunkt des Einlesens), wird eine Dialogbox mit einem entsprechenden Hinweis ausgegeben. Wahlweise kann der Arbeitsbereich unverändert bleiben, die Datei neu eingelesen oder die Prüfung für diese Datei ausgeschaltet werden.

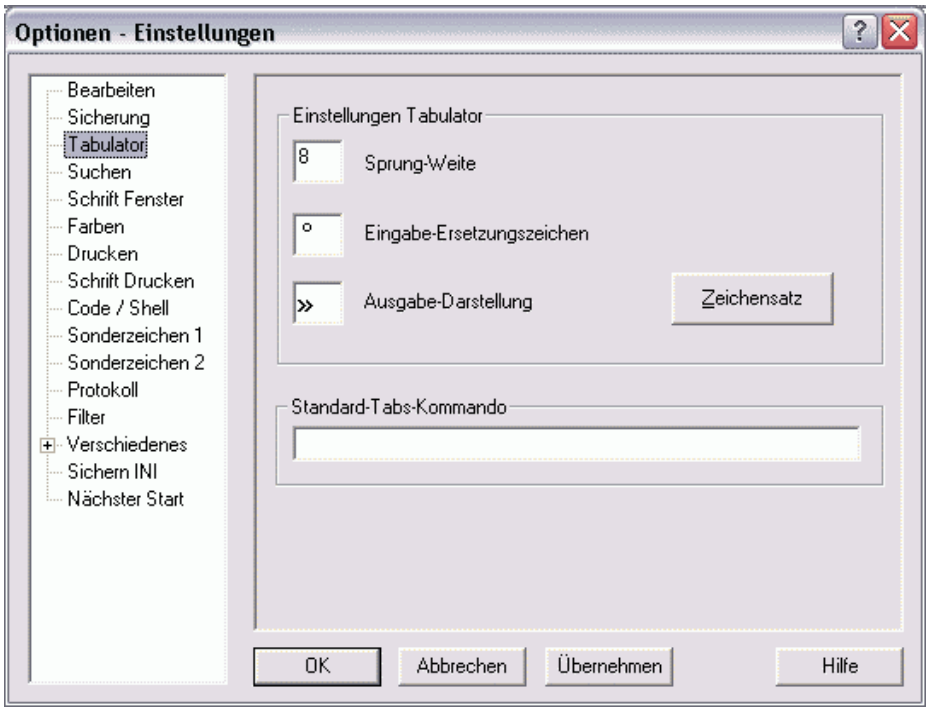
Originaldatei überschreiben ohne Temp-File

☐ Die Daten des Arbeitsbereichs werden zuerst in eine temporäre Datei im gleichen Verzeichnis geschrieben. Erst wenn dieser Vorgang fehlerfrei abgeschlossen ist, wird die Originaldatei gelöscht und die temporäre Datei in den Originalnamen umbenannt. Dies ist die sicherste Methode, um Datenverluste beim Schreiben auszuschließen.

Da bei diesem Vorgang eine neue Datei erzeugt wird, gehen die Attribute der Originaldatei (Besitzer, Gruppe, Zugriffsrechte) verloren. Diese Attribute könnten für die neue Datei nur mit Administratorrechten erzeugt werden. Insbesondere bei SAMBA-Laufwerken (Unix-Dateisysteme) kann diese Vorgehensweise zu Problemen führen (z.B. Ausführungsrechte für ein Script geht verloren).

☑ Vor dem Überschreiben von Dateien wird keine temporäre Datei erzeugt. Die Originaldatei wird direkt überschrieben. Dadurch bleiben alle Datei-Attribute erhalten. Damit in Fehler-Situationen während des Schreibvorgangs (z.B. zu wenig Speicherplatz, Leitungsprobleme) trotzdem die Originaldatei noch zur Verfügung steht, kann die Option "Backup vor dem Schreiben" (S. 106) eingeschaltet werden.

Tabulator



Standard-Tabulator-Schrittweite

Schrittweite des Tabulators. Diese Einstellung gilt für den Fall, dass in einer Datei bereits Tabulatorzeichen X'09' enthalten sind oder mit dem Eingabe-Ersetzungszeichen eingegeben wird. Die Tabulatorzeichen werden am Bildschirm mit Leerzeichen auf das nächste Vielfache von n Spalten erweitert. Beim Schreiben der Datei werden die am Bildschirm eingefügten Leerzeichen wieder entfernt.

Eingabe-Ersetzungszeichen

Zeichen, das statt der Tabulatortaste eingegeben wird. Die Tasten `<Tab_left>` und `<Tab_right>` positionieren den Cursor in das nächste bzw. vorhergehende Eingabefeld. Falls im Datenfeld ein Tabulator benutzt werden soll, muss deshalb eine Ersatz-Taste vereinbart werden. Es kann entweder ein Zeichen oder ein dreistelliger numerischer ASCII-Wert, der dem abdruckbaren Zeichen entspricht, eingegeben werden. Die Einstellung kann auch über das Kommando `HT` erfolgen.

Fehlt der Parameter, so kann im Datenfeld kein Tabulator eingegeben werden.

Das Tabulatorzeichen wird nicht unterstützt, falls für den aktuellen Arbeitsbereich ein EBCDIC-Code eingestellt ist.

Ausgabe-Darstellung

Zeichen zur Darstellung des Tabulatorzeichens. Es kann entweder ein Zeichen oder ein dreistelliger numerischer ASCII-Wert, der dem abdruckbaren Zeichen entspricht, eingegeben werden.

Parameter für das Standard-TABS-Kommando

::*tab***:***col,col...*

Im Gegensatz zu dem echten Tabulatorzeichen, das auch in der Datei gespeichert wird, gibt es im EDT auch noch einen Tabulatorfunktion, die den Cursor in eine Spalte positioniert und die Spalten davor mit Leerzeichen auffüllt (Kommando TABS siehe S. **253**). Falls eine bereits bestehende Zeile geändert wird und das Tabulatorzeichen eingegeben wird, bleibt der bisherige Inhalt erhalten.

col,col...

Wird **::***tab***:** nicht angegeben, beziehen sich die Werte auf den Hardware-Tabulator (Taste `<Tab_right>`). Beim Einfügen und Löschen werden die Daten nur innerhalb einer Spalte verschoben. Dies gilt aber nur beim Erstellen und Ändern von Daten über die Tastatur bzw. Maus im Datenbereich. Werden Daten durch Kommandos geändert, z.B. Kommando `PREFIX ON&CHANGE` usw., so wird immer der ganze Satz verschoben.

Falls in der Kommandozeile das Kommando `TABS` eingegeben wird, gelten diese Tabulatorangaben nur für die aktuelle Sitzung. Eine Speicherung dieser Werte erfolgt nicht.

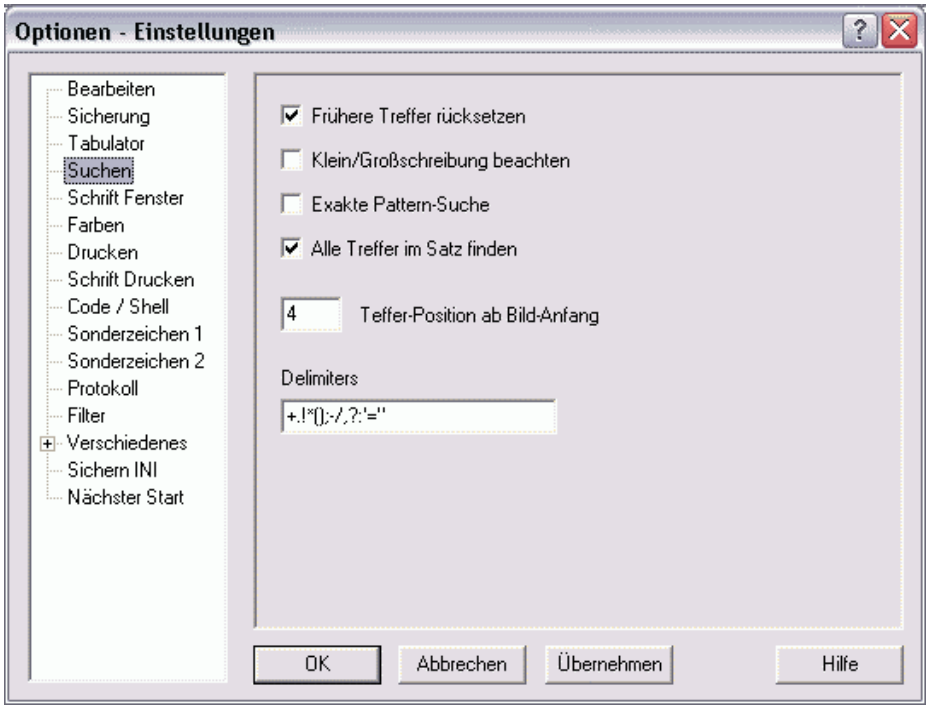
col **Tabulatorspalten für das TABS-Kommando**

Bis zu acht, durch Kommas getrennte Spaltennummern, zu denen bei Eingabe des Tabulatorzeichens positioniert wird. Befindet sich der Cursor rechts nach der letzten angegebenen Spaltennummer, gilt die Standard-Tabulatorschrittweite. Die Positionierung erfolgt sofort mit Eingabe des Tabulatorzeichens.

tab **Tabulatorzeichen für das TABS-Kommando**

Beliebiges Zeichen, das als Tabulatorzeichen interpretiert wird. Es kann entweder ein Zeichen oder ein dreistelliger numerischer ASCII-Wert, der dem abdruckbaren Zeichen entspricht, eingegeben werden.

Suchen



Rücksetzen frühere Treffer bei Suchen

☑ Die Find-Markierungen werden automatisch vor jedem Kommando `ON` *rngcol* `FIND` im gesamten Arbeitsbereich gelöscht. Siehe hierzu auch das Kommando `DMA` (S. **188**).

☐ Die Find-Markierungen bleiben erhalten.

Klein- Großschreibung beachten

Diese Option gilt sowohl für die Kommandos `ON...FIND` (S. **210**) als auch für die Funktionen `Suchen...` und `Ersetzen...` über die Menüzeile.

Es wird unabhängig von der Groß-/Kleinschreibung nach der angegebenen Zeichenfolge gesucht. Der gleiche Effekt kann auch durch Angabe der Suchzeichenfolge in der Form V'*string*' erreicht werden.

Die Option gilt nicht für einzelne Hexa-Strings.

Setzt sich ein Suchstring aus mehreren Einzelstrings zusammen, von denen ein Teil ohne und ein Teil mit Berücksichtigung der Klein-Großschreibung verglichen werden soll, wird für den ganzen String die Klein-Großschreibung nicht beachtet.

Beispiel:

Option ist deaktiviert
```
@on&find 'xxx'+V'yyy'
```

Option ist aktiviert
```
@on&find 'xxx'+x'61'
```

Die Einstellung kann auch mit dem Kommando `SEARCH-OPTION` (S. **244**) erfolgen.

Pattern-Suche exakt durchführen

Für den Parameter "PATTERN" im ON–Kommando (S. 210) kann zwischen zwei Such-Varianten gewählt werden:

☑ Der Suchstring muss exakt nach allgemeiner Syntax angegeben werden, d.h. falls sich vor oder nach dem Suchbegriff Zeichen befinden können, muss vor bzw. nach dem Suchbegriff eines der Jokerzeichen "*" oder "/" angegeben werden.

☐ Der Suchstring kann nach der BS2000-EDT-Syntax angegeben werden, d.h. vor bzw. nach dem Suchstring ist die Angabe eines der Jokerzeichen "*" oder "/" nicht notwendig.

Beispiel:

```
ON&FIND PATTERN 'ab*c'
```
wirkt wie `ON&FIND PATTERN '*ab*c*'`.

Alle Treffer im Satz finden

Wird beim Kommando ON .. FIND'*str*' (S. 212) weder die Option ALL noch ONCE angegeben, gilt der hier eingestellte Standard:

☑ Es gilt die Option ALL, d.h. alle Treffer einer Zeile werden markiert.

☐ Es gilt die Option ONCE, d.h. nur der erste Treffer einer Zeile wird markiert.

Diese Einstellung gilt nur für die einfache Variante des ON-Kommandos ohne Datenänderung, d.h. die Treffer werden nur markiert.

Anzahl Sätze vor Treffer

Nach dem Kommando ON....FIND bzw. nach der Suche über das Menü Bearbeiten wird als erste Zeile die Zeile mit dem Suchbegriff - *n* Zeilen angezeigt. Die Einstellung kann auch mit dem Kommando PAR FPOS (S. 217) erfolgen.

Aktuelle Delimiter-Zeichen

Definiert eine Menge von Zeichen, die beim Suchen einer Zeichenfolge mit dem Kommando ON als Textbegrenzerzeichen fungieren (siehe auch Parameter *str* (S. 344)). Die Begrenzungszeichen können auch mit dem Kommando DELIMIT (S. 188) definiert werden.

Schrift Fenster

Schriftart

Um die Schriftart eines Arbeitsbereichs zu ändern, wählen Sie einen Schriftartnamen.

Schriftgröße / Fett /Kursiv

Um die Schriftgröße des Arbeitsbereichs zu ändern, wählen Sie unter Schriftgröße einen Eintrag aus. Zusätzlich kann die Option "Fett" und/oder "Kursiv" ausgewählt werden.

Nur Fix-Fonts

☐ Für die Auswahl der Schriftart für das Fenster bzw. zum Drucken werden alle Schriften angeboten. Wird jedoch eine Proportionalschrift ausgewählt, wird diese in eine feste Schriftart umgewandelt. d.h. als Breite für jeden Buchstaben wird der Wert des breitesten Buchstabens benutzt. Zusätzlich kann der Prozentsatz der Verbreiterung bei der Auswahl der Schrift für Bildschirm bzw. Druck verändert werden.

☑ Für die Auswahl der Schriftart für das Fenster bzw. zum Drucken werden nur Schriften mit fester Buchstabenbreite angeboten.

Zeichenbreite Proportional-Font

Wenn eine Proportionalschriftart ausgewählt wird, so muss jeder Buchstabe auf eine mittlere Buchstabenbreite gebracht werden. Schmale Buchstaben wie I werden breiter dargestellt, Breite Buchstaben wie W werden evtl. schmäler dargestellt. Dabei kann es je nach Schriftart vorkommen, dass breite Buchstaben ineinander hineingleiten. Durch die Änderung des Prozentsatzes kann die optimale durchschnittliche Breite eingestellt werden.

Farbauswahl

Für jede Komponente im Datenfenster können die Hintergrund- und Vordergrundfarben geändert werden. Wählen Sie ein Feld aus und klicken Sie auf den Button `Vordergrund` (Farbe der Schrift) oder `Hintergrund`. Danach können Sie in einer weiteren Dialogbox (S. **114**) die Farben auswählen.

Beispieltext

In dem Feld Beispieltext wird die Hintergrund- und Vordergrundfarbe des ausgewählten Fensterteils angezeigt.

Übernehmen

Mit dem Button `Übernehmen` werden die ausgewählten Farben übernommen. Die Dialogbox wird weiter angezeigt.

Vordergrund

Es wird eine Farbtabelle zur Auswahl der Vordergrundfarbe (Schriftfarbe) angezeigt.

Hintergrund

Es wird eine Farbtabelle zur Auswahl der Hintergrundfarbe angezeigt.

Übernehmen

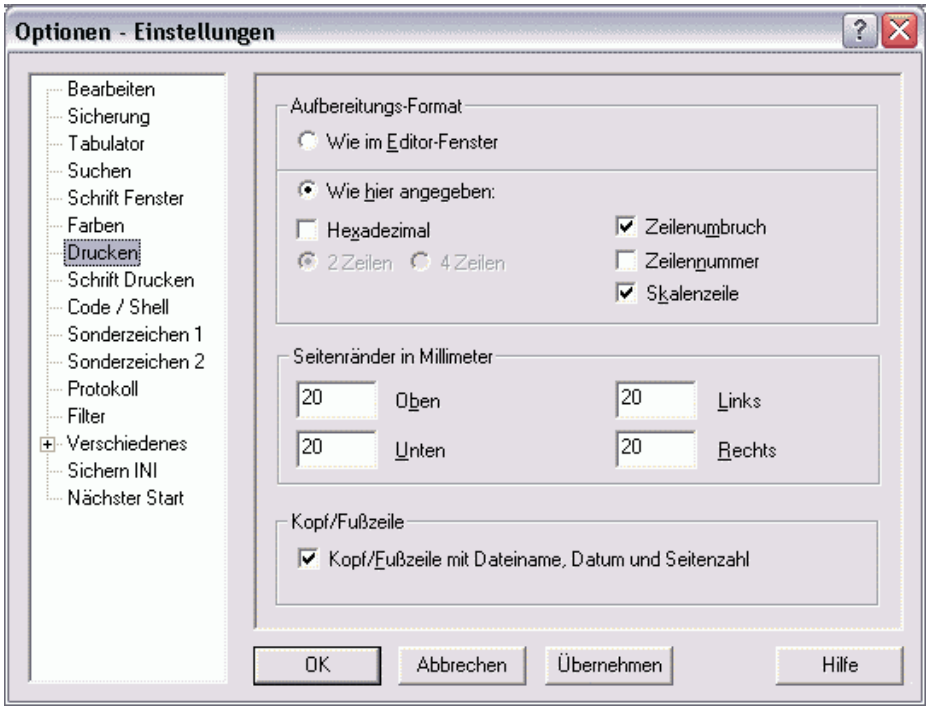
Mit dem Button `Übernehmen` werden die ausgewählten Farben übernommen. Die Dialogbox wird weiter angezeigt.

Farbtabelle

Diese Farbtabelle wird angezeigt, wenn Sie in der Dialogbox `Farbauswahl` auf die Schaltfläche `Vordergrund` oder `Hintergrund` klicken.

Wählen Sie eine Farbe aus und klicken Sie auf den OK-Button. Mit dem Button `Farbe definieren` können Sie zusätzliche Farben definieren.

Drucken...



Wie im Editor-Fenster

Die Optionen „Hexadezimal“, „Skalenzeile“, „Zeilenumbruch“ und „Zeilennummer“ werden wie in der Anzeige der Daten im Editierfenster auch für die Druckaufbereitung übernommen.

Wie hier angegeben

Die Optionen „Hexadezimal“, „Skalenzeile“, „Zeilenumbruch“ und „Zeilennummer“ werden unabhängig von der Anzeige aus den folgenden Einstellungen übernommen.

Hexadezimal

Die Daten werden in hexadezimaler Form aufbereitet.

Zeilenlineal

Nach der Kopfzeile wird ein Zeilenlineal ausgedruckt (wie Anzeige-Option SCALE ON).

Zeilenumbruch

Falls eine Zeile länger ist als die Druckbreite, wird die Zeile in voller Länge mit entsprechend vielen Fortsetzungszeilen ausgedruckt (wie Anzeige EDIT LONG). Ohne diese Option werden Zeilen nur in der Länge der Seitenbreite ausgedruckt. Im Gegensatz zur Bildschirmausgabe ist beim Drucken eine mehrzeilige hexadezimale Darstellung möglich.

Zeilennummer

Die Zeilennummer wird links neben den Daten ausgedruckt (wie Anzeige-Option INDEX ON).

Seitenränder in Millimeter

Hier können die Seitenränder unten, oben. links und rechts in Millimeter eingegeben werden.

Kopf/Fußzeile

Als Kopfzeile wird der Dateiname und als Fußzeile das Datum, die Uhrzeit und die Seitennummer ausgedruckt.

Schrift Drucken

Schriftart

Um die Schriftart für den Ausdruck zu ändern, wählen Sie einen Schriftartnamen.

Schriftgröße

Um die Schriftgröße für den Ausdruck zu ändern, wählen Sie unter Schriftgröße einen Eintrag aus. Zusätzlich kann die Option "Fett" und/oder "Kursiv" ausgewählt werden.

Nur Fix-Fonts

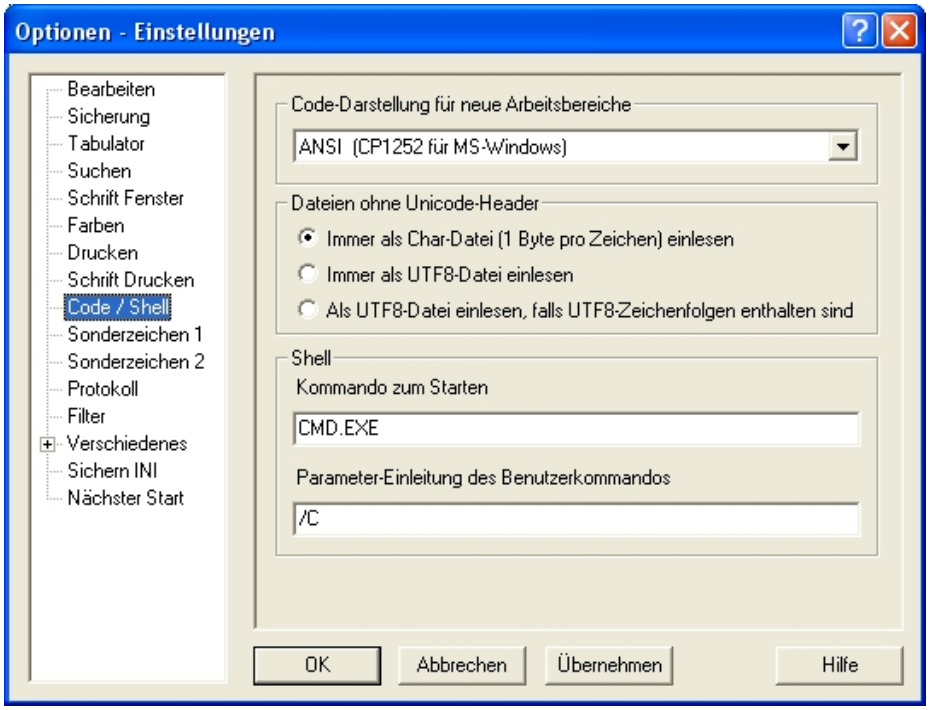
☐ Für die Auswahl der Schriftart für das Fenster bzw. zum Drucken werden alle Schriften angeboten. Wird jedoch eine Proportionalschrift ausgewählt, wird diese in eine feste Schriftart umgewandelt. d.h. als Breite für jeden Buchstaben wird der Wert des breitesten Buchstabens benutzt. Zusätzlich kann der Prozentsatz der Verbreiterung bei der Auswahl der Schrift für Bildschirm bzw. Druck verändert werden.

☑ Für die Auswahl der Schriftart für das Fenster bzw. zum Drucken werden nur Schriften mit fester Buchstabenbreite angeboten.

Zeichenbreite Proportional-Font

Wenn eine Proportionalschriftart ausgewählt wird, so muss jeder Buchstabe auf eine mittlere Buchstabenbreite gebracht werden. Schmale Buchstaben wie I werden breiter dargestellt, Breite Buchstaben wie W werden evtl. schmäler dargestellt. Dabei kann es je nach Schriftart vorkommen, dass breite Buchstaben ineinander hineingleiten. Durch die Änderung des Prozentsatzes kann die optimale durchschnittliche Breite eingestellt werden.

Code / Shell



Code-Darstellung für neue Arbeitsbereiche:

Hier kann die Code-Variante eingestellt werden, die beim Öffnen eines neuen Fensters bzw. beim Einlesen einer Datei gelten soll, soweit es sich bei einer neuen Datei nicht um eine UNICODE-Datei handelt. Wählen Sie aus der Liste die gewünschte Codierung bzw. "Auto" aus. Weitere Informationen siehe Kapitel 17 Code-Tabellen (S. 359) und 18 Unterstützung von UNICODE (S. 375).

Auto

Nach dem Öffnen eines neuen Fensters wird ANSI eingestellt. Nach dem Einlesen einer Datei wird geprüft, ob code-spezifische UNICODE-Header oder UNICODE-Satzende-Kennzeichen `X'000D000A'` bzw. `X'0D000A00'` in der Datei enthalten sind und ggf. auf diesen Code umgeschaltet.

Soweit es sich nicht um eine UNICODE-Datei handelt, werden die ersten 50.000 Bytes einer Datei mit allen Codes, die in der Datei `codepage.txt` enthalten sind, nach UNICODE umgewandelt und nach Buchstaben, Umlauten und Zahlen untersucht. Der Code mit den meisten Treffern wird ausgewählt. Bei gleichen Treffern wird ANSI bevorzugt.

In Dateien, die im Binär-Modus (Option "B" bei READ oder keine Satzende-Kennzeichen vorhanden) eingelesen werden, wird immer ANSI eingeschaltet.

Übrige Codes aus der Liste

Nach dem Öffnen eines neuen Fensters bzw. nach dem Einlesen einer Datei wird ohne weitere Prüfung der Daten der ausgewählte Code eingestellt.

Ausnahme:

Falls die Datei einen UNICODE-Header enthält (siehe Kapitel 18 UNICODE-Unterstützung (S. 375)) oder die UNICODE-Satzende-Kennzeichen `X'000D000A'` bzw. `X'0D000A00'` vorkommen, wird immer auf UNICODE umgeschaltet.

Umsetzung:

Über das Menü `Funktionen / Code-Umsetzung` (S. 83) können Sie die bestehenden Daten des aktuellen Arbeitsbereichs konvertieren.

Code für den aktuellen Arbeitsbereich einstellen

Über das Menü `Ansicht / Code-Anzeige` / (S. 94) kann eine andere Code-Variante zum Editieren und Anzeigen des aktuellen Arbeitsbereichs ausgewählt werden. Es ist zu beachten, dass dabei nicht die bestehenden Daten des aktuellen Arbeitsbereichs konvertiert werden, sondern nur der richtige Editier-Code für eine korrekte Darstellung und die Änderung von Daten ausgewählt wird.

Dateien ohne Unicode-Header:

Falls eine Datei kein UNICODE-Header enthält, werden die ersten 5.000 Bytes nach Unicode-Merkmalen durchsucht. In folgenden Fällen können UNICODE-Dateien automatisch erkannt werden:

a) Die Datei enthält das Zeilenende-Zeichen X'000D000A': Es handelt sich um eine Unicode- Datei mit UTF-16BE-Codierung (Big Endian).

b) Die Datei enthält das Zeilenende-Zeichen X'0D000A00': Es handelt sich um eine Unicode-Datei mit UTF-16LE-Codierung (Little Endian).

c) Die Datei enthält ein XML-Header mit dem Zusatz "encoding=utf": Es handelt sich um eine Unicode-Datei mit UTF-8-Codierung.

Die automatische Code-Erkennung kann beim Einlesen einer Datei mit dem Kommando READ (S. 220) und der Option B (binär), CHAR oder einer andere Code-Angabe außer Kraft gesetzt werden.

Nach diesem automatischen Suchvorgang richtet sich das weiter Vorgehen nach den folgenden Optionen:

Immer als Char-Datei (1 Byte pro Zeichen) einlesen

Dateien ohne UNICODE-Header und XML-Header werden ohne weitere Prüfung als Char-Datei (1 Byte pro Zeichen) eingelesen. Diese Einstellung gilt als Standard.

Immer als UTF8-Datei einlesen

Dateien ohne UNICODE-Header und XML-Header werden ohne weitere Prüfung als UTF8-Dateien eingelesen.

Als UTF8-Datei einlesen, falls UTF8-Zeichenfolgen enthalten sind

Dateien ohne UNICODE-Header und XML-Header werden als UTF8-Dateien eingelesen, falls in den ersten 5.000 Bytes eine UTF8-Zeichenfolge gefunden wird und keine ungültigen UTF8-Zeichenfolgen (wie z.B. bei EBCDIC-Dateien) in diesem Bereich vorkommen.

Als gültige UTF8-Zeichenfolgen gelten die Zeichenfolgen X'C2A0' - X'C2BF' (ISO-Zeichen X'A0' - X'BF') und die Zeichenfolgen X'C380' - X'C3BF' (ISO-Zeichen X'C0' - X'FF'). Dabei handelt es sich im wesentlichen um Umlaute und weitere diakritische Zeichen (Buchstaben mit Punkte, Striche, Häkchen oder Kreise), z.B. AÖÜÁÂÀÄ usw.

Diese Einstellung kann zu einer falschen Code-Erkennung führen, falls mit dem EDTW auch EBCDIC-Dateien eingelesen werden, da im EBCDIC-Code sowohl das 1. Zeichen als auch das 2. Zeichen der Zeichenfolge normale Buchstaben sind.

Die automatische Code-Erkennung kann beim Einlesen einer Datei mit dem Kommando READ (S. 220) und der Option B (binär), CHAR oder einer andere Code-Angabe außer Kraft gesetzt werden.

Kommando zum Starten der Shell

Mit dem EDT-Kommando SYS'*cmd*' (S. 178) kann ein MS-DOS-Befehl in einer DOS-Box gestartet werden. Dazu können hier Optionen festgelegt werden.

Kommando zum Starten der Shell. Es kann entweder ein Programm, in der Regel C:\COMMAND.COM (Win-NT = CMD.EXE) oder eine PIF-Datei eingetragen werden. Die Datei edtsys.pif wird bei der Installation von EDTW auf das Installations-Verzeichnis übertragen.

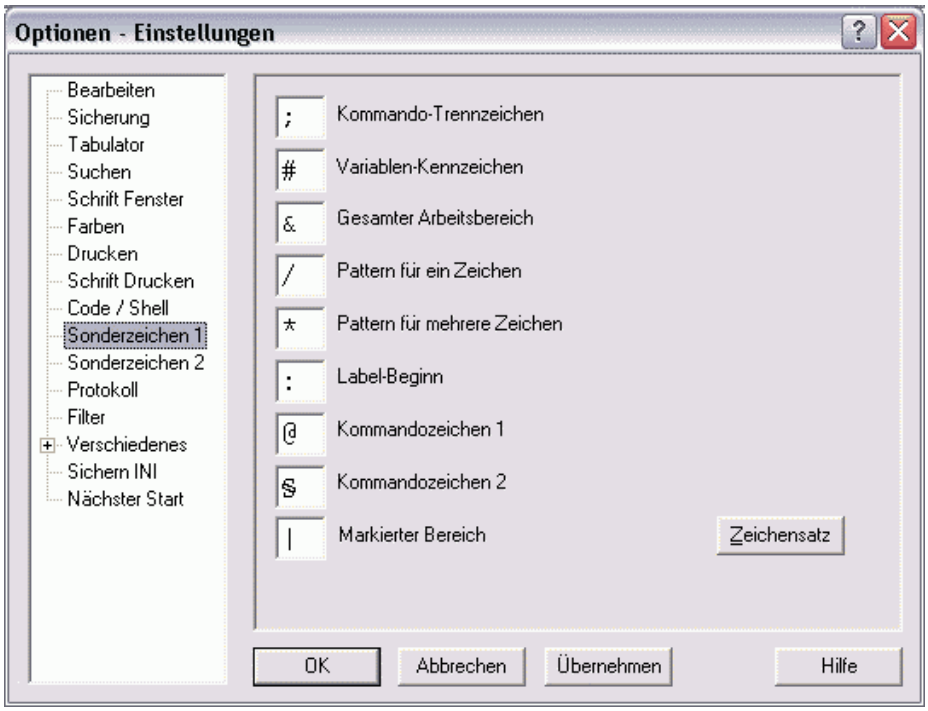
Parameter-Einleitung des Benutzerkommandos

Das MS-DOS-Kommando wird mit dem Befehlsinterpreter COMMAND.COM (Win 3.x, Win 95) bzw. CMD.EXE (Win-NT, XP, Vista) und der Option /C *befehl* oder /K *befehl* gestartet.

/C Nach Ausführung des Befehls wird der Befehlsinterpreter beendet.

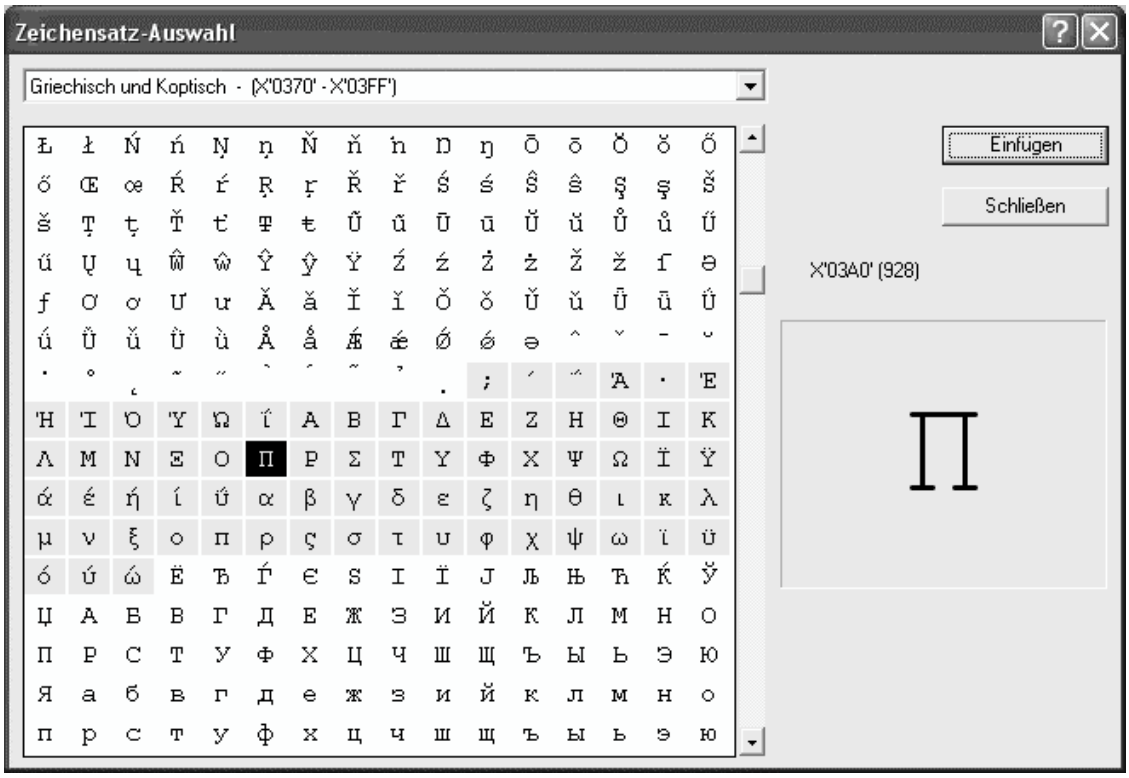
/K Nach Ausführung des Befehls wird die MS-DOS-Eingabeaufforderung angezeigt. Es können dann weitere MS-DOS-Kommandos eingegeben werden.

Sonderzeichen 1



Button Zeichensatz

Nach Anklicken dieses Buttons wird eine Dialogbox (S. **120**) mit allen möglichen Zeichen des aktuell eingestellten Zeichensatzes ausgegeben, aus der ein Sonderzeichen ausgewählt werden kann, insbesondere für Zeichen, die über die Tastatur nicht eingegeben werden können,. z.B. Nil-Zeichen, "Tabulator-Ausgabedarstellung" und "Darstellung nicht abdruckbarer Zeichen".



Button Einfügen

Das markierte Zeichen wird in das aktuelle Eingabefeld der Dialogbox `Sonderzeichen` eingetragen.

Kommando-Trennzeichen

Trennzeichen zur Trennung von mehreren Kommandos in der Kommandozeile.

Variablen-Kennzeichen

Erstes Zeichen für die Namen der Integer-, Float-, Line- und String-Variablen (#I0 bis #I99, #F0-#F99, #L0 bis #L99 und #S0 bis #S99).

Gesamter Arbeitsbereich

Symbolische Bereichsangabe für alle Zeilen eines Arbeitsbereichs.

Pattern für ein Zeichen

Musterzeichen zum Ersatz genau eines Zeichens. Dieses Zeichen kann in dem Suchstring der `ON`-Kommandos angegeben werden.

Das Zeichen kann auch mit dem Kommando `SYMBOLS` (S. **252**) umdefiniert werden.

Pattern für mehrere Zeichen

Musterzeichen zum Ersatz einer beliebig langen, auch leeren Zeichenfolge. Es wird durch die kürzeste mögliche Teilkette der überprüften Zeile befriedigt. Dieses Zeichen kann in dem Suchstring der `ON`-Kommandos angegeben werden.

Das Zeichen kann auch mit dem Kommando `SYMBOLS` (S. **252**) umdefiniert werden.

Label-Kennzeichen

Erstes Zeichen für den Namen eines Sprungziels in Prozeduren.

Kommandozeichen1

Anweisungssymbol für Kommandos in Prozedurdateien. Prozedurdateien können Kommandos und Daten enthalten. Damit die Kommandos von den Daten unterschieden werden können, muss den Kommandos dieses Zeichen vorangestellt werden.

Kommandozeichen2

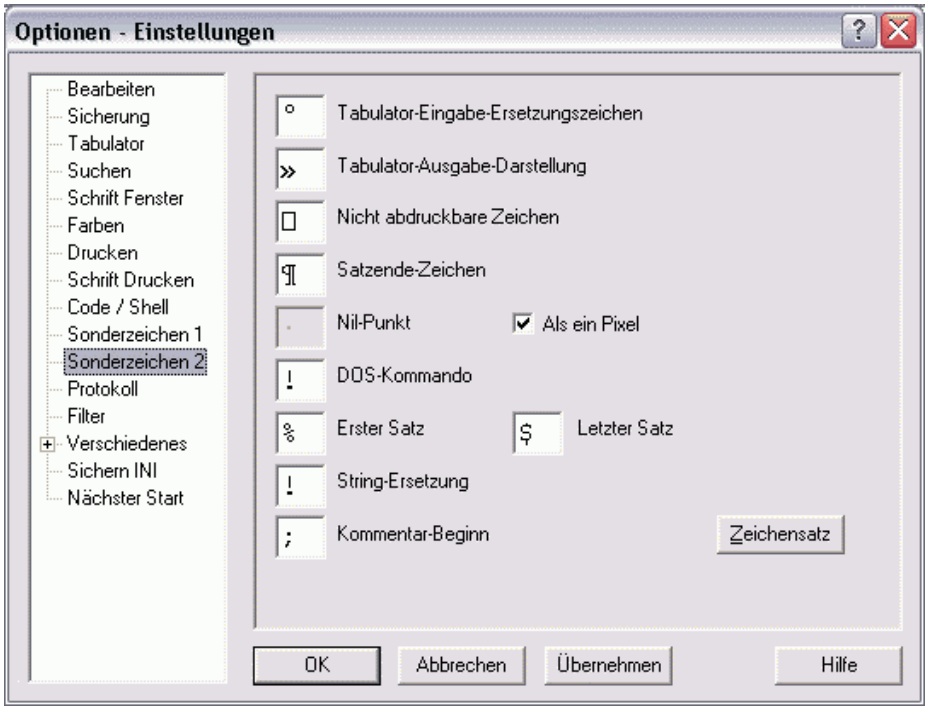
Für Tastaturen, auf denen das Anweisungssymbol nicht vorhanden ist, kann ein alternatives Anweisungssymbol definiert werden.

Markierter Bereich

Symbolisches Zeichen für den markierten bzw. alle markierten Bereiche. Dieses Zeichen kann in allen EDT-Kommandos verwendet werden, in denen eine Zeilenbereichsangabe vorgesehen ist (genauso wie das Zeichen & für alle Zeilen).

Beispiel:. `pre | with 'xxx'`

Sonderzeichen 2



Button Zeichensatz

Nach Anklicken dieses Buttons wird eine Dialogbox (S. **120**) mit allen möglichen Zeichen des aktuell eingestellten Zeichensatzes ausgegeben, aus der ein Sonderzeichen ausgewählt werden kann, insbesondere für Zeichen, die über die Tastatur nicht eingegeben werden können,. z.B. Nil-Zeichen, "Tabulator-Ausgabedarstellung" und "Darstellung nicht abdruckbarer Zeichen".

Tabulator-Eingabe-Ersetzungszeichen

Dieses Zeichen kann auch in der Dialogbox `Tabulator...` des Menüs `Optionen` geändert werden. Ausführliche Beschreibung siehe S. **108**.

Tabulator-Ausgabe-Darstellung

Dieses Zeichen kann auch in der Dialogbox `Tabulator...` des Menüs `Optionen` geändert werden. Ausführliche Beschreibung siehe S. **108**.

Darstellung nicht abdruckbarer Zeichen

Ersatzzeichen für nicht abdruckbare Zeichen.

Satzende-Zeichen

Falls im Menü `Ansicht, weitere Optionen` die Einstellung "Satz-Trennzeichen" für die Kennzeichnung des Satzendes (alternativ zu den NIL-Zeichen) aktiv ist, kann hier dieses Zeichen definiert werden.

Nilpunkt-Darstellung

Falls im Menü `Ansicht, weitere Optionen` die Einstellung "Nilzeichen" für die Kennzeichnung des Satzendes (alternativ zu den NIL-Zeichen) aktiv ist, kann hier dieses Zeichen definiert werden. Für die Darstellung des NIL-Zeichens (X'00') in den Daten und die Eingabe von NIL-Zeichen siehe das Kommando `SHOWNIL` (S. **247**). Wenn Sie "Als ein Pixel" auswählen, wird nur ein Pixel in der Zeichenmitte ausgegeben. Das Feld für die Zeichenauswahl ist dann deaktiviert.

DOS-Kommando

Statt des Kommandos `SYS` kann alternativ ein Sonderzeichen verwendet werden, das hier definiert werden kann.

Erster Satz

Symbolische Zeilennummer für die erste Zeile eines Arbeitsbereichs.

Letzter Satz

Symbolische Zeilennummer für die letzte Zeile eines Arbeitsbereichs.

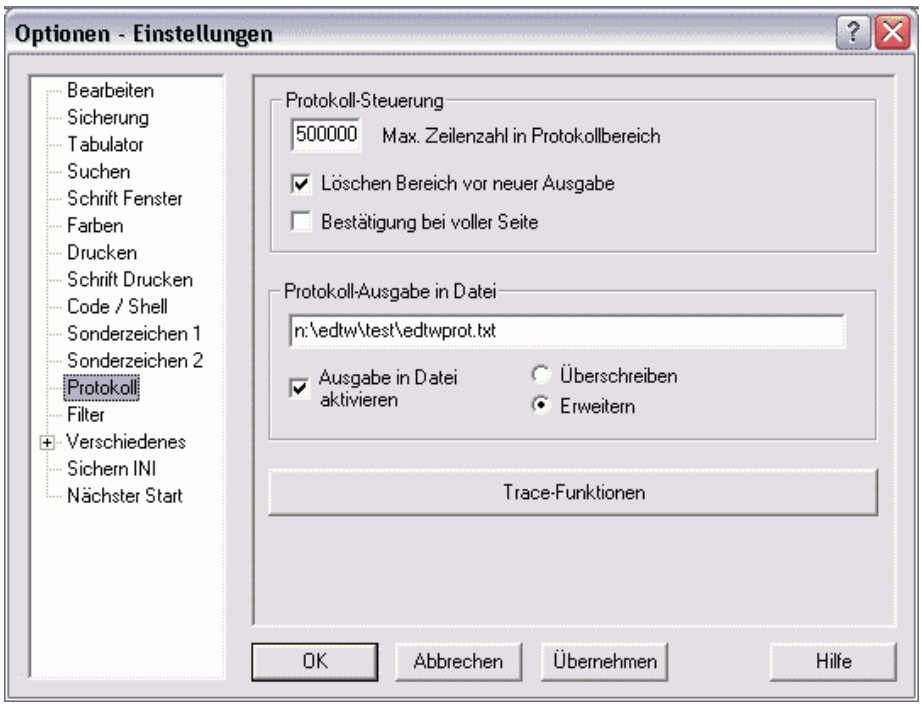
Beginn Variable für String-Substitution

Einleitungszeichen für Variablennamen in Zeichenfolgen. Die Substitution der Variablen in Zeichenfolgen wird nur durchgeführt, wenn die Option in `Optionen / Einstellungen / Verschiedenes` (S. **127**) aktiviert wird. Weitere Informationen siehe S. **344**.

Kommentar-Beginn

Trennzeichen für Kommentare nach EDT-Kommandos. Das Trennzeichen muss zweimal angegeben werden (z.B. `@rea'&file' ;;Lesen Eingabedatei`).

Protokoll



Alle variabel große Ausgaben des EDT (z.B. Kommandos `STA`, `ON&PRINT`, `PRINT`) werden in den internen Arbeitsbereich 32 geschrieben und dementsprechend auch in einem eigenen Fenster angezeigt.

Max. Satzanzahl

Maximale Anzahl von Zeilen im Protokoll-Arbeitsbereich. Dieser Parameter wirkt nur, falls die Protokolldatei laufend fortgeschrieben wird (Kontrollkästchen `Löschen Bereich ...` ist nicht aktiviert). Ist die maximale Anzahl erreicht, werden die überzähligen Zeilen ab Beginn des Arbeitsbereichs gelöscht. Die aktuelle Ausgabe wird jedoch immer vollständig in den Arbeitsbereich geschrieben.

Löschen Bereich vor neuer Ausgabe

☑ Der Protokoll-Arbeitsbereich wird vor jeder Ausgabe gelöscht.

☐ Der Protokoll-Arbeitsbereich wird laufend fortgeschrieben. Die Anzahl der Zeilen kann beschränkt werden.

Bestätigung bei voller Seite

☑ Falls die Ausgabe mehr als eine Bildschirmseite umfasst, wird vor jedem Seitenwechsel eine Bestätigung angefordert.

☐ Falls die Ausgabe mehr als eine Bildschirmseite umfasst, werden alle Daten in den Arbeitsbereich geschrieben und es wird die erste Seite angezeigt.

Protokolldatei

In Datei schreiben:

Hier kann eine Datei angegeben werden, in die alle Daten des Protokollbereichs geschrieben werden.

☑ Die Daten des Protokollbereichs werden in den Arbeitsbereich 32 und in die Protokolldatei geschrieben. Die Ausgabe in die Protokolldatei kann auch temporär mit dem Kommando `DO ...PF` (S. 277) eingeschaltet werden.

☐ Die Protokolldaten werden nur in den Arbeitsbereich 32 geschrieben.

Option Überschreiben:

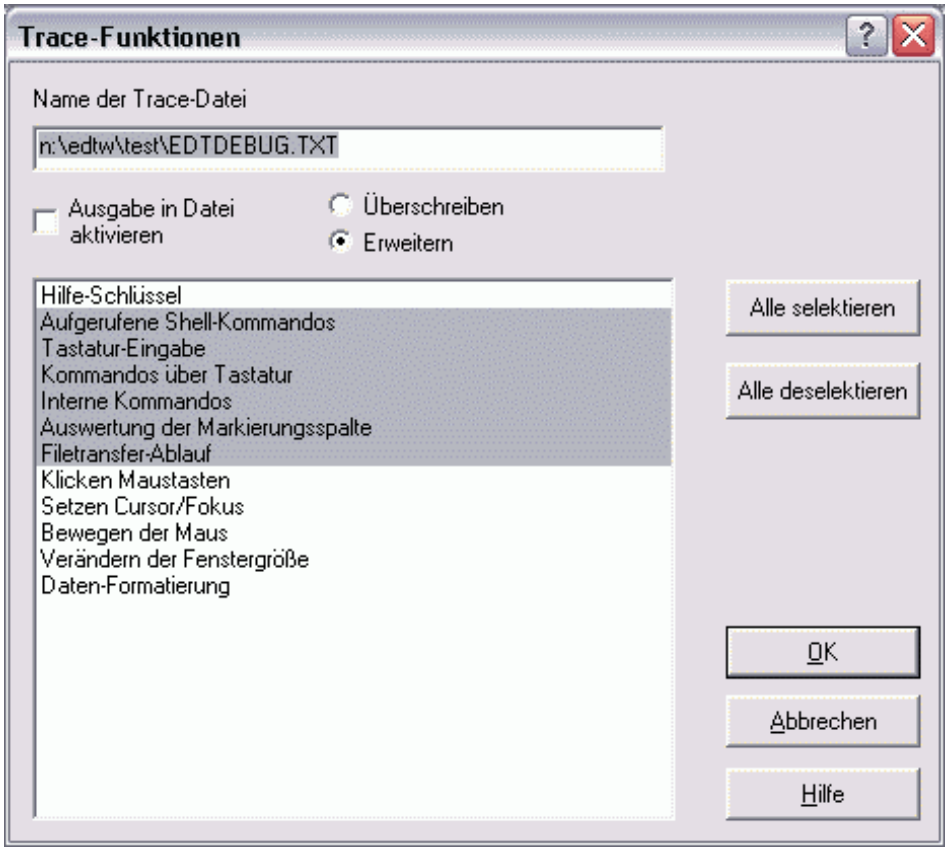
Falls die Protokolldatei bereits vorhanden ist, wird Sie vor der ersten Ausgabe gelöscht. Während einer EDT-Sitzung werden jedoch unabhängig von den Einstellungen "max. Satzanzahl.." und "Löschen Bereich.." alle Protokollausgaben fortlaufend in die Datei geschrieben.

Option Erweitern:

Falls die Protokolldatei bereits vorhanden ist, wird Sie erweitert.

Trace-Funktionen

Es wird folgende Dialogbox zur Einstellung von Trace-Funktionen für Diagnosezwecke ausgegeben.



Name der Trace-Datei

Hier kann eine Datei angegeben werden, in die alle Trace-Ausgaben geschrieben werden. In dieser Datei werden je nach gewähltem Umfang z.B. alle Kommandos, Befehle der Markierungsspalte für Diagnosezwecke protokolliert. Es kann auch die gleiche Datei wie die Protokolldatei benutzt werden. Dann werden die Protokollausgaben des Arbeitsbereiches 32 und die Trace-Daten in die gleiche Datei geschrieben.

Ausgabe in Datei aktivieren

☑ Die Protokollierung ist aktiviert. Die Ausgabe in die Trace-Datei kann auch temporär mit dem Kommando `TRACE` eingeschaltet werden.

☐ Die Protokollierung ist deaktiviert.

Option Überschreiben:

Falls die Protokolldatei bereits vorhanden ist, wird sie vor der ersten Ausgabe gelöscht.

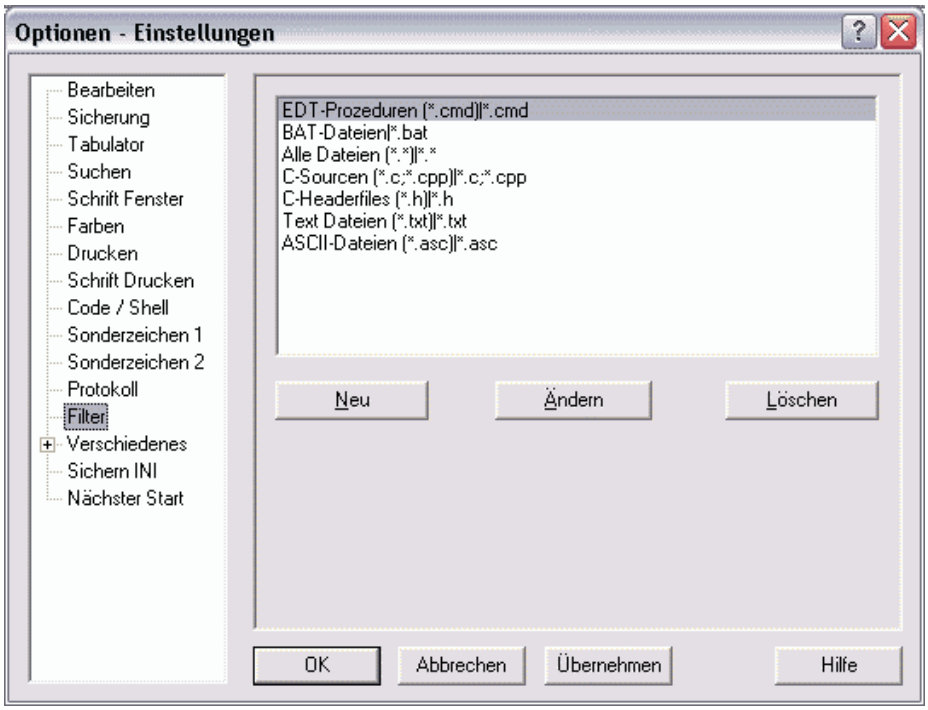
Option Erweitern:

Falls die Protokolldatei bereits vorhanden ist, wird sie erweitert.

Auswahlliste

Hier können die Aktionen ausgewählt werden, die protokolliert werden sollen. Es können auch mehrere Aktionen ausgewählt werden. Ein nochmaliges Anklicken einer bereits ausgewählten Aktion deselektiert die Aktion. Mit dem Button `Alle selektieren` können alle Aktionen ausgewählt werden. Mit dem Button `Alle deselektieren` können alle Aktionen deselektiert werden.

Filter



Listbox

Die Listbox enthält alle möglichen Varianten der Dateimuster, die in den Dialogboxen `Einlesen` und `Speichern als` zur Auswahl angeboten werden. Mit den Buttons `Neu`, `Ändern` und `Löschen` können die Elemente in der Listbox modifiziert werden. Die Dateimuster dienen nur zur Auswahl der angezeigten Dateien. Der Dateinamen wird dadurch nicht beeinflusst (z.B. durch Anhängen der Dateierweiterung).

Neu

Format der Filterangaben

beschreibung|*dateimuster*[*;dateimuster.....*]

Jeder Filtereintrag besteht aus der Beschreibung und dem Dateimuster, getrennt durch ein Pipe-Zeichen ("|").

beschreibung Die Beschreibung besteht aus einer beliebigen Zeichenfolge und wird in der Listbox des Öffnen-Dialogs angezeigt.

dateimuster Das Dateimuster besteht aus einem Suchbegriff für Dateinamen mit beliebig vielen Musterzeichen "*" und "?". Das Zeichen "*" steht dabei für eine beliebig lange, auch leere Zeichenfolge. Das Zeichen "?" steht für genau ein Zeichen. Es können auch mehrere Dateimuster, durch Semikolon getrennt, angegeben werden.

Beispiele:

```
Textdateien (*.txt)|*.txt
TXT / INI / BAT|*.txt;*.ini;*.bat
Test1|test.*
Test2 (test*.*)|test*.*
Test3 (test?.*)|test?.*
Test4 (a?e*.*)|a?e*
```

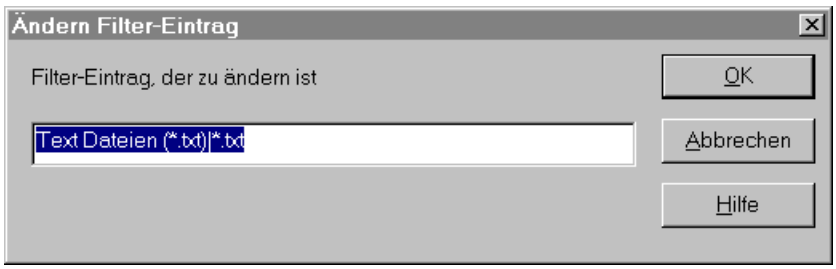
Vor aktueller Selektion

Der neue Filter wird vor dem aktuell markierten Filter in der Liste aller Filter eingefügt.

Nach aktueller Selektion

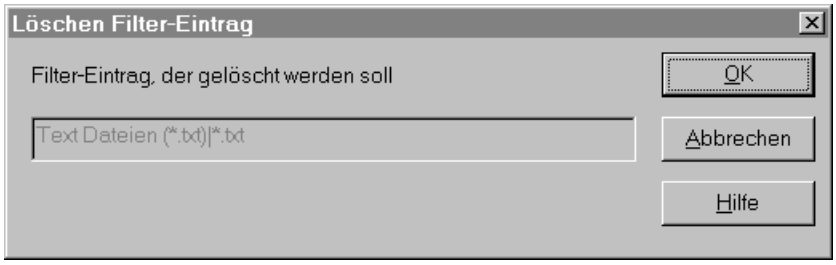
Der neue Filter wird nach dem aktuell markierten Filter in der Liste aller Filter eingefügt.

Ändern



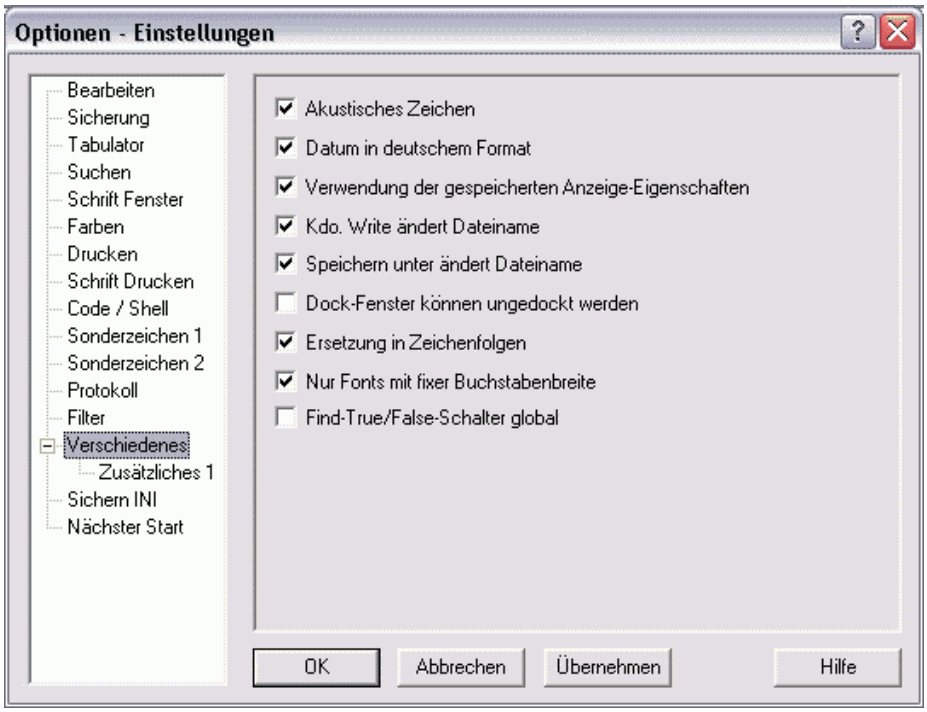
Format des Filter-Eintrags wie "Neuzugang Filter-Eintrag" (siehe oben).

Löschen



Vor dem Löschen wird der vollständige Eintrag nochmals angezeigt.

verschiedene Optionen



Akustisches Zeichen

☑ In Situationen, in denen Datenverluste eintreten können, oder bei unzulässigen Tasteneingaben wird ein akustisches Signal ausgegeben.

☐ Die Ausgabe des akustischen Signals wird unterdrückt.

Datum im deutschem Format

Bis zur Version 1.505 des EDT für MS-DOS wurden beim Kommando `SET` *stringvar*`=DATE/TIME....` (S. 299) und `SET` *linevar*`=DATE/ TIME ....` (S. 299) die Datums- und Zeitangaben in einem vom BS2000-EDT abweichenden Format aufbereitet.

☑ Dieses abweichende, nicht BS2000-kompatible Format gilt weiter. Alte Prozeduren des DOS-EDT bringen das gleiche Ergebnis wie früher.

☐ Das neue bzw. BS2000-kompatible Format gilt. Zu den Kommandos SET *stringvar*=DATE/TIME .... (S. 299) und SET *linevar*=DATE/ TIME .... (S. 299) gibt es zusätzliche Parameter, mit denen weitere verschiedene Formate (Deutsch, Englisch usw.) für das Datum und die Uhrzeit erzeugt werden.

Verwendung der gespeicherten Anzeige-Eigenschaften

☑ Nach dem Einlesen einer Datei, die in der Liste der zuletzt verarbeiteten Dateien enthalten ist (siehe Parameter Anzahl Dateinamen) werden nach dem Einlesen alle gespeicherten Einstellungen wieder aktiviert.

Folgende Eigenschaften werden für jeden Arbeitsbereich bzw. für jeden View (siehe Kommando VIEW) gespeichert:

- Position im Arbeitsbereich (erste Zeile und erste Spalte);
- Tabulatorangaben aus Kommando TABS;
- Full-Modus on/off (Kommando EDIT FULL);
- Hexa-Modus on/off (Kommando HEX);
- Scale-Modus on/off (Kommando SCALE);
- Index-Modus on/off (Kommando INDEX);
- Long-Modus on/off (Kommando EDIT LONG);
- Low-Modus on/off (Kommando LOW);
- Treffer aus Kommando ON ....FIND: Es werden sowohl die Zeile als auch die Positionen des gefundenen Suchstrings in der Zeile gespeichert. Ist der gleiche Suchstring in mehreren Views gesucht worden, so wird die Position innerhalb der Zeile nur in der View farblich markiert, in der er zuletzt ermittelt wurde.

☐ Die gespeicherten Anzeige-Eigenschaften werden nicht aktiviert.

Kommando Write ändert Dateinamen

☐ Wird nach dem Lesen einer Datei der Arbeitsbereich unter einem anderen Namen mit dem Kommando WRITE (S. 261) in eine Datei geschrieben, bleibt wie im BS2000 der ursprüngliche Dateinamen als aktuelle Datei erhalten. Dies hat zur Folge, dass in der Kopfzeile der bisherige Dateiname angezeigt wird und bei den folgenden Aktionen die eingelesene Datei überschrieben wird:

Schaltfläche 
Menü Datei / Speichern
Kommando WRITE ohne Dateinamen.

☑ Nach dem Kommando WRITE wird der neue Dateinamen zum aktuellen Dateinamen. Der neue Dateiname wird in der Kopfzeile angezeigt. Alle nachfolgenden Aktionen (siehe oben), die den Arbeitsbereich ohne Angaben eines Dateinamens in eine Datei schreiben, benutzen den neuen Dateinamen.

Speichern unter ... ändert Dateinamen

☐ Wird nach dem Lesen einer Datei der Arbeitsbereich unter einem anderen Namen mit dem Menü-Kommando Datei / Speichern unter ... (S. 62) in eine Datei geschrieben, bleibt wie im BS2000 der ursprüngliche Dateinamen als aktuelle Datei erhalten. Dies hat zur Folge, dass in der Kopfzeile der bisherige Dateiname angezeigt wird und bei den folgenden Aktionen die eingelesene Datei überschrieben wird:

Schaltfläche 
Menü Datei / Speichern
Kommando WRITE ohne Dateinamen.

☑ Nach dem Menü-Kommando `Datei / Speichern unter ...` wird der neue Dateinamen zum aktuellen Dateinamen. Der neue Dateiname wird in der Kopfzeile angezeigt. Alle nachfolgenden Aktionen (siehe oben), die den Arbeitsbereich ohne Angaben eines Dateinamens in eine Datei schreiben, benutzen den neuen Dateinamen.

Dock-Fenster können ungedockt werden

☐ Die Teilfenster (Reiter, Baumfenster, Protokollfenster) können nicht verschoben werden, sie bleiben immer am oberen, unteren bzw. linken Rand angedockt.

☑ Die Teilfenster (Reiter, Baumfenster, Protokollfenster) können beliebig verschoben werden. Sie erhalten zwei zusätzliche Schaltflächen:

 Teilfenster schließen.

 Teilfenster in freies Fenster mit Titelzeile ändern.

Wenn das Teilfenster an eine Stelle verschoben wird, die vom Fensterrand entfernt ist, wird es automatisch zu einem freien Fenster. Durch einen Doppelklick auf die Titelleiste oder durch Verschieben an den Fensterrand (Fetter Rahmen wird zur feinen Linie) wird es wieder an den Fensterrand angedockt.

Ersetzung in Zeichenfolgen

☐ In Zeichenfolgen werden spezielle EDT-System-Variablen nicht substituiert.

☑ In Zeichenfolgen werden spezielle EDT-System-Variablen, wie z.B. Dateinamen, Datum, Uhrzeit usw., substituiert. Das Einleitungszeichen für die Variablennamen kann geändert werden (Menü `Optionen / Einstellungen / Sonderzeichen2` (S. 121)) Standard = "!". Weitere Informationen siehe S. 344.

Das Ein- und Ausschalten ist auch über das Kommando `PAR VARSUBST` (S. 219) möglich.

Nur Fonts mit fixer Buchstabenbreite

☐ Für die Auswahl der Schriftart für das Fenster bzw. zum Drucken werden alle Schriften angeboten. Wird jedoch eine Proportionalschrift ausgewählt, wird diese in eine feste Schriftart umgewandelt. d.h. als Breite für jeden Buchstaben wird der Wert des breitesten Buchstabens benutzt. Zusätzlich kann der Prozentsatz der Verbreiterung bei der Auswahl der Schrift für Bildschirm bzw. Druck verändert werden.

☑ Für die Auswahl der Schriftart für das Fenster bzw. zum Drucken werden nur Schriften mit fester Buchstabenbreite angeboten.

Find-True/False-Schalter global

Die Abfrage mit `IF .TRUE./.FALSE.` (S. 282) bezog sich bis zur Version 2.24 auf das letzte ON-Kommando im aktuellen Arbeitsbereich. Im BS2000-EDT bezieht sich die `.TRUE.`- bzw. `.FALSE.`-Abfrage global auf das letzte ON-Kommando. Mit dieser Option kann das Verhalten des BS2000-EDT eingestellt werden kann.

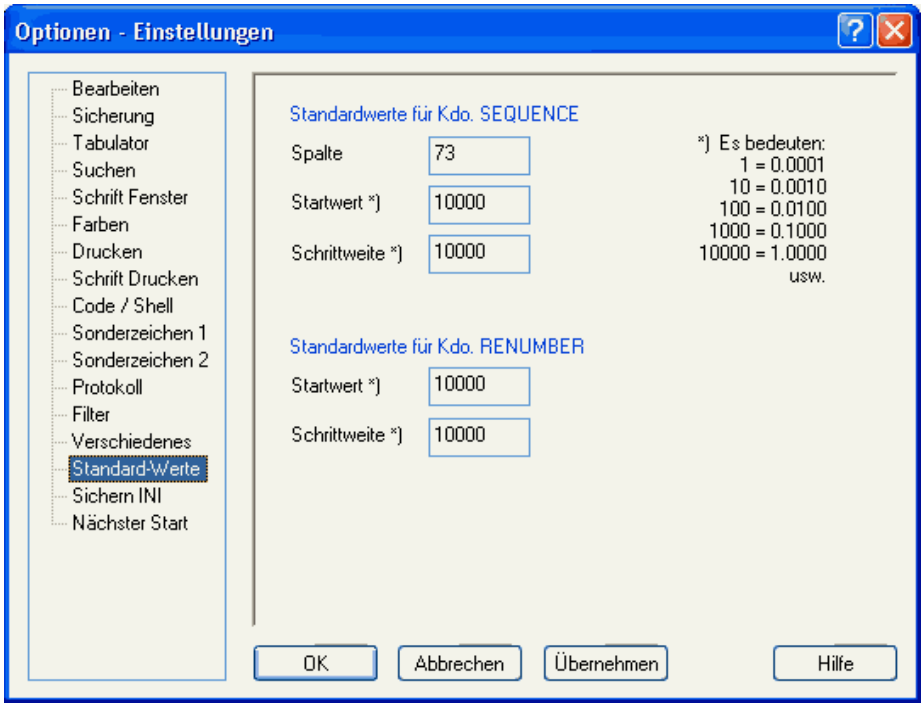
☐ Funktion wie EDT für Windows bis V 2.24: Die True-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando im aktuellen Arbeitsbereich ein Treffer gefunden wurde. Die False-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando im aktuellen Arbeitsbereich kein Treffer gefunden wurde oder im aktuellen Arbeitsbereich noch kein ON-Kommando aufgerufen wurde.

☑ Funktion wie EDT für BS2000: Die True-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando in einem beliebigen Arbeitsbereich ein Treffer gefunden wurde. Die False-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando in einem beliebigen Arbeitsbereich kein Treffer gefunden wurde oder in dieser Session noch kein ON-Kommando aufgerufen wurde.

Hinweis:

Mit dem Kommando `PAR FALSE-TRUE` (S. 219) kann der Modus für das Kommando `IF` ebenfalls eingestellt werden. Das Kommando `IF .TRUE./.FALSE.` (S. 282) wurde um die Möglichkeit erweitert, einen Arbeitsbereich anzugeben. Damit kann auch das Ergebnis eines ON-Kommandos in einem anderen Arbeitsbereich geprüft werden, ohne dass das Verhalten des BS2000-EDT eingeschaltet wird.

Standard-Werte



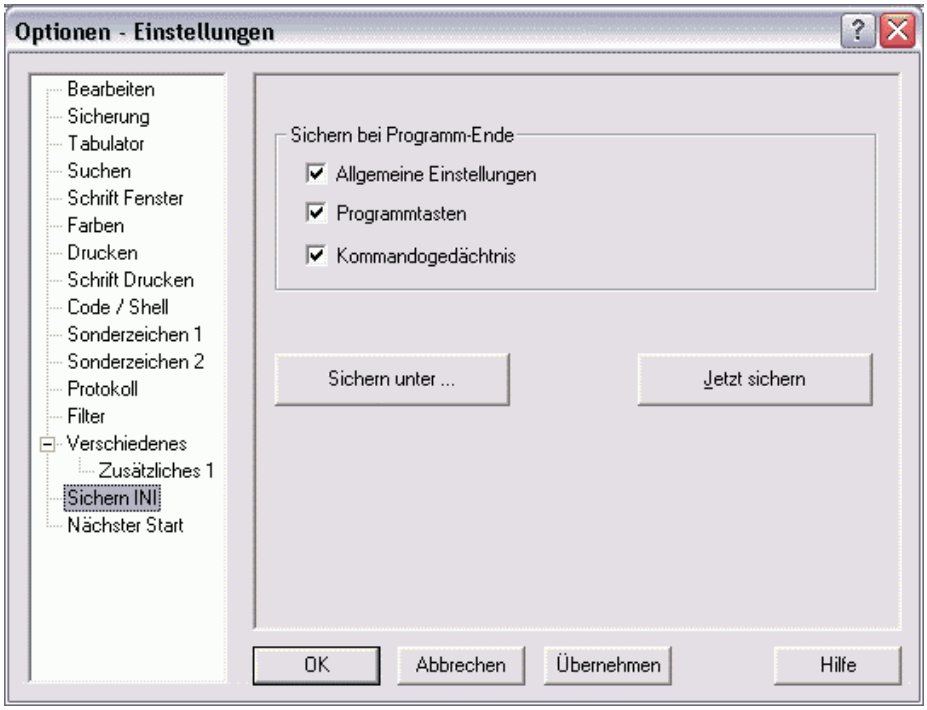
Standardwerte für das Kommando SEQUENCE

Hier kann die Spalte, der Startwert und die Schrittweite für das Kommando `SEQUENCE` (S. 245) festgelegt werden für den Fall, dass beim Kommando keine Parameter angegeben werden.

Standardwerte für das Kommando RENUMBER

Hier kann der Startwert und die Schrittweite für das Kommando `RENUMBER` (S. 236) festgelegt werden für den Fall, dass beim Kommando keine Parameter angegeben werden.

Optionen sichern



Sichern der allgemeinen Einstellungen bei Programm-Ende

☑ Alle Optionen der Menüs `Optionen` und `Ansicht` werden bei Programm-Ende in die Datei EDTW.INI gesichert und beim nächsten Laden des EDT automatisch wieder aktiviert.

☐ Die Optionen der Menüs `Optionen` und `Ansicht` werden bei Programm-Ende nicht gesichert.

Sichern der Programmtasten am Programm-Ende

☑ Die Tastenwerte werden beim Beenden von EDT in der Datei EDTW.INI gespeichert und beim nächsten Programmstart wieder geladen.

☐ Die Tastenwerte werden bei Programm-Ende nicht gespeichert.

Sichern des Kommandogedächtnisses am Programm-Ende

☑ Das Kommandogedächtnis wird beim Beenden von EDT in der Datei EDTW.INI gespeichert und beim nächsten Programmstart wieder geladen.

☐ Das Kommandogedächtnis wird bei Programm-Ende nicht gespeichert.

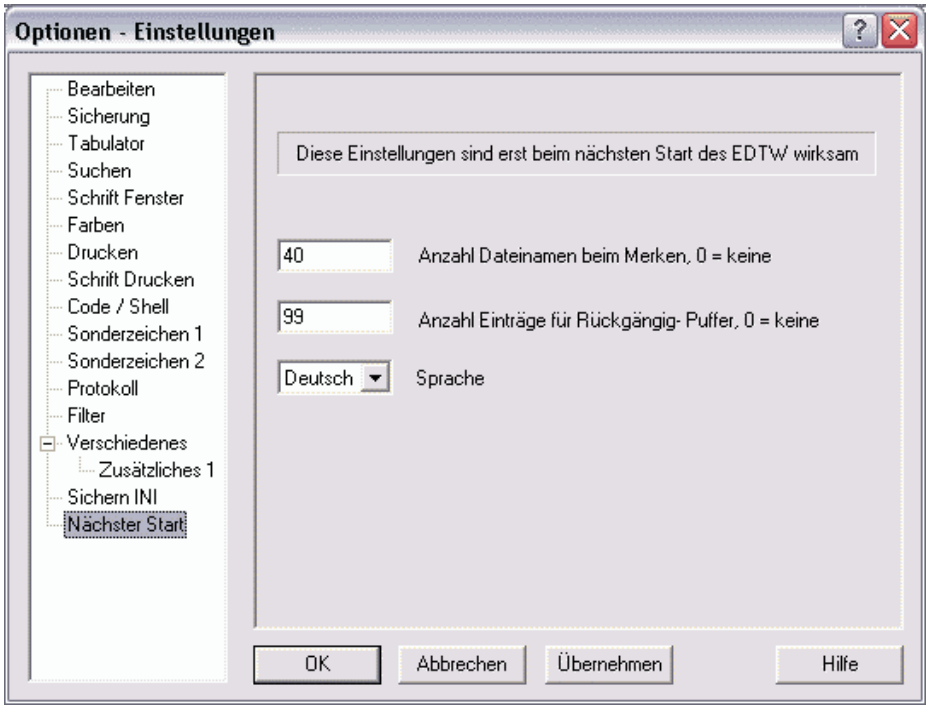
Jetzt sichern

Alle Optionen der Menüs `Optionen` und `Ansicht` werden sofort in die Datei EDTW.INI gesichert und beim nächsten Laden des EDT automatisch wieder aktiviert.

Sichern unter ...

Alle Optionen der Menüs `Optionen` und `Ansicht` werden in eine beliebige Datei gesichert, z.B. um sie bei Prozeduraufrufen mit dem Parameter -p (S. 35) zuweisen zu können. Der Dateiname wird in einer Dialogbox angefordert.

Nächster Start



Anzahl Dateinamen

Anzahl der Dateinamen, die im Menü Datei als zuletzt verarbeiteten Dateien angezeigt werden.

Anzahl Einträge für Rückgängig-Puffer

Anzahl der Einträge im Puffer für die `Rückgängig`-Aktionen. Der Eintrag des Wertes "0" bewirkt, dass die UNDO-Funktion ausgeschaltet wird. Die UNDO-Funktion kann auch temporär mit dem Kommando `UNDO ON/OFF` aus- und eingeschaltet werden.

Es ist zu beachten, dass für die UNDO-Informationen teilweise viel Speicher benötigt wird. Wenn z.B. alle Sätze mit dem Kommando ON&CHANGE geändert werden, verdoppelt sich der Speicherbedarf, weil alle Sätze zweimal vorhanden sind. Die UNDO-Informationen werden erst automatisch beim Schließen des Arbeitsbereich-Fensters bzw. nach dem Kommando READ in einem leeren Arbeitsbereich gelöscht. Auch nach einem Löschen aller Zeilen eines Arbeitsbereichs sind also noch die UNDO-Informationen verfügbar.

Im Prozedurmodus (Schalter -i beim Laden) ist die UNDO-Funktion automatisch deaktiviert.

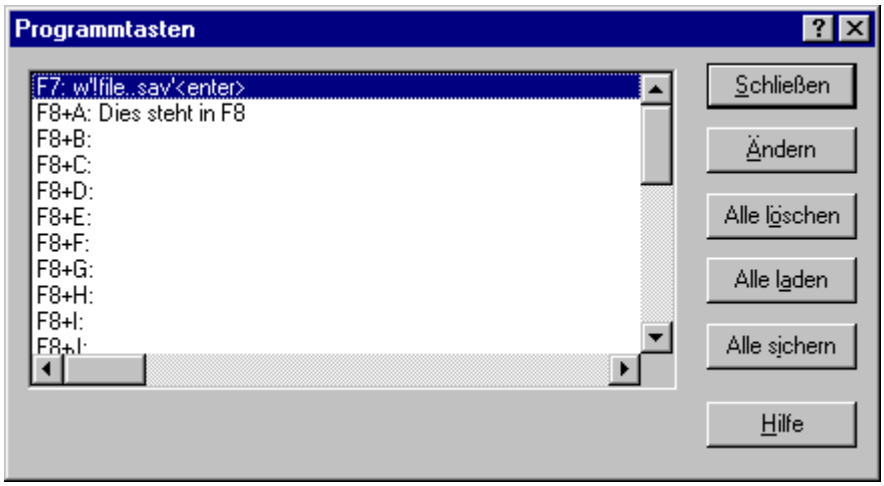
Sprache

Das Programm kann wahlweise in englischer oder deutscher Sprache (Menüs, Dialogboxen, Meldungen usw.) gestartet werden. Dieser Parameter ist erst beim nächsten Start des Programms wirksam. Beim ersten Start wird die Sprache von der Windows-Einstellung übernommen und in der Datei `EDTW.INI` gespeichert. Beim Starten des Programms kann die Sprache auch über den Schalter `-L` (S. **35**) angegeben werden. Die so eingestellte Sprache wird aber nicht gespeichert und ändert somit nicht die Einstellung in der `INI`-Datei.

Menü Extras

Tasten Programmieren

Programmtasten ....



Wartung der Programmtasten

EDT bietet die Möglichkeit, 37 Tasten mit einem beliebigen Inhalt von bis zu 256 Zeichen zu programmieren. Als Wert können alle Tasten verwendet werden, also normaler Text (Buchstaben von a - z, Sonderzeichen, Zahlen), Tastencodes wie (`ENTER, ESC` usw.) und Positioniertasten (`<Tab_left>, <Page_up>` usw.).

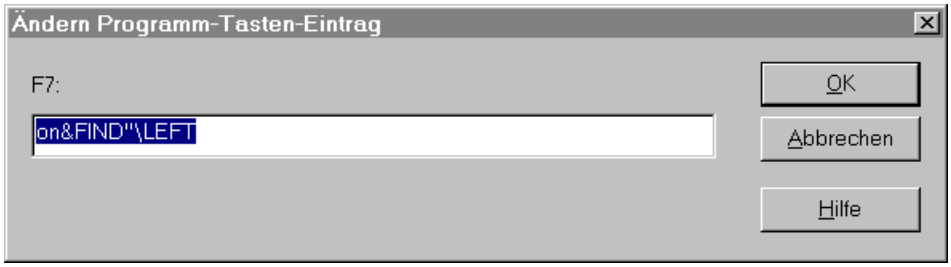
Es sind zwei verschiedene Kategorien von programmierbaren Tasten vorgesehen. Abruf des Wertes mit:

`<F7>` Der programmierte Wert kann mit einer Taste abgerufen werden. In dieser Kategorie gibt es nur eine Abrufmöglichkeit, die Taste `F7`.

`<F8> + A-Z oder <F8> + 0-9`
Für den Abruf des programmierten Wertes müssen zwei Tasten verwendet werden, nämlich die Taste `F8` als Einleitungstaste und eine der Tasten **A** bis **Z** oder **0** bis **9**. Insgesamt können also 36 Tasten mit je max. 256 Einzeltasten programmiert werden.

Shortcut: KEY

Ändern



Hier können die Daten eingegeben werden, die mit der markierten Taste aus der Listbox verknüpft werden soll.

Sondertasten müssen in die Zeichen "<" und ">" eingeschlossen werden. Falls die Tasten `<Shift>, <Cntrl>, <Alt> oder <AltGr>` zusammen mit anderen Tasten gesendet werden sollen, ist innerhalb der Tastenangabe der Präfix `S (Shift), A (Alt), C (Cntrl) oder M (AltGr)` anzugeben (z.B. `<S+F4>`). Für die Sondertasten sind folgende Schlüsselwörter zu verwenden:

<ENTER>	Neue Zeile oder Daten übernehmen
<ESC>	Aktion abbrechen
<F1...F12>	Tasten F1 bis F12
<INSERT>	Einfüge-Modus ein- bzw. ausschalten
<DELETE>	Löschen eines Zeichens rechts vom Cursor
<HOME>	Cursor in die linke Ecke des Datenfensters positionieren
<END>	Cursor an das Zeilenende positionieren
<PGUP>	Datenfenster um eine Seite in Richtung Dateiende verschieben
<PGDN>	Datenfenster um eine Seite in Richtung Dateianfang verschieben
<UP>	Cursor eine Zeile nach oben verschieben
<RIGHT>	Cursor um eine Spalte nach rechts verschieben
<LEFT>	Cursor eine Spalte nach links verschieben
<DOWN>	Cursor eine Zeile nach unten verschieben
<BACK>	Ein Zeichen links vom Cursor entfernen.
<TAB>	Tabulator nach rechts

Soll das Zeichen "<" oder ">" als Text eingegeben werden, muss "<<>" bzw. "<>>" eingegeben werden.

In Zeichenfolgen können auch spezielle EDT-System-Variablen verwendet werden, wie z.B. aktueller Dateiname, Datum, Zeit usw. Weitere Informationen siehe S. **344**.

Beispiel: `1<enter>`
Beim Abruf der Taste wird an der aktuellen Cursorposition "1" eingetragen und die Funktion `<Enter>` (neue Zeile oder Daten übernehmen) ausgeführt.

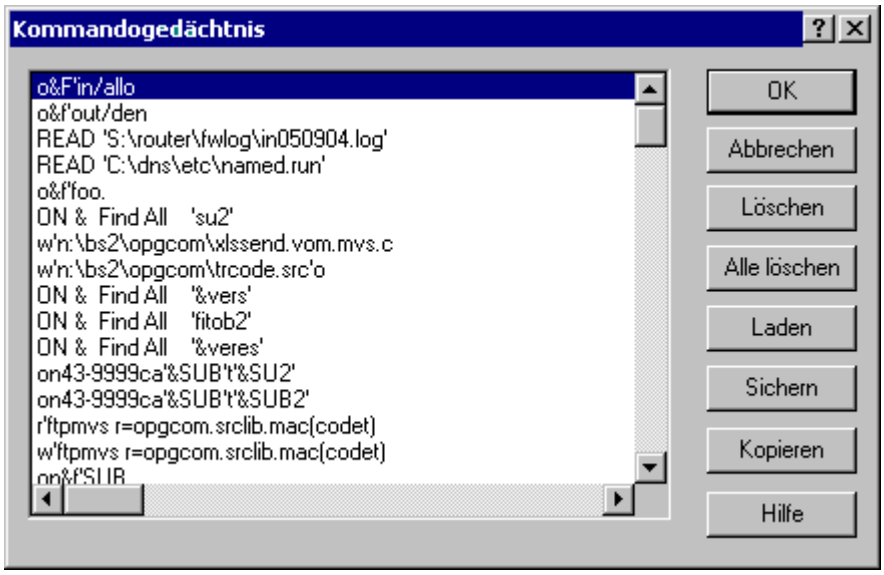
`<S+F4>`	Shift + F4
`<C+S+F4>`	Cntrl + Shift + F4
`<<>test<>>`	<test>
`w'!file..sav'<enter>`	`w'`*lokalerDateiname*.sav' + Enter

Der Inhalt der programmierbaren Tasten wird gelöscht. Wenn die Option "automatische Sicherung bei Programm-Ende" eingestellt ist, werden damit bei Programm-Ende die Tastenwerte auch in der Datei EDTW.INI gelöscht.

Die programmierbaren Tasten werden mit den Werten aus der Datei EDTW.INI mit dem Stand der letzten EDT-Sitzung bzw. der letzten Sicherung der Tastenwerte geladen.

Die programmierbaren Tasten werden in die Datei EDTW.INI gesichert. Bei Programm-Ende erfolgt in Abhängigkeit der Option "Sichern bei Programm-Ende" eine automatische Sicherung.

Kommandogedächtnis anzeigen



Shortcuts:

Symbol: CMD

Tasten: <F5> Letztes Kommando in Kommandozeile übernehmen
-<F5> Dialogbox zum Kommandogedächtnis anzeigen

Listbox mit den Kommandos

In der Listbox werden alle Eingaben der Kommandozeile angezeigt, die in der aktuellen und evtl. in vorhergehenden Sitzungen ausgeführt wurden. Mit der Maus oder den Cursortasten kann das gewünschte Kommando ausgewählt werden. Mit einem Doppelklick oder dem OK-Button wird der Dialog beendet und das Kommando in die Kommandozeile übertragen. Durch Eingabe eines beliebigen Buchstabens wird nacheinander auf die Einträge positioniert, die mit dem Buchstaben beginnen. Weitere Informationen zum Kommandogedächtnis siehe S. **351**.

Löschen

Die markierte Zeile bzw. mehrere markierte Zeilen des Kommandogedächtnisses werden gelöscht. Weitere Informationen zum Kommandogedächtnis siehe S. **351**.

Alle löschen

Der Inhalt des Kommandogedächtnisses wird gelöscht. Weitere Informationen zum Kommandogedächtnis siehe S. **351**.

Laden

Das Kommandogedächtnis wird mit den Werten aus der Datei EDTW.INI mit dem Stand der letzten EDT-Sitzung bzw. der letzten Sicherung des Kommandogedächtnisses geladen.

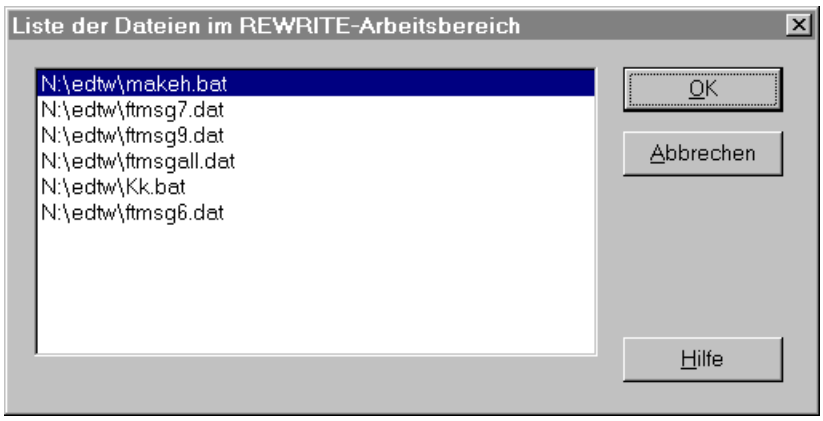
Sichern

Das Kommandogedächtnis wird in die Datei EDTW.INI gesichert. Bei Programm-Ende erfolgt in Abhängigkeit der Option "Sichern bei Programm-Ende" eine automatische Sicherung.

Kopieren

Die markierte Zeile bzw. mehrere markierte Zeilen des Kommandogedächtnisses werden in die Zwischenablage kopiert.

Dateienliste für Schreiben mehrfach



In einer Listbox werden alle Dateinamen angezeigt, die mit dem Kommando `READ` (S. **224**) oder über die Menüzeile `Datei Öffnen mehrfach` gemeinsam in einen Arbeitsbereich eingelesen wurden. Durch Doppelklick oder mit dem OK-Button kann zu der markierten Datei positioniert werden.

Shortcut:

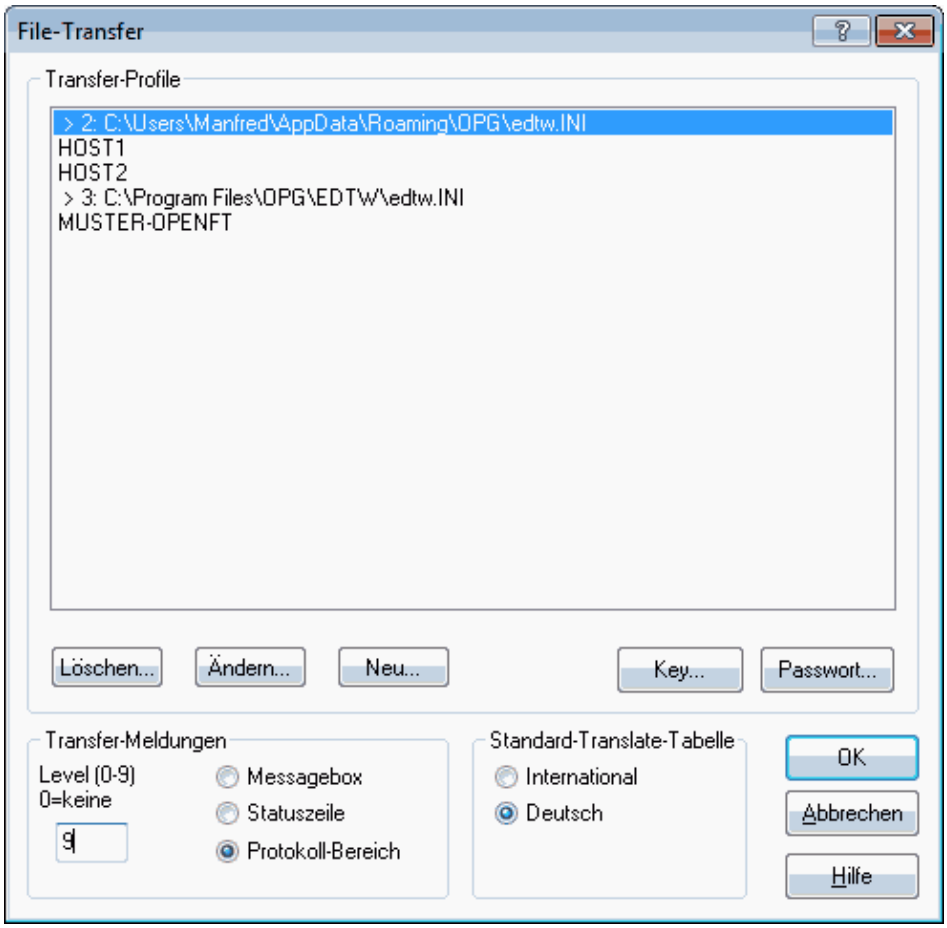
Symbol:

Filetransfer

Mit dieser Funktion können Filetransfer-Profile für die Übertragung von Daten mit openFT und FTP angelegt oder geändert werden, in denen alle allgemeinen Parameter zu und von anderen Rechnern enthalten sind. Der Filetransfer selber wird mit dem Kommandos `READ` (S. **226**) und `WRITE` (S. **264**) initiiert. Die Daten werden mit dem internen Filetransfer des EDT übertragen. Auf dem PC wird dazu keine zusätzliche Software benötigt.

Alle Parameter zu den Kommandos `READ` und `WRITE` können auch beim Kommando `FILE` (S. **192**) angegeben werden, so dass bei den nachfolgenden READ- und WRITE-Aktionen diese Parameter nicht mehr angegeben werden müssen.

Dateien können auch nach dem Empfangen von Daten durch das Programm FServer und den Browser zum BS2000 zurück geschrieben werden. Siehe dazu die Beschreibung zum Kommando `WRITE ferne Dateien` (S. 270).



Transfer-Profile

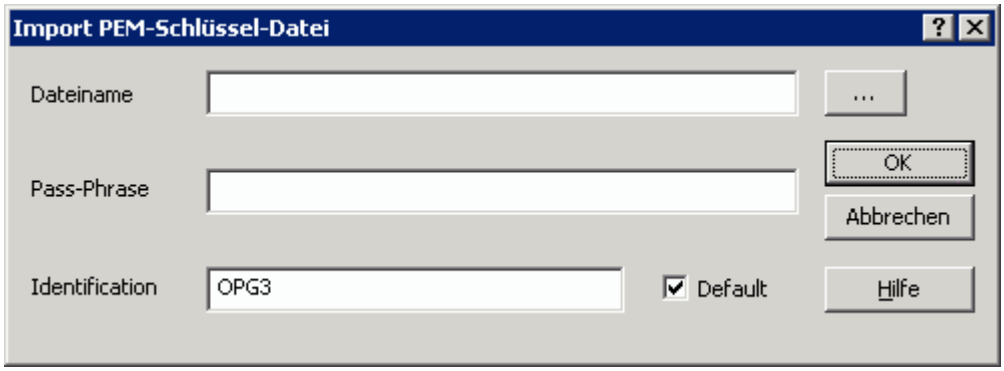
In der Listbox werden die bestehenden Profile angezeigt. Es werden Profile aus max. drei INI-Dateien ausgewertet:

1. INI-Datei mit beliebigem Namen, die über den Startparameter `-p`*filename* zugewiesen wurde.
2. `EDTW.INI` vom Verzeichnis mit dem persönlichen Daten `%APPDATA%\OPG` bzw. dem Windows-Verzeichnis.
3. `EDTW.INI` vom Ladeverzeichnis. In dieser Datei sind in der Regel die Profile enthalten, die im Falle einer Serverinstallation für alle Benutzer gelten sollen.

Die Zeilen, die mit dem Zeichen ">" beginnen, bezeichnen die INI-Datei, aus der das Profil stammt.

Mit den Buttons `Neu und Ändern` wird eine Dialogbox (S. 140) für die Bearbeitung des markierten bzw. des neuen Profils angezeigt. Mit dem Button `Löschen` kann ein bestehendes Profil gelöscht werden.

Key



Über die Import-Funktion kann ein Schlüsselpaar aus einer bereits vorhandenen, extern erzeugten **PEM-Schlüsseldatei** (z.B. mit OpenSSL command) importiert werden. Damit ist die Verwendung eines gemeinsamen Schlüsselpaares durch unterschiedliche Programme (z.B. openFT, LOG-FT und EDTW) möglich.
Falls die vorhandene PEM-Datei mit einer **Pass-Phrase** verschlüsselt abgelegt wurde, so muss das Feld Pass-Phrase mit dem entsprechenden Passwort versorgt werden.
Über **Identification** kann die Instanzidentifikation des Partnersystems gegenüber dem Host angegeben werden. Diese wird in die erzeugte Public-Key-Datei übernommen. Beim Import werden eventuell bereits vorhandene Schlüsseldateien des Benutzers im AppData-Verzeichnis überschrieben:

Default: on
`edtw_ssl_keyfile.private`
`edtw_ssl_keyfile.public`
Default: off
`edtw_ssl_keyfile.<identification>.private`
`edtw_ssl_keyfile.<identification>.public`

Bei der Durchführung einer verschlüsselten Dateiübertragung wird zuerst geprüft, ob ein Schlüsselpaar mit dem im Profil angegebenen FT-Name (entspricht: identification) vorhanden ist, ansonsten wird das Default-Schlüsselpaar verwendet.

Die erzeugte Public-Key-Datei muss für die sichere Authentifizierung dem Filetransferpartner übergeben und der darin enthaltene Schlüssel dort bereitgestellt werden.

Passwort

Alle Daten zu den Filetransfer-Profilen können in der Datei EDTW.INI aus Sicherheitsgründen verschlüsselt gespeichert werden. Ist dies gewünscht, so muss ein Passwort eingerichtet werden. Das Passwort wird für die Codierung der Daten verwendet und dient gleichzeitig als Zugangsschutz zum Ändern und Anzeigen der Filetransfer-Profile. Für jede der drei möglichen INI-Dateien kann das Passwort getrennt vergeben werden. Die Passwort-Bearbeitung mit dem Button `Passwort` gilt jeweils für die Datei, die gerade markiert ist bzw. deren Profile markiert sind. Durch das Passwort werden auch die Profilnamen geschützt, d.h. ohne Passwort wird die Liste der Profile nicht angezeigt.

Neben diesem Schutz ist es zusätzlich möglich, jedes einzelne Profil mit einem Execute-Passwort (S. **145**) zu versehen.

Transfermeldungen

Mit dem Level 0-9 kann der Detailgrad der Filetransfer-Meldungen eingestellt werden:

0 keine Meldungen
1 nur Fehlermeldungen
2 Fehlermeldungen und Transfer-Mode, Dateinamen und Dateigröße
5 wie 2, zusätzlich bestimmte interne FT-Befehle
9 wie 5, zusätzlich alle internen FT-Befehle

Die Filetransfer-Meldungen können wahlweise ein einer Messagebox, in der Statuszeile oder im Arbeitsbereich 32 angezeigt werden. Die Einstellung "Statuszeile" und "Messagebox" ist nur bei Level 1 sinnvoll. Bei der Einstellung "Messagebox" wird für jede Meldung eine eigene Messagebox ausgegeben.

Standard-Translate-Tabelle für openFT-BS2000

Diese Einstellung gilt nur, soweit im Profil das CCS EDF03 (S. 144) eingestellt ist und für die zu übertragende Datei **kein CCS-Attribut** eingestellt ist.

Ab XHCS 2.0 und openFT Version 10 gilt folgendes:

Die Code-Einstellung "Deutsch" ist nicht mehr notwendig, falls die entsprechenden Dateien mit dem CCS-Attribut EDF03DRV versehen werden. Dann wird von openFT automatisch beim Lesen die Umsetzung von 7-Bit-EBCDIC-Umlauten in 8-Bit-EBCDIC-Umlaute und beim Schreiben von 8-Bit-EBCDIC-Umlauten in 7-Bit-EBCDIC-Umlaute vorgenommen.

Mit dieser Option kann gewählt werden, ob bei der Code-Konvertierung Sonderzeichen oder deutsche Umlaute zugrunde gelegt werden sollen, falls die Datei **kein CCS-Attribut** besitzt.

Als Standard ist im BS2000 der 7-bit-Leitungscode ISO646-IRV und der dazugehörige EBCDIC-Code EBCDIC.DF.03-IRV (CCS **EDF03IRV** bzw. kein CCS) eingestellt. Wenn Sie im 7-bit-Mode arbeiten, verwenden Sie interne Tabellen. Das hat die Vorteile, dass die Arbeit unabhängig von XHCS (Verfügbarkeit, Änderung von Tabellen) ist und volle Kompatibilität mit älteren Versionen der Anwendung besteht.

Das Problem bei dieser Code-Variante besteht darin, dass bestimmte internationale Sonderzeichen und deutsche Umlaute nicht gleichzeitig dargestellt werden können. Dabei geht es um folgende Zeichen:

Deutsche Umlaute:	**Ä**	**Ö**	**Ü**	**ä**	**ö**	**ü**	**ß**
Hexa-Wert	**BB**	**BC**	**BD**	**FB**	**4F**	**FD**	**FF**
Internationale Zeichen:	[	\	]	{	|	}	~

Durch Einstellungen des Terminals bzw. der Terminalemulation kann gewählt werden, ob internationalen Zeichen oder die deutschen Zeichen dargestellt werden sollen.

In Abhängigkeit der Einstellung werden die Umlaute bzw. Sonderzeichen wie folgt übersetzt:

International

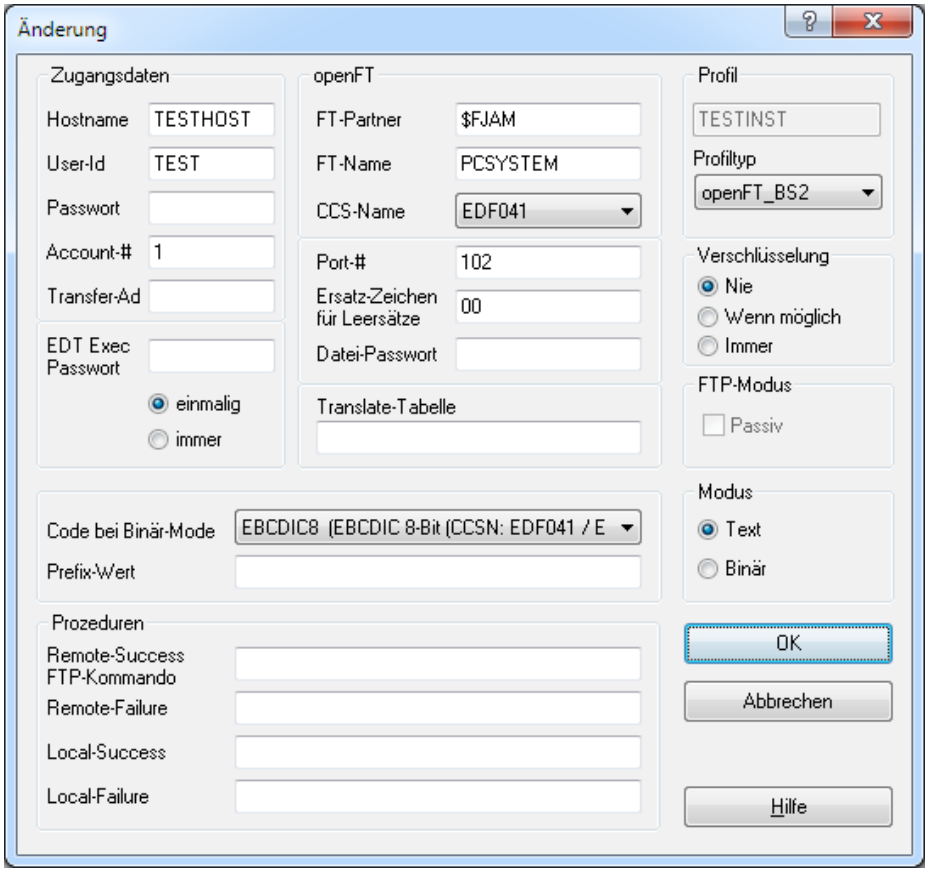
EBCDIC EDF03	HEXA	ANSI	HEXA
[oder Ä	X'BB'	[	X'5B'
\ oder Ö	X'BC'	\	X'5C'
] oder Ü	X'BD'	]	X'5D'
{ oder ä	X'FB'	{	X'7B'
| oder ö	X'4F'	|	X'7C'
} oder ü	X'FD'	}	X'7C'
~ oder ß	X'FF'	~	X'7E'

Deutsch

```
EBCDIC EDF03   HEXA       ANSI   HEXA

[ oder Ä       X'BB'      Ä      X'C4'
\ oder Ö       X'BC'      Ö      X'D6'
] oder Ü       X'BD'      Ü      X'DC'
{ oder ä       X'FB'      ä      X'E4'
| oder ö       X'4F'      ö      X'F6'
} oder ü       X'FD'      ü      X'FC'
~ oder ß       X'FF'      ß      X'DF'
```

Filetransfer-Profile ändern oder neu erstellen



Profilname

Name des Filetransfer-Profils. Dieser Name kann bei den Kommandos READ (S. 226) und WRITE (S. 264) angegeben werden.

Profiltyp

openFT_BS2

Die Datenübertragung erfolgt mit der Filetransfer-Software des EDT und dem Programm openFT. Auf dem BS2000-Rechner muss openFT zur Verfügung stehen. Gegenüber den FTP-Verbindungen bietet dieser Zugang folgende zusätzlichen Möglichkeiten:

1. Übertragung von Bibliothekselementen.

2. Daten können binär (im EBCDIC-Code) satzstrukturiert übertragen werden (Option M=BV und M=BF*nnn* zum Kommando READ (S. 226) und WRITE (S. 264)). Das gespeicherte CCS wird automatisch berücksichtigt.

openFT_Unix

Die Datenübertragung erfolgt mit der Filetransfer-Software des EDT und dem Programm openFT. Auf dem Unix-Rechner muss openFT zur Verfügung stehen. Die User-ID und der Dateinamen werden nicht in Großbuchstaben umgewandelt und der Dateinamen wird nicht um die User-ID ergänzt. Die Daten werden immer im Text-Modus übertragen. Eine Code-Umwandlung wird nur durchgeführt, falls die Daten im EBCDIC-Code übertragen werden (bei openFT bis Version 9). Die Code-Darstellung für den Arbeitsbereich wird nach dem Lesen einer Datei automatisch auf ISO8859-1 oder auf UNICODE eingestellt.

Folgende nicht benötigten Felder des Profils werden schreibgeschützt dargestellt:

Account-#:	wird vom Betriebssystem Unix nicht unterstützt
Modus:	wird nicht benötigt, nur Textmodus zulässig
Translate-Tabelle:	wird nicht benötigt
Code bei Binärmodus:	wird nicht benötigt, wird automatisch eingestellt
Datei-Passwort:	wird vom Betriebssystem Unix nicht unterstützt
CCS-Name	wird nicht benötigt

FTP_BS2000 / FTP_POSIX / FTP_UNIX / FTP_MVS / FTP_OMVS / FTP_SSH

Die Datenübertragung erfolgt mit den FTP-Funktionen des EDT und dem FTP des fernen Rechners. Auf dem fernen System muss der FTP-Server geladen sein.

Die Angabe der verschiedenen Plattformen ist notwendig, weil bei bestimmten Rechnern Besonderheiten zu berücksichtigen sind, wie z.B. anderes Layout der FSTAT-Listen im BS2000 oder andere Optionen der FTP-Funktionen.

Verschlüsselung (gilt nur für openFT)

Die Verschlüsselung der Datenübertragung verwendet das RSA/AES-Verfahren, ist aber vollkommen unabhängig von der Authentifizierung. Wenn eine verschlüsselte Übertragung gewünscht/gefordert ist, so wird von EDTW ein Schlüsselaustausch initiiert. Der Filetransferpartner muss der verschlüsselten Übertragung zustimmen und entweder den vorgeschlagenen Schlüssel akzeptieren oder eine Schlüsselbestätigung auf Basis des für den Partner hinterlegten Schlüssels initiieren. Beim Start eines Filetransfers wird, falls noch nicht vorhanden, ein konfigurationsabhängiges Default-Schlüsselpaar erstellt und in Dateiform im AppData-Verzeichnis des Benutzers abgelegt:
`edtw_ssl_keyfile.private` (für den Private Key)
`edtw_ssl_keyfile.public` (für den Public Key)
Die erzeugte Public-Key-Datei muss/kann für die sichere Authentifizierung dem Filetransferpartner übergeben und der darin enthaltene Schlüssel dort importiert/bereitgestellt werden. Als Länge des automatisch generierten Default-Schlüsselpaares wird momentan immer 2048 angenommen.

Das vom EDTW erzeugte Schlüsselpaar gilt nur für den Rechner, auf dem es erstellt wurde. Sollte sich der Rechner oder das Betriebsystem ändern, müssen die beiden Schlüssel-Dateien neu erzeugt und dem Partner bekannt gemacht werden. Dazu müssen die bereits vorhandenen Dateien umbenannt oder gelöscht werden. Anschließend wird bei der nächsten Verwendung eines FT-Profiles ein neues Schlüsselpaar erzeugt.

Es kann auch ein Schlüsselpaar aus einer bereits vorhandenen, extern erzeugten PEM-Schlüsseldatei (z.B. mit OpenSSL command) importiert (S. 138) werden. Damit ist die Verwendung eines gemeinsamen Schlüsselpaares durch unterschiedliche Programme (z.B. openFT, LOG-FT und EDTW) möglich.

Nie

Die Übertragung soll immer ohne Verschlüsselung erfolgen. Falls das Partnersystem eine Verschlüsselung zwingend fordern sollte, wird die Übertragung mit Fehler abgebrochen.

Wenn möglich

Es wird versucht eine verschlüsselte Übertragung durchzuführen. Falls das Partnersystem die Verschlüsselung nicht unterstützen sollte, wird die Übertragung ohne Verschlüsselung durchgeführt.

Immer

Die Übertragung soll zwingend mit Verschlüsselung erfolgen. Falls das Partnersystem eine Verschlüsselung nicht unterstützen sollte, wird die Übertragung mit Fehler abgebrochen.

FTP-Modus (gilt nur für FTP)

Passive Modus

Eine FTP-Verbindung besteht aus zwei Kanälen: dem Befehlskanal und dem Datenkanal. Im aktiven Modus (Standard) verbindet sich der Client mit dem Server, um den Befehlskanal zu erstellen, aber der Server verbindet sich wieder mit dem Client, um den Datenkanal zu erstellen. Das Problem ist, dass, wenn sich der Client hinter einer Firewall befindet, Remoteverbindungen blockiert werden können. In einem solchen Fall kann der passive Modus nützlich sein.

Der passive Modus ermöglicht es dem FTP-Client, beide Kanäle zu erstellen, so dass die Firewall die FTP-Verbindung nicht blockiert.

Hostname

Rechnername des entfernten Rechners, wie er in der Datei `hosts` im WINDOWS-Verzeichnis (Standard = `C:\WINDOWS` bzw. `C:\WINNT\SYSTEM32\DRIVERS\ETC`) definiert ist. Statt des symbolischen Rechnernamens können Sie auch direkt die IP-Adresse (z.B. 123.12.123.1) angeben. Bitte beachten Sie, dass ein- oder zwei-stellige Teilnummern ohne führende Nullen anzugeben sind (z.B. 194.11.222.1).

Wird ein Fragezeichen (?) eingetragen, so wird bei der Aktivierung des Filetransfers der Hostname. angefordert. Der Hostname kann auch bei den Kommandos `FILE` (S. **192**), `READ` (S. **226**) und `WRITE` (S. **264**) mit dem Parameter LH angegeben werden.

USER-ID, Account-# und Passwort oder FTAC-Profil

Zugangsdaten für den fernen Rechner:

openFT_BS2 und openFT_Unix: Es bestehen zwei Alternativen:

a) USER-ID, Account-# und wahlweise das Passwort wie beim LOGON-Kommando.

b) Transfer-Admission eines FTAC-Profils, in dem die Zugangsdaten und die Zugriffsrechte festgelegt sind.

FTP_BS2 / FTP_POSIX / FTP_UNIX / FTP_MVS / FTP_OMVS / FTP_SSH:

Benutzername (im Feld USER-ID) und Passwort wie beim LOGIN. Das Feld Account-# bleibt leer.

User-ID

Die USER-ID ist im BS2000 ohne "$" einzugeben. Wird ein Fragezeichen (?) eingetragen, so wird bei der Aktivierung des Filetransfers mit dem Kommando `READ` oder `WRITE` die USER-ID. angefordert. Die USER-ID kann auch bei den Kommandos `READ` und `WRITE` über den Parameter `LU=`*user* angegeben werden.

Logon-Passwort

Das Passwort kann im Format `cccc`, `'cccc'`, `X'xxxxxxxx'` (nur BS2000) oder `?` eingegeben werden. Wird ein Fragezeichen (?) eingetragen, hat das zur Folge, dass bei der Aktivierung des Filetransfers mit dem Kommando `READ` oder `WRITE` das Passwort in einem dunkel gesteuerten Feld angefordert wird. Das Passwort kann auch bei den Kommandos `READ` und `WRITE` über den Parameter `LP=`*pass*|`?` angegeben werden.

Account-#

Abrechnungs-Nummer für BS2000-Filetransfer. Wird ein Fragezeichen (?) eingetragen, so wird bei der Aktivierung des Filetransfers mit dem Kommando `READ` oder `WRITE` die Abrechnungs-Nummer angefordert. Das Passwort kann auch bei den Kommandos `READ` und `WRITE` über den Parameter `LA=`*account* angegeben werden.

Transfer-Ad

FTACTransfer-Admission: FTAC-Profile können mit der Software FTAC erstellt werden. Die Transfer-Admission muss mindestens 8 Stellen und darf höchstens 32 Stellen lang sein. Enthält der Name Blanks muss er in Hochkommas eingegeben werden. (gilt nur für openFT_BS2).

Wird ein Fragezeichen (?) eingetragen, so wird bei der Aktivierung des Filetransfers mit dem Kommando `READ` oder `WRITE` die Transfer-Admission angefordert. Die Transfer-Admission kann auch bei den Kommandos `READ` und `WRITE` über den Parameter `TA=`*ta* angegeben werden.

Partnername des fernen Systems (gilt nur für openFT)

BS2000

Stationsname der FT-Kopfstation im Partnersystem, unter dem der FT-Partner im Prozessor NETWORK-ADDRESS bekannt ist, in der Regel $FJAM (Parameter `ENTITY` des BS2000-Kommandos `ADD-FT-PARTNER`).

Lokaler FT-Name (gilt nur für openFT)

Prozessorname, wie er im Netzbeschreibungsbuch des openFT eingetragen ist. Über den Prozessornamen PCSYSTEM oder ANYBODY ist es möglich, von mehreren PC's Daten zu übertragen, ohne dass für jeden PC ein FT-Partner eingetragen wird.

Es ist der Prozessorname einzutragen, wie er im BS2000-Kommando ADD-FT-PARTNER beim Parameter PARTNER-ADDRESS=*FTNEA-ADDRESS(PROCESSOR=<alphanum-name1..8>) bzw. PARTNER-ADDRESS=*OPENFT-ADDRESS(PROCESSOR=<alphanum-name1..8>) eingetragen wird (siehe auch Kommando SH-FT-PARTNERS, Spalte FTNEA-ADDRESS oder OPENFT-ADDRESS). Bei BS2000-Partnern ist der Name identisch mit dem in BCAM festgelegten Host-Namen; bei SINIX-Partnern wird der Name durch Angabe der FT-Betriebsparameter festgelegt.

Mit dem Kommando SH-FT-LOG NUM=10 können Sie sich über die letzten 10 Datei-Übertragungsaufträge informieren, die openFT-BS2000 bislang protokolliert hat. Unter der Spalte PARTNER können Sie feststellen, welcher Prozessorname verwendet wurde. Weitere FT-Optionen können Sie mit dem BS2000-Kommando MOD-FT-OPTIONS einstellen bzw. mit dem Kommando SH-FT-OPTIONS auflisten.

Datei-Passwort (gilt nur für openFT-BS2000)

Passwort für die Datei im fernen System im Format 'cccc', cccc, X'xxxxxxxx' oder ?. Wird ein Fragezeichen (?) eingetragen, hat das zur Folge, dass bei der Aktivierung des Filetransfers mit dem Kommando READ oder WRITE das Passwort in einem dunkel gesteuerten Feld angefordert wird. Das Passwort kann auch bei den Kommandos READ und WRITE über den Parameter FP=*pass*|? angegeben werden.

Socket-Portnummer für Filetransfer

Hier kann die Socket-Portnummer für den Filetransfer eingegeben werden (openFT-Standard = 1100, früher 102, FTP-Standard = 21).

Ersatzzeichen für Leersätze

Diese Option ist hauptsächlich für den Filetransfer mit dem BS2000 gedacht. Im BS2000 sind Leersätze nicht üblich. Der BS2000-EDT zeigt zum Beispiel Leersätze überhaupt nicht an. Damit auch im BS2000-EDT Leersätze von Dateien, die mit dem EDT für Windows erstellt worden sind, angezeigt werden können, muss ein Satz mit einem Byte erzeugt werden, z.B. ein Leerzeichen oder das Zeichen X'00'.

Soll für einen Leersatz ein Datensatz mit einem Ersatzzeichen erstellt werden, so ist in diesem Feld das Ersatzzeichen hexadezimal im Code ANSI einzugeben. Beim Schreiben von Dateien wird aus Leersätzen ein Satz, der nur aus diesem Zeichen besteht, erzeugt, beim Lesen wird aus Sätzen, die nur aus diesem Zeichen bestehen, ein Leersatz erzeugt.

CCS-Name

Hier kann ein CCS (Coded Character Set) ausgewählt werden, mit dem festgelegt wird, wie die Datei im BS2000 codiert ist. **Beim Schreiben von neuen Dateien** wird das CCS als Dateiattribut übernommen. Eine vom BS2000 eingelesene Datei wird immer, unabhängig von den Angaben im Profil mit den im **BS2000 gespeicherten CCS** zurückgeschrieben.

Als Standard ist das CCS EDF03 eingestellt (wie bisher intern auch verwendet). Bei diesem CCS wird wie bisher die Einstellung Deutsch oder International in der Dialogbox Extras / Filetransfer (S. 139) für die Codierung verwendet.

Bei dem CCS-Namen **EDF03-Deutsch** wird unabhängig von der globalen Einstellung immer die Deutsche Variante für die Codierung verwendet. Das dazugehörige CCS EDF03DRV wird jedoch aus Kompatibilitätsgründen nicht an den openFT weitergegeben. Falls trotzdem dieses CCS gesetzt werden soll, muss es als Parameter CCS beim Kommando `WRITE` (S. **264**) angegeben werden.

Bei dem CCS-Namen **EDF03-International** wird unabhängig von der globalen Einstellung immer die Internationale Variante für die Codierung verwendet. Das dazugehörige CCS EDF03IRV wird jedoch aus Kompatibilitätsgründen nicht an den openFT weitergegeben. Falls trotzdem dieses CCS gesetzt werden soll, muss es als Parameter CCS beim Kommando `WRITE` (S. **264**) angegeben werden.

Bei allen anderen Profilen wird unabhängig von der globalen Einstellung immer die Internationale Variante (ISO8859-1 → EDF041) für die Codierung verwendet.

Bis Version 9 des openFT im BS2000 sind folgende Einschränkungen zu beachten:

a) CCS EDF041 wird nur als Dateiattribut übernommen, wenn das CCS EDF041 auch als Standard-CCS (global oder nur für die User-ID) eingestellt ist.

b) Falls die Daten binär übertragen werden (`M=B` oder `B=BF`), wird das CCS nicht als Dateiattribut übernommen.

Das CCS kann auch beim Kommando `WRITE` (S. **264**) angegeben werden. In diesem Fall hat das CCS des Kommandos `WRITE` Vorrang vor dem CCS im Profil.

Execute-Passwort

Passwort für die Verwendung des Filetransfer-Profils. Ist hier ein Passwort eingetragen, so wird beim Starten des Filetransfers über die Kommando `READ` oder `WRITE` ein Passwort angefordert und mit dem gespeicherten Execute-Passwort verglichen. Über den Radio-Button kann gewählt werden, ob das Passwort bei jedem Lesen bzw. Schreiben oder nur bei der ersten Verwendung dieses Profils angefordert werden soll.

Translate-Tabelle (gilt nur für openFT_BS2)

Dateiname einer Datei mit Translate-Anweisungen EBCDIC <-> ANSI. Die Standard-Übersetzungstabelle (S. **373**) für die Konvertierung von EBCDIC nach ANSI und umgekehrt ist im Programm enthalten. Eine abweichende Konvertierung kann erreicht werden, indem für einzelne Zeichen oder für alle Zeichen Translate-Informationen in dieser Datei zur Verfügung gestellt werden. Eine Musterdatei befindet sich auf dem Installationsverzeichnis des EDT unter dem Namen EDF041.TrT.

Syntaxbeschreibung der Datei:

```
e2a xx yy bemerkung
a2e yy xx bemerkung
```

`e2a`	Translate-Anweisungen EBCDIC --> ANSI
`a2e`	Translate-Anweisungen ANSI --> EBCDIC
`xx`	Hexadezimaler Code EBCDIC
`yy`	Hexadezimaler Code ANSI
bemerkung	Wahlweise Bemerkung

Die einzelnen Parameter sind durch eine Leerstellen getrennt (vor der Bemerkung beliebig viele Leerstellen).

Beispiel (Umlaute ANSI <--> EBCDIC):

```
e2a fb e4       ä ebcdic
e2a bb c4       Ä ebcdic
e2a 4f f6       ö ebcdic
e2a bc d6       Ö ebcdic
e2a fd fc       ü ebcdci
e2a bd dc       Ü ebcdic
e2a ff df       ß ebcdic
a2e e4 fb       ä ansi
a2e c4 bb       Ä ansi
a2e f6 4f       ö ansi
a2e d6 bc       Ö ansi
a2e fc fd       ü ansi
a2e dc bd       Ü ansi
a2e df ff       ß ani
```

Code bei Binärmode

Werden Daten binär gelesen, kann hier die Codepage für die Anzeige der eingestellt werden. Als Standard wird die Codepage EBCDIC eingestellt. Nach dem Einlesen einer Datei im Binärmodus wird intern das Kommando CODE *codepage* aufgerufen. Die Einstellung ist auch von Bedeutung, falls die Option "Binär" nicht im Profil ausgewählt ist, weil der Binärmodus auch mit dem Kommando READ (S. **226**) aktiviert werden kann.

Die Einstellung kann auch beim Filetransfer zwischen Windows und UNIX-Systemen von Bedeutung sein, weil die Codierung der Umlaute unterschiedlich sein kann.

Mit der Einstellung "Binär" und der Angabe der entsprechenden Codepage können so Dateien im Original-Code editiert werden. Das Umschalten auf einen anderen Code kann natürlich auch nach dem Einlesen mit dem Kommando CODE (S. **181**) erfolgen.

Datei-Prefix/Bibliothek

prefix | *lib*([*typ* /] [*elem*])

prefix Prefix für den Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID im BS2000 bzw. Unix-Pfad. Dieser Prefix wird dem Dateinamen im READ bzw. WRITE-Kommando vorangestellt.

lib(*typ* / *elem*) Name einer PLAM-Bibliothek, wahlweise einschl. Elementtyp und teilqualifiziertem Elementnamen. Bei nachfolgenden Kommandos READ bzw. WRITE ist nur noch der Elementname bzw. der Suffix zum Elementnamen anzugeben.

lib Name der PLAM-Bibliothek auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID.

typ Typ des PLAM-Elements. Es sind nur die Angaben S, M, J, P und D zulässig. Wird kein Typ angegeben, so wird der Typ "S" als Standard benutzt.

elem Prefix zum Namen des PLAM-Elements.

Hinweis:

Beginnt der Dateiname im READ bzw. WRITE-Kommando mit einer CAT-ID (:*xxx*) oder einer User-ID ("$"), wird der Parameter *prefix* | *plam-lib* ignoriert. Beginnt der Dateiname im READ bzw. WRITE-Kommando mit einem Laufwerksbuchstaben oder dem Zeichen "\", wird das Profil ignoriert und eine lokale Datei gelesen bzw. geschrieben.

Ist ein Kommando CHDIR (S. **179**) mit FT-Angaben aktiv, werden die Angaben in diesem Feld ignoriert.

Modus

Text

Die Daten werden mit der Standard-Translatetabelle bzw. der Translatetabelle, die im FT-Profil angegeben ist, übersetzt (ANSI <--> EBCDIC/ASCII). und satzstrukturiert eingelesen. Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden.

Profil-Typ **openFT_BS2**:

Beim Schreiben ohne weitere Formatangabe wird die ferne Datei allerdings mit dem Format "Variable Satzlänge" erzeugt. Beim Kommando `READ` (S. **226**) und `WRITE` (S. **264**) kann zusätzlich die Option "M=TF*nnn*" zum Lesen und Schreiben von Dateien mit fester Satzlänge angegeben werden.

Profil-Typ **FTP_UNIX, FTP_POSIX und FTP_BS2000**:

Dateien, die im BS2000 mit fester Satzlänge gespeichert sind, werden beim Schreiben in die gleiche Datei automatisch im richtigen Format erzeugt. Neue Dateien können nur mit variabler Satzlänge erzeugt werden.

Binär

Die Daten werden transparent ohne Übersetzung übertragen.

Profil-Typ **openFT_BS2**:

Die Daten werden satzstrukturiert im Originalcode dargestellt. In der lokalen Datei (Name wird entweder vom EDT vergeben oder Name aus "L=*file*") werden die Daten mit 2 Byte Satzlängenfeld ohne Satzende-Kennzeichen gespeichert. Nach dem Einlesen der Datei in den Arbeitsbereich wird die Codierung automatisch auf EBCDIC umgeschaltet (wie Kommando `CODE EBCDIC` (S. **181**)). Dadurch ist es möglich, dass in einem Satz beliebige nicht abdruckbare Zeichen, auch das BS2000-Satzende-Kennzeichen (X'15'), vorkommen können. Beim Kommando `READ` (S. **226**) und `WRITE` (S. **264**) kann zusätzlich die Option "M=TF*nnn*" zum Lesen und Schreiben von Dateien mit fester Satzlänge und die Option "M=U" für Dateien ohne Satzstruktur angegeben werden.

Profil-Typ **FTP-UNIX, FTP-POSIX und FTP-BS2000**:

Die Daten werden transparent ohne Übersetzung übertragen und im Arbeitsbereich ohne Satzstruktur dargestellt. Die Daten enthalten in diesem Fall folgende Satzende-Kennzeichen: Windows = X'0D0A', Unix = X'0A' und BS2000 = X'15'. Die Codierung kann im Profil schon auf den Wert ANSI, ASCIIUNIX oder EBCDIC7 eingestellt werden.

Nach dem Einlesen kann die Satzstruktur des fernen Rechners mit dem Kommando `REFORMAT` (S. **234**) wieder hergestellt werden.

Beispiel:

```
REFORMAT BS2    (BS2000, variable Satzlänge)
REFORMAT RS80   (BS2000, feste Satzlänge 80)
REFORMAT UNIX   (Unix-Datei)
REFORMAT DOS    (Windows-Datei)
```

Vor dem binären Schreiben eines solchen Arbeitsbereichs muss mit dem Kommando `UNFORMAT` wieder das Binärformat erzeugt werden.

Wird der Arbeitsbereich mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, werden die Daten automatisch wieder im Binär-Modus transferiert und die Datei auf dem fernen Rechner mit den bestehenden Dateiattributen erstellt.

Folgeverarbeitung im fernen und lokalen System

Remote-Success / Remote-Failure (gilt nur für openFT)

Folgeverarbeitung im fernen System im Falle der erfolgreichen (Remote-Success) bzw. fehlerhaften (Remote-Failure) Übertragung.

Hier können Kommandos in der Syntax des fernen Systems angegeben werden (z.B.: `/EXEC PROG` oder `/CALL PROC`). In Prozeduren für das BS2000 können auch mehrere Kommandos, durch Semikolon getrennt, angegeben werden (z.B. /do cmd1;/do cmd2;/do cmd3).

Die Prozeduraufrufe können mit symbolischen Parametern (Groß-/Kleinschreibung beliebig) versehen werden, die dann durch den aktuellen Wert ersetzt werden. Beschreibung siehe Abschnitt Symbolische Parameter (S. 148).

Local-Success / Local Failure

Folgeverarbeitung im lokalen System im Falle der erfolgreichen (Local-Success) bzw. fehlerhaften (Local-Failure) Übertragung.

Hier können entweder Kommandos in der Syntax des lokalen Systems (z.B. `sucess.bat` oder `success.exe`) oder EDT-Kommandos angegeben werden. EDT-Kommandos müssen mit dem Zeichen "@" beginnen. Es ist auch erlaubt, das Kommando `@INPUT` anzugeben und damit eine Input-Prozedur zu starten.

Die Prozeduraufrufe können mit symbolischen Parametern (Groß-/Kleinschreibung beliebig) versehen werden, die dann durch den aktuellen Wert ersetzt werden. Beschreibung siehe Abschnitt Symbolische Parameter (S. 148).

Symbolische Parameter für die Folgeverarbeitung

Die symbolischen Parameter können beliebig in Klein- oder Großbuchstaben angegeben werden. Die folgende Schreibweise dient nur der bessern Lesbarkeit.

Aufrufparameter

&Call	EDT-Kommando, über das der FT gestartet wurde (`W=Write, R=READ, F=FSTAT`)
&Mode	FT-Modus: `T=Text, B=Binär`
&Lpass	Logon-Passwort aus dem Parameter `LP` des Kommandos `READ`, `WRITE` oder `FILE` bzw. aus dem Profil.
&Fpass	Datei-Passwort für die entfernte Datei aus dem Parameter `FP` des Kommandos `READ`, `WRITE` oder `FILE` bzw. aus dem Profil.
&1 - &9	Parameter 1 bis 9, die wahlweise beim Kommandos READ und WRITE nach dem Dateinamen angegeben werden können (z.B. `WRITE'std r=file1 par1 par2').`

Name der eingelesenen Datei

In diesen Variablen enthalten den Namen der eingelesenen Datei bzw. die Teile des Namens und werden für den Fall benötigt, dass eine Datei unter einem anderen Namen geschrieben wird und der ursprüngliche Dateiname aber für eine Folgeverarbeitung noch benötigt wird. Dabei kann es sich um eine lokale oder entfernte Datei handeln.

&ReadFile	Vollständiger lokaler Dateiname einschl. Laufwerk und Pfadname oder vollständiger entfernter Dateiname `UserID.filename` bzw. `UserID.Lib(typ/elem/vers)`
&ReadLib	Bibliotheksname oder leer

&ReadType	Elementtype des Bibliothekelements oder leer
&ReadEle	Name des Bibliothekelements oder leer
&ReadVers	Version des Bibliothekelements oder leer

Entfernter Dateiname

&Remfile	Entfernter Dateiname, der beim Parameter `r=` des Kommandos `READ` und `WRITE` angegeben wird (z.B. `WRITE std r=file1`).
&RCatID	CAT-ID der BS2000-Datei
&RUserID	User-ID der BS2000-Datei
&RFile	Dateiname der BS2000-Datei bzw. Bibliotheksname
&RName	Dateiname der Elementname einer Bibliothek
&LibName	Bibliotheksname
&EleType	Elementtype des Bibliothekelements
&EleName	Name des Bibliothekelements
&EleVers	Version des Bibliothekelements
&EleDate	Datum des Bibliothekelements

Lokaler Dateiname

&Locfile	Lokaler Dateiname. Dabei handelt es sich entweder um einen temporären Dateinamen, der von EDT automatisch erzeugt wird oder um den Namen, der beim Parameter `l=` des Kommandos `READ` und `WRITE` angegeben wird (z.B. `READ std l=file1 r=bs2file1`).
&LName	Dateiname einschl. Extension der lokalen Datei
&LDrive	Laufwerksbuchstabe bzw. UNC-Pfad der lokalen Datei
&LPath	Pfadname der lokalen Datei
&LFile	Dateiname der lokalen Datei ohne Extension
&LExt	Extension der lokalen Datei

Werte aus dem FT-Profil:

&Host	entfernter Hostname
&userid	User-ID
&Account	Account-Nummer
&FTac	Transfer-Admission
&FTName	Name des fernen FT-Systems ($FJAM)
&FTPartner	Name des lokalen FT-Systems (PCSYSTEM
&FTProfil	Name des EDT-FT-Profils

Sonstige Variablen:

&ownip	IP-Adresse des eigenen Rechners

Syntax-Highlighting

In diesem Dialogfenster können die Optionen für das Syntax-Highlighting eingestellt werden.

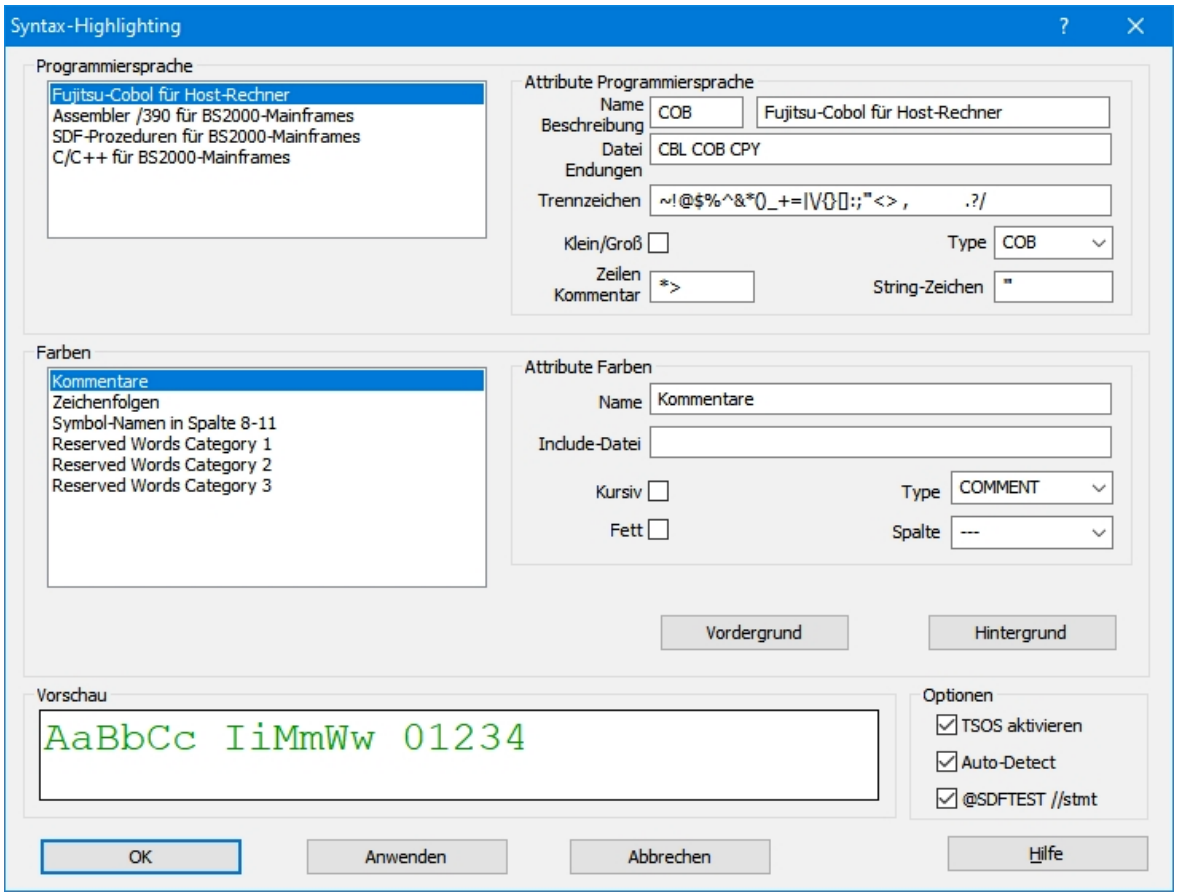
Das Syntax-Highlighting kann entweder mit dem Kommando `shl` (S. 247) oder mit der Schaltfläche in der Ansicht-Toolbar (S. 164) oder mit dem Menübefehl `Ansicht/Syntax-Highlighting` (S. 96) aktiviert oder deaktiviert werden.

Wenn die Option `Auto-Dedect` eingeschaltet ist, wird nach dem Lesen einer Datei automatisch geprüft, ob die Daten Merkmale einer der unterstützten Programmiersprachen enthalten und ggf. die entsprechende Programmiersprache aktiviert.

Zur Zeit werden Syntax-Dateien für folgende Programmiersprachen ausgeliefert:

- Cobol für BS2000 Mainframes
- Assembler /390 für BS2000 Mainframes (Source-Datei und Compiler-Liste)
- SDF-Prozeduren für BS2000 Mainframes
- C/C++ für BS2000 Mainframes

Eine Erweiterung mit neuen benutzerdefinierten Sprachen ist zur Zeit nicht möglich.



Auswahl Programmiersprache

Programmiersprache

In der Listbox werden die unterstützten Programmiersprachen angezeigt:

- Cobol für BS2000 Mainframes
- Assembler /390 für BS2000 Mainframes
- SDF-Prozeduren für BS2000 Mainframes
- C/C++ für BS2000 Mainframes

Wird eine Sprache ausgewählt, so werden die dazugehörigen Attribute rechts neben der Listbox und die dazugehörigen Parameter für die Farbeinstellungen in dem Bereich unter der Listbox angezeigt.

Attribute Programmiersprache

Die Einstellungen für Type, Zeilenkommentar und String-Zeichen sollten nicht geändert werden, da sonst die Erkennung der einzelnen Elemente der Programmiersprache nicht mehr funktioniert.

Name Kurzbezeichnung der Programmiersprache für die Auswahl im Menü und mit dem Kommando SHL.

Beschreibung Beschreibung der Programmiersprache als Erläuterung.

Datei-Endung Datei-Endungen für die automatische Erkennung. Wird eine Datei mit der hier angegebenen Endung gelesen, so wird automatisch die entsprechende Programmiersprache aktiviert, wenn die automatische Erkennung eingeschaltet ist.

Trennzeichen Die Liste der angegebenen Zeichen dient zur Trennung der Wörter bei der Prüfung, welche Farbe für die entsprechenden Daten angezeigt werden soll.

Klein/Groß Beim Vergleichen der Daten mit den Definitionen der Syntaxdateien die Beachtung der Klein/Großschreibung ein/ausschalten

☑ Beim Vergleich wird die Klein/Großschreibung beachtet

☐ Beim Vergleich wird die Klein/Großschreibung nicht beachtet

Type Typ der Programmiersprache. Die Auswahl eines bestimmten Typs bewirkt, daß neben den Definitionen in den Syntaxdateien zusätzliche Funktionen aktiviert werden, die das Syntax-Highlighting steuern. Der Typ für die ausgelieferten Standardsprachen darf nicht geändert werden, da sonst die Erkennung der einzelnen Elemente der Programmiersprache nicht mehr funktioniert.

Zeilenkommentare

Definition der Zeichen, die einen Zeilenkommentar einleiten.

String-Zeichen Zeichen, das ein Zeichenfolge begrenzt, in der Regel das Zeichen Anführungszeichen oder Hochkomma.

Auswahl Farbe

Farbauswahl In der Listbox werden die definierten Farben angezeigt:

Wird eine Farbe ausgewählt, so werden die dazugehörigen Attribute rechts neben der Listbox angezeigt.

Eine Erweiterung mit neuen benutzerdefinierte Farben ist zur Zeit nicht möglich.

Attribute Farben

Die Einstellungen für Type und Spalte sollten nicht geändert werden, da sonst die Erkennung der einzelnen Elemente der Programmiersprache nicht mehr funktioniert.

Name Beschreibung der Farbe.

Include-Datei Dateiname mit den Definitionen der Elemente, die mit dieser Farbe angezeigt werden sollen. Enthält der Dateiname keinen Pfadnamen oder einem relativen Pfadnamen, wird die Datei zuerst auf dem Benutzer-Verzeichnis %APPDATA% und falls nicht vorhanden auf dem Installationsverzeichnis gesucht.

Kursiv Attribut Kursiv für die Schrift ein/ausschalten.

Fett Attribut Fett für die Schrift ein/ausschalten.

Spalte Falls die definierten Elemente für diese Farbe nur in einer bestimmten Spalte gesucht werden sollen, ist hier die Spalte anzugeben, z.B. Assembler-Befehle stehen immer in der Spalte 2.

Type In Abhängigkeit des Farb-Typs werden zusätzliche Funktionen zur Erkennung der definierten Elemente in der Include-Datei aktiviert:

a) für alle Programmiersprachen:

COMMENT	Kommentare
STRING	Zeichenfolgen

b) nur für Cobol

PARAGRAPH	Paragraph

c) nur für SDF-Prozeduren

CMDPAR	Parameter zu SDF-Kommandos
CMDVAL	Werte zu SDF-Kommandos
FCTPAR	Parameter zu SDF-Funktionen
FCTVAL	Werte zu SDF-Funktionen
COMMAND	SDF-Kommandos
FUNCTION	SDF-Funktionen

Vordergrund Aufruf des Dialogs zur Auswahl der Schriftfarbe.

Hintergrund Aufruf des Dialogs zur Auswahl der Farbe für den Hintergrund.

Vorschau

Vorschau In diesem Bereich wird die Schrift mit allen Attributen (Vordergrund, Hintergrund, kursiv, fett) angezeigt:

Optionen

TSOS aktivieren

Soweit die Sprache SDF aktiviert ist, werden zusätzlich alle Kommandos verarbeitet, die nur unter TSOS zulässig sind.

Auto-Detect Ein/Ausschalten der automatischen Erkennung der Programmiersprache. Bei Aktivierung werden nach dem Einlesen die Sätze nach bestimmten Merkmalen untersucht und bei Übereinstimmungen die entsprechende Sprache aktiviert.

Die automatische Aktivierung in Abhängigkeit der Datei-Endung wird davon unabhängig immer durchgeführt.

@SDFTEST //stmt

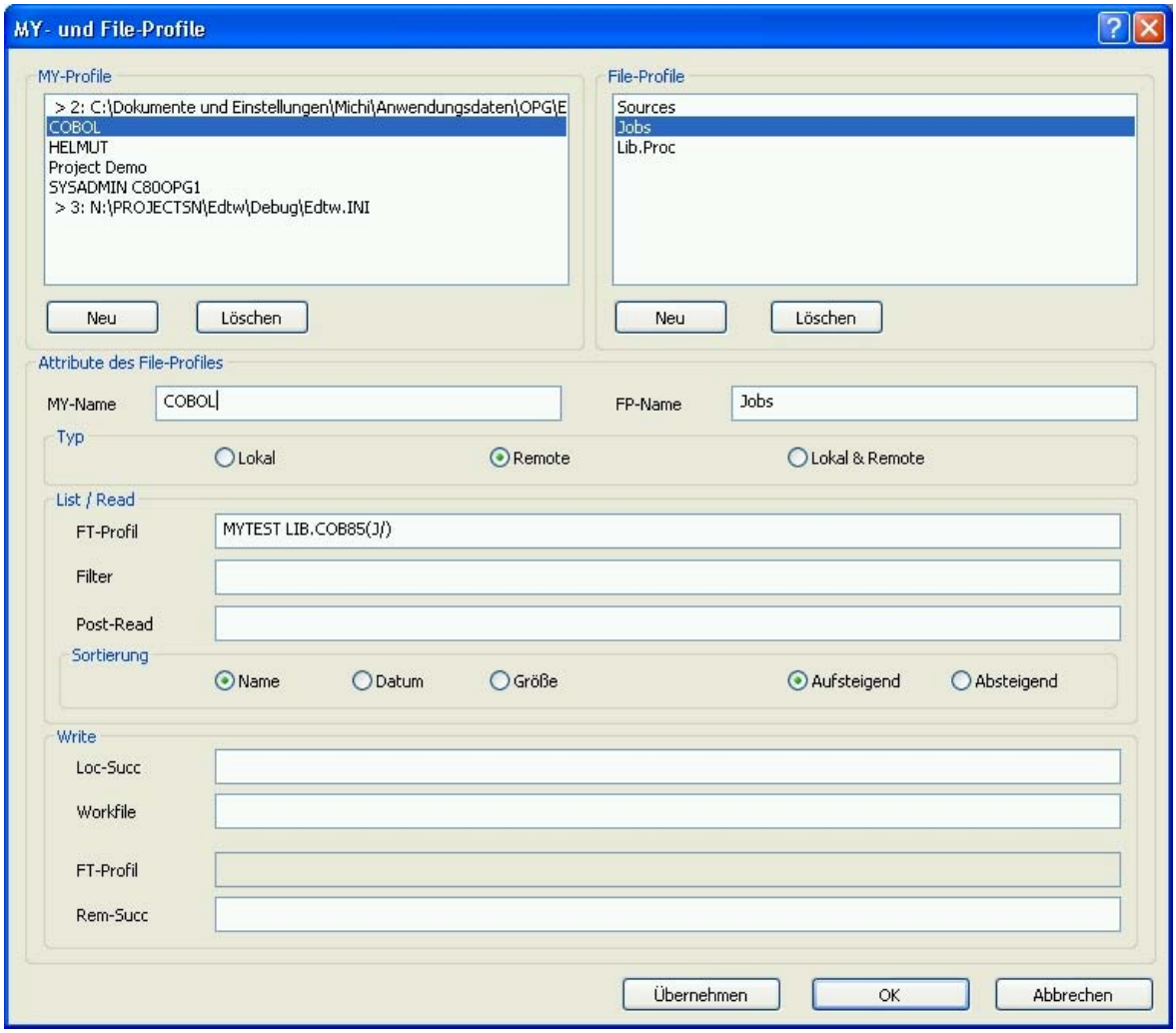
Ein/Ausschalten der Online-Syntaxprüfung der //-Statements für bestimmte Fujitsu Standard-Programme (BINDER, BS2ZIP, HSMS, ...). Software-Voraussetzung: ab OSD V11.0 mit SP19.1.

MY-Profiles

In dieser Dialogbox kann einer individuelle Arbeitsumgebung definiert werden. Diese Arbeitsumgebung wird im linken Navigationsframe, Registerkarte MY-Profile angezeigt.

Die Arbeitsumgebung besteht aus beliebig vielen Projekten. In einem Projekt wird beschrieben, welche Dateien benötigt werden und wie diese Dateien verarbeitet werden sollen.

Da die Dateien aus unterschiedlichsten Quellen (lokale Dateien, entfernte Dateien auf verschiedenen Rechnern) stammen können, wird das Projekt wiederum in File-Profile unterteilt. In einem File-Profil wird definiert, welche Dateien aus einer Quelle benötigt werden. Zusätzlich wird hier angegeben, welche Aktionen nach dem Lesen und nach dem Schreiben ausgeführt werden sollen.



My-Profile In diesem Bereich werden die vorhandenen Projekt (My-Profile) angezeigt. Die My-Profile können entweder in der lokalen edtw.ini (>2: C:\Dokumente....) oder in der INI-Datei auf dem Ladeverzeichnis (>3: .....) gespeichert werden.

Mit dem Button **Neu** kann ein neues My-Profile angelegt werden. Mit dem Button **Löschen** kann ein Profil gelöscht werden.

File-Profile In diesem Bereich werden die vorhandenen Profile angezeigt, die einem bestimmten My-Profile zugeordnet sind.

Mit dem Button **Neu** kann ein neues File-Profil angelegt werden. Mit dem Button **Löschen** kann ein File-Profil gelöscht werden.

Attribute des File-Profiles

MY-Name Name des MY-Profiles.

FP-Name Name des File-Profiles.

Typ Typ des Profils.

Lokal: Es handelt sich um Dateien, die sich auf dem lokalen PC befinden und die auch nach der Bearbeitung wieder auf den lokalen PC zurückgeschrieben werden sollen.

Remote Es handelt sich um Dateien, die sich auf einem entfernten System befinden und die auch nach der Bearbeitung wieder auf das entfernte System zurückgeschrieben werden sollen.

Lokal &Remote

Es handelt sich um Dateien, die sich auf dem lokalen PC befinden und die auch nach der Bearbeitung auf einem entfernten System zurückgeschrieben werden sollen.

List / Read

In diesem Bereich wird die Quelle der Dateien (Pfad bzw. Filetransfer-Profil) und die Aktion nach dem Einlesen einer Datei beschrieben.

Path / FT-Name

Path Pfadname mit Dateiauswahl bei lokalen Dateien (Typ Lokal).

Nach dem Pfadnamen folgt durch das Zeichen "\" getrennt, die Auswahlkriterien mittels Wildcard-Syntax.

Beispiele:

\\server\disk1\testpath*.cob
c:\testpath*.src

oder (in Abhängigkeit des Fileprofil-Typs)

FT-Name Name eines FT-Profils (Typ Remote oder lokal & Remote).

Nach dem Namen kann ein Prefix angegeben werden in der Syntax des Feldes Prefix im Filetransfer-Profil (S. 146). Als Prefix kann entweder ein Datei-Prefix oder eine BS2000-Bibliothek mit ggf. Angabe eines Element-Typs sein. Diese Auswahl bewirkt, dass vom openFT oder FTP nur die Dateien angezeigt werden bzw. auch nur die Attribute der Dateien übertragen werden müssen, die diesen Auswahlkriterien entsprechen.

Ein evtl. vorhandener Prefix im Filetransfer-Profil wird ignoriert bzw. von dem hier angegebenem Prefix überschrieben.

Das Filetransfer-Profil muss sich in der gleichen INI-Datei befinden wie das My-Profile.

Zusätzlich können im Feld `Filter` weitere Auswahlkriterien angegeben werden, die jedoch erst nach dem Übertragen aller Datei-Attribute wirksam werden.

Beispiele:

`fttest cobol.lib(s/)`	alle Elemente vom Typ S aus der Library cobol.lib
`fttest cobol.lib()`	alle Elemente aus der Library cobol.lib
`src.`	Alle BS2000-Dateien, die mit src. beginnen

Filter zusätzlicher Filter nach der Suchsyntax des EDTW (S. 346)

searchstr { *such* [*vk such .....*] | (*s-dat*) }

Komfortable Auswahlmöglichkeit von Dateien mit mehreren Suchbegriffen und Angabe einer Datei mit Suchbegriffen nach der Syntax des EDTW und CFS. Ausführliche Beschreibung siehe S. 346.

Insgesamt gibt es drei Möglichkeiten, die gewünschten Dateien auszuwählen:

a) Prefix im Feld FT-Name: Aus Performance-Gründen sollten die Auswahlmöglichkeiten im Feld `Path` bzw. die `FT-Name` genutzt werden, da dieser Filter schon von Windows, bzw. von openFT und FTP für die Vorauswahl der Dateien verwendet wird. Dieser Filter wird dann zusätzlich auf die verbleibenden Dateien angewandt.

b) Dieser Filter wird sofort nach dem Öffnen eines File-Profils angewendet und ermöglicht eine weitere Verfeinerung der Auswahl.

c) Filter im Navigationsframe mittels des Menüs der rechten Maustaste. Dieser Filter wirkt nur temporär und kann auch wieder aufgehoben werden.

Beispiele:

`'.cob'`	
`-'.cbl'+-'cpb'`	alle Elemente einer BS2000-Bibliothek außer den Elementen mit dem Suffix cbl und cpb
`'test'+'.cob'`	

Post-Read EDT-Kommandos, die nach dem Einlesen der Datei ausgeführt werden sollen.

Mehrere EDT-Kommandos werden durch das Zeichen `";"` getrennt

Beispiele:

```
edit indent on;lower off
i'edtproc.edt'
```

Sortierreihenfolge der Dateien

Die Dateien werden nach dem Öffnen des Profils nach diesen Sortierkriterien sortiert. Zusätzlich ist es auch möglich, über das Menü der rechten Maustaste im Navigationsframe temporär eine andere Sortierreihenfolge festzulegen.

Name Sortierung nach dem Dateinamen.

Datum Sortierung nach dem Änderungsdatum der Dateien.

Größe Sortierung nach der Größe der Dateien.

Aufsteigend Die Dateien werden aufsteigend sortiert.

Absteigend Die Dateien werden absteigend sortiert.

Write

In diesem Bereich werden die Aktion nach dem Schreiben Einlesen einer Datei beschrieben.

Loc-Succ DOS-Kommando, das nach dem erfolgreichen Schreiben einer Datei ausgeführt wird.

Das Kommando kann mit symbolischen Parametern (Groß-/Kleinschreibung beliebig) versehen werden, die dann durch den aktuellen Wert ersetzt werden. Beschreibung siehe Abschnitt Symbolische Parameter (S. 148).

Hinweis:

Falls Dateinamen Leerstellen enthalten können, sollten diese Parameter mit Anführungszeichen übergeben werden. Soweit eine DOS-Batchdatei verwendet wird, müssen diese Parameter mit dem Zeichen ~ verwendet werden, z.B. der erste Parameter muss mit `%~1` bezeichnet werden. Dadurch werden die umschließenden Leerzeichen gelöscht.

Beispiele:

```
test.bat "&lpath" "&lfile" &lext
```

Workfile Dateinamen auf dem entfernten System bei den File-Profilen der Typen "Remote" und "Local & Remote". Falls dieses Feld leer ist, wird der Name der Eingabedatei verwendet.

Der Dateiname kann symbolischen Parametern (Groß-/Kleinschreibung beliebig) enthalten, die dann durch den aktuellen Wert ersetzt werden. Beschreibung siehe Abschnitt Symbolische Parameter (S. 148).

Beispiele:

```
pre.&Rname
&Libname(&Rname)
```

FT-Profil Name eines FT-Profils, falls es sich um ein File-Profile des Typs "Local & Remote" handelt.

Rem-Succ Folgeverarbeitung im fernen System im Falle der erfolgreichen Übertragung.

Hier können Kommandos in der Syntax des fernen Systems angegeben werden (z.B.: `/EXEC PROG` oder `/CALL PROC`). In Prozeduren für das BS2000 können auch mehrere Kommandos, durch Semikolon getrennt, angegeben werden (z.B. /do cmd1;/do cmd2;/do cmd3).

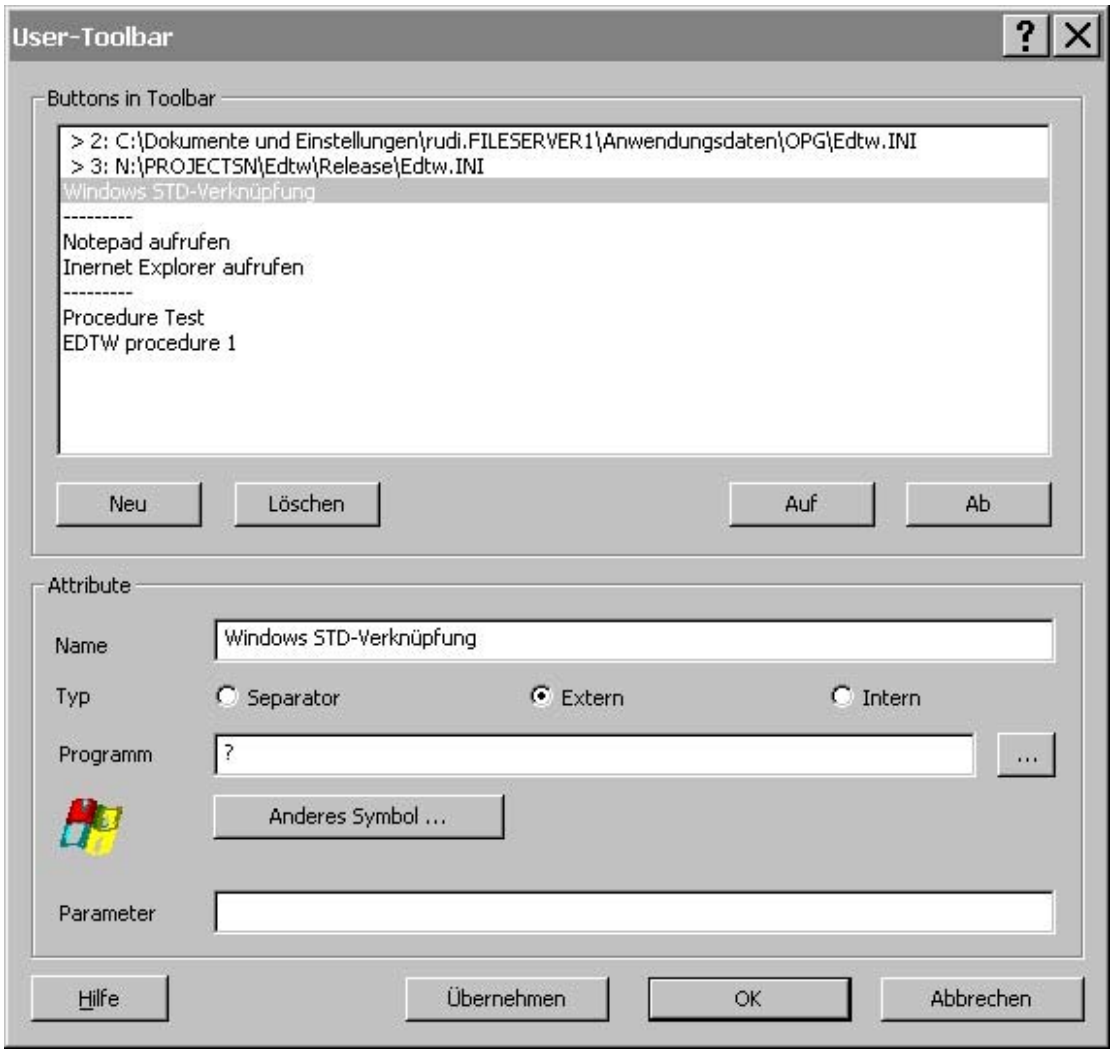
Das Kommando kann mit symbolischen Parametern (Groß-/Kleinschreibung beliebig) versehen werden, die dann durch den aktuellen Wert ersetzt werden. Beschreibung siehe Abschnitt Symbolische Parameter (S. 148).

Beispiele:

```
/clp lib.proc(vartest),('&locfile','&remfile')
```

User-Toolbar

In dieser Dialogbox können individuelle Schaltflächen definiert werden. Damit kann für den Inhalt des aktuellen Arbeitsbereichs eine interne EDTW Anweisung (z..B. @INPUT 'prozedur') abgesetzt werden oder ein externes Programm zur Weiterverarbeitung (z.B. Browser oder anderer Editor) gestartet werden.



Buttonlist Die Namen der definierten User-Buttons werden unter der entsprechenden INI-File angezeigt.

Neu/Löschen Über die Schaltflächen können neue User-Buttons aufgenommen bzw. gelöscht werden.

Auf/Ab Über die Schaltflächen kann die Reihenfolge der vorhandenen User-Buttons verändert werden. Ein Verschieben ist allerdings nur innerhalb derselben INI-File möglich.

User-Toolbar Button Attribute

Name Name eines User-Buttons. Dieser wird in der Buttonliste angezeigt und dient automatisch auch als Tooltip für das entsprechende Icon in der User-Toolbar

Typ Typ des User-Button.

Separator Erzeugt in der User-Toolbar nur einen senkrechten Trennstrich, um bestimmte Verarbeitungen optisch voneinander zu trennen.

Extern Aufruf eines "externen" Programms, wobei standardmäßig veränderte Dateninhalte vorher automatisch gesichert werden und anschließend an das Programm der vom EDTW verwendete lokale Dateiname übergeben wird. Falls im Eingabefeld Parameter Angaben gemacht werden, so muss selbst dafür gesorgt werden, dass der lokale Dateiname über Platzhalter ("!file") an der richtigen Stelle an das Programm übergeben wird.

Intern Absetzen einer belibigen EDTW-Anweisung in der aktuellen Arbeitsebene. In der Regel wird es sich dabei um ein @INPUT Statement handeln, womit eine umfangreiche EDT-Prozedur zum Ablauf gebracht werden kann.

Programm Typ Extern: Name des externen Windows-Programms. Das Programm kann auch über die danebenliegende Schaltfläche im Windows-Explorer-Dialog ausgewählt werden.

Sonderfall:
?
Ein Fragezeichen als Programm-Name bewirkt, dass die lokale Datei automatisch mit dem in der Windows-Verknüpfung (Datei-Extension z.B. .HTML) hinterlegten Programm geöffnet wird.

Statement Typ Intern: interne EDTW-Anweisung. Das Kommando kann mit symbolischen Parametern (Groß-/Kleinschreibung beliebig) versehen werden, die dann durch den aktuellen Wert ersetzt werden. Beschreibung siehe Abschnitt Symbolische Parameter (S. 344).

Symbol Icon, dass in der User-Toolbar angezeigt werden soll. Standardmäßig wird das Erste in einer EXE-File vorhandene Icon angezeigt. Über die danebenliegende Schaltfläche kann ein beliebiges anderes Icon ausgewählt werden.

Parameter Typ Extern: für externe Programm kann es notwendig sein, dass neben dem lokalen Dateiname, noch zusätzliche Parameter übergeben werden müssen. Falls hier Eintragungen vorhanden sind, werden diese, nach einer eventuellen Ersetzung von symbolischen Parametern (Groß-/Kleinschreibung beliebig) als Aufrufparameter an das ausgewählte Programm übergeben. Beschreibung siehe Abschnitt Symbolische Parameter (S. 344).

Sonderfälle:

***EXE**
Die aktuelle Arbeitsebene wird nicht gespeichert, es wird kein Dateiname übergeben, sondern nur das unter Programm angegebene externe Programm gestartet.

EXE *param
Die aktuelle Arbeitsebene wird nicht gespeichert und das externe Programm wird mit den unter *param* angegebenen Aufrufparametern gestartet.

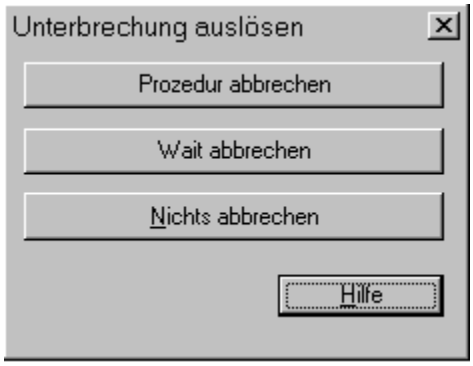
Beispiele:

```
Name: Windows Standard Verknüpfung
Typ: Extern
Programm: ?
```

```
Name: SDF Syntaxcheck
Typ: Intern
Statement: @SDFTEST
Icon: sdftest.ico

Name: TextPad
Typ: Extern
Programm: C:\Program Files (x86)\TextPad 7\TextPad.exe
Parameter: -u !file
```

Prozedur oder WAIT abbrechen



Falls gerade eine EDT-Prozedur, ein READ-Kommando oder ein WAIT-Kommando (S. 261) ausgeführt wird, kann durch Anklicken des entsprechenden Buttons eine Unterbrechung erreicht werden.

Shortcut: 

Menü Fenster

Neues Fenster

Es wird ein neues Fenster mit dem gleichen Inhalt wie das aktuelle Fenster geöffnet. Sie können diese Funktion benutzen, um eine Datei in verschiedenen Formaten (z. B. Hexadezimal- oder Character-Darstellung, Edit und Edit Long) oder verschiedene Teile einer Datei zu editieren. Falls Sie die Datei in einem Fenster ändern, werden die Änderungen auch in allen anderen Fenstern angezeigt, die dieselbe Datei enthalten. Das neue Fenster wird zum aktiven Fenster und im Vordergrund angezeigt.

Überlappend Alle geöffneten Fenster werden überlappend angeordnet.

Nebeneinander

Ordnet alle Fenster nebeneinander ohne Überlappung an. Falls der Platz nicht mehr ausreicht, werden die Fenster neben- und untereinander angeordnet.

Untereinander Ordnet alle Fenster untereinander ohne Überlappung an. Falls der Platz nicht mehr ausreicht, werden die Fenster neben- und untereinander angeordnet.

Icons anordnen

Die ikonisierten Fenster im unteren Bereich des Hauptfensters werden neu angeordnet. Falls sich im unteren Bereich des Hauptfensters ein aktives Fenster befindet, können Icons durch dieses Fenster überlagert sein.

Nächster Arbeitsbereich

Der nächste Arbeitsbereich wird aktiviert. Der Arbeitsbereichswechsel kann auch durch die Eingabe der Arbeitsbereichs-Nummer in der Kommandozeile erreicht werden.

Shortcuts:

Symbol: 

Tasten: `<Strg>` + `<Tab Right>`

Vorheriger Arbeitsbereich

Der nächste Arbeitsbereich wird aktiviert. Der Arbeitsbereichswechsel kann auch durch die Eingabe der Arbeitsbereichs-Nummer in der Kommandozeile erreicht werden.

Shortcuts:

Symbol: 

Tasten: `<Strg>` + `<Tab Left>`

Nächste View Die nächste View des Arbeitsbereichs wird aktiviert. Der View-Wechsel kann auch durch das Kommando `VIEW` (S. 260) erreicht werden.

Vorherige View

Vorherige View

Die vorherige View des Arbeitsbereichs wird aktiviert. Der View-Wechsel kann auch durch das Kommando `VIEW` (S. **260**) erreicht werden.

1, 2, ... Im unteren Bereich des Menüs werden alle geöffneten Fenster aufgelistet. Als Name wird der Inhalt der Titelzeile verwendet. Wählen Sie ein Fenster aus, um es zu aktivieren.

Menü Hilfe

Inhalt Das Inhaltsverzeichnis der Online-Hilfe wird angezeigt. Von hier aus können Sie alle Hilfe-Informationen zur Benutzung des EDT erreichen.

Sie können im HELP-System jederzeit mittels des Buttons Inhalt zum Inhaltsverzeichnis zurückkehren.

Hilfe benutzen

Es werden Informationen zur Benutzung des HELP-Systems ausgegeben.

Benutzungshinweise

Es werden Benutzungshinweise und die Lizenzdaten angezeigt.

Über EDT Es werden die Copyright-Informationen und die Versionsnummer des EDT angezeigt.

5 Schaltflächenzeile

Haupt-Toolbar

Die Schaltflächenzeile wird unterhalb der Menüzeile angezeigt und erlaubt Ihnen einen schnellen Zugriff mittels Mausklick zu vielen EDT-Funktionen.

Die Schaltflächenzeile kann mit der Option Schaltflächen im Menü Ansicht ein- bzw. ausgeblendet werden.

Öffnen eines neuen leeren Fensters

Öffnen einer bestehenden Datei. EDT für Windows zeigt die Dialogbox Open an, in der Sie den Dateinamen und das Verzeichnis wählen können.

Sichern des aktiven Arbeitsbereichs.

Speichern aller veränderten und noch nicht gesicherten Arbeitsbereiche.

Drucken des aktiven Arbeitsbereichs.

Ausschneiden der markierten Daten und kopieren in die Zwischenablage.

Kopieren der markierten Daten in die Zwischenablage.

Einfügen der Daten aus der Zwischenablage.

Letzte Aktion rückgängig machen. (UNDO)

Letzte UNDO-Aktion rückgängig machen. (REDO)

Dialogbox zum Suchen einer Zeichenfolge anzeigen.

Zum nächsten Treffer positionieren.

Zum vorherigen Treffer positionieren.

Markierung der Treffer löschen.

Nächstes Datenfenster aktivieren.

Vorhergehendes Datenfenster aktivieren.

Einträge des Kommandogedächtnisses anzeigen.

Strings für die Programmierbaren Tasten anzeigen.

Dialogbox mit dem Inhaltsverzeichnis aller Dateien eines REWRITE-Arbeitsbereichs anzeigen.

EDT-Prozedur oder Kommando WAIT unterbrechen.

Kontextsensitive Hilfe.

Ansicht-Toolbar

Hexa-Modus ein- / ausschalten.

Low-Modus ein- / ausschalten.

EDIT LONG-Modus ein- / ausschalten.

EDIT FULL-Modus ein- / ausschalten.

EDIT WORD-Modus ein- / ausschalten.

Scalezeile (Zeilenlineal) ein- / ausblenden.

Zeilennumerierung umschalten - aus, 6-stellig, 8-stellig.

Syntax-Highlighting ein/ausschalten (S. 149).

Karteikarten ein- / ausblenden (S. 96).

Liste der Dateien / Filetransfer-Profile bzw. Arbeitsbereiche ein- / ausblenden (S. 96).

Protokollfenster ein- / ausblenden (S. 96).

6 Kontextsensitive Hilfe

Benützen Sie dieses Symbol, um ganz gezielt Hilfe zu ausgewählten Themen zu erhalten. Der Mauszeiger wechselt zu einem Pfeil mit Fragezeichen. Sobald Sie an irgendeiner Stelle im EDT-Fenster mit der linken Maustaste klicken, werden zu diesem Thema die Hilfe-Informationen angezeigt. Sie können auf die Menüzeile, auf Icons, auf Buttons oder sonstige Felder (überschreibbare oder nicht überschreibbar) klicken.

Shortcut
Tasten: SHIFT+F1

Zu jedem Zeitpunkt kann durch Betätigung der Taste `F1` das Hilfe-System aufgerufen werden. Es werden dann Informationen zu dem Feld ausgegeben, in dem der Cursor bei Betätigung der F1-Taste stand. Je nach Situation haben Sie die Möglichkeit, detailliertere Informationen bzw. Hilfe zu artverwandten Themen anzufordern.

In bestimmten Feldern, wie z. B. in der Kommandozeile bestehen sehr viele Eingabemöglichkeiten. Dementsprechend umfangreich sind auch die Hilfe-Informationen. In der Regel wird bei solchen Feldern nach dem Aufruf des Hilfe-Systems ein Menü angeboten, aus dem dann wieder Informationen ausgewählt werden können. In diesen Fällen kann die gewünschte Information sofort erreicht werden, indem das Hilfe-System in der Form *key* `F1` aufgerufen wird. Als *key* wird das entsprechende Schlüsselwort, also der Name des Kommandos angegeben.

Kontext-sensitive Worte und Menüs

Im Hilfetext können kontext-sensitive Worte vorhanden sein. Diese Worte sind in einer anderen Farbe dargestellt. Die kontext-sensitiven Worte können auch in Form eines Menüs vorkommen, d.h. die erste Maske nach Aufruf des Hilfe-Systems enthält nur eine Liste von kontext-sensitiven Worten.

Durch Auswahl eines solchen Wortes mit der linken Maustaste werden weitere Informationen zu diesem Thema ausgegeben. Weitere Informationen zum WINDOWS-HELP-System erhalten Sie über den Menüpunkt `Hilfe benutzen` des Menüs `Hilfe`.

7 Hauptfenster und Datenfenster

Felder der Titelzeile

Die Titelzeile wird als erste Zeile im Fenster angezeigt.

Um die Position des Fensters im Bildschirm zu ändern, klicken Sie mit der Maus auf die Titelzeile und ziehen das ganze EDT-Fenster an die gewünschte Stelle.

Kontrollmenü Durch Doppelklick mit der Maus wird das Datenfenster geschlossen bzw. das Programm beendet.

Überschrift Die Überschrift der Titelzeile besteht aus drei Teilen:
Nummer des Arbeitsbereichs in eckigen Klammern.
Dateinamen.

Falls die Daten einer bestehenden Datei geändert bzw. in einen neuen Arbeitsbereich Daten geschrieben wurden, wird nach dem Dateinamen das Zeichen "*" angezeigt.

Schaltflächen der Titelzeile

Minimieren

 Das Datenfenster bzw. das Hauptfenster wird zum Icon verkleinert.

Umschalten auf Vollbild WINDOWS

Das Symbol wird in zwei verschiedenen Zuständen angezeigt:

 Ein Datenfenster wird von der maximalen Größe zur zuletzt eingestellten Fenstergröße verkleinert.

Das Hauptfenster wird vom Vollbild bis zur zuletzt eingestellten Fenstergröße verkleinert.

 Ein Datenfenster wird bis zur maximalen Größe innerhalb des Hauptfensters vergrößert. Eventuell bestehende weitere Datenfenster werden verdeckt.

Das Hauptfenster wird zum Vollbild vergrößert.

Fenster Schließen

Das Hauptfenster bzw. das Datenfenster wird geschlossen. Falls geänderte Daten noch nicht gesichert sind, wird eine Aufforderung zur Sicherung und ggf. die Dialogbox Speichern unter angezeigt.

Kontrollmenü

Das Kontrollmenü enthält folgende Optionen

Fenster wiederherstellen
Fenster verschieben
Größe ändern
Fenster minimieren
Fenster Maximierung
Fenster schließen
Nächstes Fenster aktivieren

Durch Doppelklick mit der Maus wird das Datenfenster geschlossen bzw. das Programm beendet.

Größe ändern Mit Hilfe der Richtungstasten kann die Größe des Fensters geändert werden.

Nachdem sich der Mauszeiger zum Symbol  geändert hat:

1. Betätigen Sie eine der Richtungstasten (links, rechts, unten, oben), um den Cursor an den Fensterrand zu bewegen.
2. Betätigen Sie eine Richtungstaste, um den Rand zu verschieben.
3. Betätigen Sie die Enter-Taste, wenn die gewünschte Größe erreicht ist.

Diese Funktion ist nicht verfügbar, falls das Fenster im Vollbild dargestellt wird.

Shortcut
Maus: Ziehen Sie mit der Maus die Ränder oder die Ecken auf die gewünschte Größe.

Verschieben Mit dieser Funktion kann das Fenster mit Hilfe der Richtungstasten verschoben werden.

Diese Funktion ist nicht verfügbar, falls das Fenster im Vollbild dargestellt wird.

Shortcut
Tasten: <Strg>+F7

Minimieren Das Fenster wird zum Icon verkleinert.

Shortcut

Maus: Klicken Sie auf das Icon  in der Titelzeile.
Tasten: <Alt>+F9

Maximieren Das Fenster wird auf die Größe des Bildschirms bzw. das Datenfenster auf die Größe des Hauptfensters vergrößert.

Shortcut

Maus: Klicken Sie auf das Icon  in der Titelzeile oder Doppelklicken Sie auf die Titelzeile.
Tasten: <Strg>+F10

Nächstes Das nächste Fenster wird aktiviert. Als nächstes Fenster gilt das Fenster, das nach dem aktuellen Fenster geöffnet wurde.

Shortcut
Tasten: <Strg>+F6

Vorhergehendes

Das vorhergehende Fenster wird aktiviert. Als vorhergehendes Fenster gilt das Fenster, das vor dem aktuellen Fenster geöffnet wurde.

Shortcut
Tasten: SHIFT+<Strg>+F6

Schließen Das aktuelle Fenster wird geschlossen.

Das Fenster kann ebenfalls durch einen Doppelklick auf das Kontrollmenü in der Titelzeile geschlossen werden.

Falls eine Datei in mehreren Fenstern eingelesen wurde, wird nur das aktuelle Fenster geschlossen. Falls Sie alle Fenster mit der aktuellen Datei schließen wollen, können Sie die Funktion Schließen im Menü Datei verwenden.

Shortcuts
Tasten: <Strg>+F4 schließt das aktuelle Datenfenster
<ALT>+F4 schließt das Hauptfenster mit allen Datenfenstern

Wiederherstellen

Das Fenster wird in der Position und in der Größe wie vor der Aktion "Minimieren" bzw. "Maximieren" dargestellt.

Zeilenlineal

Diese Zeile enthält eine Skala mit den Spaltennummern.

Statusfeld in der Kommandozeile

In diesem Feld werden die Zeilennummer der ersten Zeile im Fenster und die erste Spaltennummer angezeigt.

Zeilennummer

In der Zeilennummern-Spalte wird die Zeilennummer entweder 6-stelllig in der Form 1234.12 oder 8-stellig in der Form 1234.1234 angezeigt. Statt des Punktes kann auch das Zeichen "*", "u" oder "d" angezeigt werden (z.B. 1234*12, 1243u12 oder 1234d12). Diese Kennzeichnung weist auf Besonderheiten des Satzformats hin. Siehe hierzu die Ausführungen im Abschnitt Dateistruktur (S. **40**).

Die Art der Zeilennummern-Darstellung kann mit der Schaltfläche 

, im Menü `Ansicht/Zeilennummer` oder mit dem Kommando `INDEX` (S. **206**) eingestellt werden.

Im EDIT-LONG-Modus (Zeilenumbruch von Sätzen, die länger sind als eine Bildschirmzeile) wird bei den Folgezeilen statt der Zeilennummer die Spaltennummer angezeigt.

Scrollbars

Die Scrollbars werden wahlweise an der rechten und unteren Seite des Fensters angezeigt. Um einen Bildlauf durchzuführen, ziehen Sie das Bildlauffeld oder klicken auf die Pfeile im Scrollbar. Das Bildlauffeld zeigt die Position innerhalb der Datei an.

Ein Mausklick unterhalb des oberen Pfeils bzw. oberhalb des unteren Pfeils beim vertikalen Scrollbar bewirkt ein Blättern um eine Seite nach unten bzw. oben.

Ein Mausklick neben den Pfeilen beim horizontalen Scrollbar bewirkt das Verschieben um eine Fensterbreite.

Die Anzeige der Scrollbars im Datenfenster kann mit dem Menü-Befehl Ansicht Scrollbar waagrecht bzw. Scrollbar senkrecht ein- bzw. ausgeschaltet werden.

Statuszeile

Die Statuszeile wird als letzte Zeile im Fenster unterhalb der Kommandozeile angezeigt. Um die Statuszeile ein- bzw. auszublenden, schalten Sie die Option Statuszeile im Menü Ansicht ein bzw. aus.

1. Feld
Hier werden Hinweise auf Eingabemöglichkeiten angezeigt.

2. Feld
Anzahl der Zeilen im aktuellen Arbeitsbereich.

3. Feld
Zeilennummer der Zeile, in der sich der Cursor befindet.

4. Feld
Spaltenposition des Cursors.

5. Feld
Länge des Fensters in Zeilen / Breite des Fensters in Spalten.

6. Feld
Einfügemodus:

EINF: Insert. An der aktuellen Cursorposition werden Zeichen eingefügt.
UEB: Überschreiben. Bestehende Zeichen werden überschrieben.
Der Modus kann mit der Taste <Einfg> bzw. <Ins> umgeschalten werden.

7. Feld
ASCII / ANSI / EBCDIC: Eingestellte Code-Darstellung (S. **94**) des aktuellen Arbeitsbereichs.

8 Markierungsspalte

+ Positioniert das Datenfenster. Die markierte Zeile wird als erste Zeile angezeigt.

- Positioniert das Datenfenster. Die markierte Zeile wird als letzte Zeile angezeigt.

A After. Mit C, M oder R markierte Zeilen werden nach dieser Zeile eingefügt.

B Before. Mit C, M oder R markierte Zeilen werden vor dieser Zeile eingefügt.

C Copy. Die Zeile wird zum einmaligen Kopieren an eine andere Stelle vorgemerkt. Im Unterschied zum BS2000-EDT kann dieser in einer Aktion einmalig auch an mehreren Stellen eingefügt werden.

D Delete. Löscht diese Zeile.

E Extend. Löscht bzw. setzt das Zeilenende-Kennzeichen.
Bei gelöschtem Zeilenende-Kennzeichen wird die Zeile beim Zurückschreiben mit der folgenden Zeile verkettet, d.h. es wird kein CR/LF eingefügt. Ein gelöschtes bzw. fehlendes Zeilenende-Kennzeichen wird in der Zeilennummer mit einem Stern "*" anstelle eines Punktes "." dargestellt. Die Markierung E ist als Toggle-Schalter implementiert, d.h. der jeweils letzte Status wird umgekehrt.

Mit dem Kommando ERS (S. 191) kann das Format von mehreren Sätzen oder für die ganze Datei geändert werden.

F Setzt die FIND-Markierung für diese Zeile. Die Zeilennummer wird hervorgehoben dargestellt.

H Schaltet nur für den ausgewählten Satz den Hexadezimalmodus ein und stellt alle zu diesem Satz gehörenden Bildschirmzeilen auf überschreibbar. Die Hexa-Zeilen werden beim neuen Bildschirmaufbau nach dem Betätigen der Enter-Taste nicht mehr angezeigt.

I Insert. Bewirkt ein ständiges Einfügen von jeweils neun Zeilen. Es werden bei jedem <Enter> neun Leerzeilen vor der markierten Zeile eingefügt. Der Einfügemodus wird beendet, sobald die leeren Zeilen mit <Enter> bestätigt werden.

J Verkettet zwei Zeilen. Mit J muss die Zeile markiert werden, die an die davor liegende Zeile angefügt werden soll. Die markierte Zeile wird anschließend gelöscht.

K Die markierte Zeile wird in die Kommandozeile übernommen. Die Zeile kann auch länger als der sichtbare Teil der Kommandozeile sein. Eine Zeile aus einem Arbeitsbereich, der durch das Kommando FSTAT erzeugt worden ist, wird nur in der Länge des Dateinamens kopiert, d.h. beim ersten Leerzeichen wird abgebrochen.

L Low. Wandelt alle Großbuchstaben der markierten Zeile in Kleinbuchstaben um. Siehe auch Kommando LOWER.

M Move. Merkt diese Zeile zum Verschieben an eine andere Position vor.

N Not find. Hebt eine FIND-Markierung auf.

O Overwrite. Diese und evtl. folgende Zeilen werden mit dem Inhalt des Kopierpuffers (Markierung C, M oder R) überschrieben.

R Retain. Kopiert die Zeile in den Kopierpuffer zum mehrmaligen Einfügen an einer anderen Stelle.

S Split. Trennt die Zeile auf. Nach Eingabe der Markierung S wird die Zeile hervorgehoben dargestellt. Ab der Spalte, an der sich der Cursor befindet, wird die Zeile aufgeteilt, nachdem die ENTER-Taste betätigt wird.

U Uppercase letters. Wandelt alle Kleinbuchstaben der markierten Zeile in Großbuchstaben um. Siehe auch Kommando UPPER.

X Gibt die Zeile zum Überschreiben frei.

n Fügt n Leerzeilen vor der markierten Zeile ein (n = Ziffer 1 - 9).

***** Löscht den Kopierpuffer. Alle vorhergehenden Markierungen C, M, R gehen dadurch verloren.

Eingaben in FSTAT-Arbeitsbereichen

Zu den Eingaben für die Markierungsspalte bei FSTAT-Arbeitsbereichen siehe S. **204**

9 Kommandos

Abkürzungsregeln für Kommandos und Parameter
Wie im BS2000 können alle EDT-Kommandos und Parameter beliebig abgekürzt werden. Die kürzeste Form ist in der Beschreibung fett gedruckt. Die Langform wird in einer kleineren Schrift dargestellt, z.B. **CH**ECK. Sie können dieses Kommando also in den Varianten `check`, `chec`, `che` oder `ch` eingeben.

Eine darüber hinausgehende Abkürzung, z.B. `copyf` für `copyfile` ist bis zur Eindeutigkeit möglich. In Prozeduren sollte dies aber vermieden werden, da diese Abkürzungen für neue Versionen nicht garantiert werden können.

Die Syntax des EDT unter WINDOWS ist mit der des EDT unter BS2000 grundsätzlich identisch. Für einige Kommandos werden zusätzliche Optionen angeboten.

Verkettung mehrerer Kommandos
In der Kommandozeile des EDT können mehrere Kommandos nacheinander eingegeben werden. Die einzelnen Kommandos werden durch das Semikolon ";" getrennt. Beispiel: `dma;o&f'edt'`

Datenfenster verschieben

+ / **+***n* / **-** / **-***n* / **--** / **++** — Verschiebt das Datenfenster nach oben / unten.

#*n* — Positionieren auf die Zeilennummer *n*

+P — Positionieren nach dem Kommando ON..FIND auf den ersten Treffer in der nächsten Seite.

> / **>***n* / **>>** / **<** / **<***n* / **<<** — Verschiebt das Datenfenster nach links / rechts.

C *n* — Datenfenster auf Spalte *n* positionieren. Satzanfang = C1.

CX'*hex*' — Datenfenster auf Spalte X'*hex*' positionieren. Satzanfang = CX'0'.

Arbeitsbereich wechseln

0 ..... **32** — Im EDT stehen 33 Arbeitsbereiche (0 bis 32) zur Verfügung. In jeden Bereich können Dateien eingelesen bzw. eingegeben werden. Nach dem Laden von EDT wird der Arbeitsbereich 0 angezeigt. Wurden mehrere Dateien eingelesen, so wird der Arbeitsbereich angezeigt, in den die letzte Datei eingelesen wurde. Durch Eingabe einer Zahl von 0 bis 32 in der Kommandozeile wechseln Sie den Arbeitsbereich.

Inhalt einer String-Variablen als EDT-Kommando ausführen

#s*nn* — Ausführen des EDT-Kommandos, das in der Stringvariablen enthalten ist.

Beispiel:

```
set #s01 = 'Hallo'
set #s30 = 'do 1(' + #s01 + ')'
#s30
```

Hintergrund dieses Beispiels ist, dass es beim Kommando DO nicht möglich ist, den Inhalt einer Stringvariablen zu übergeben. Das Kommando `@do1(#s01)` würde den String '#s01' als Parameter übergeben und nicht den Inhalt der String-Variablen.

MS-DOS-Kommando ausführen

!['*cmd*'|*str-var*] [,W|NW] [,P|NP] [,MIN|MAX|RES] [,NC]

Führt das MS-DOS-Kommando *cmd* aus. Als *cmd* kann ein beliebiges MS-DOS-Kommando angegeben werden. Das Kommando ist identisch mit dem Kommando SYS (S. 252).

Akustisches Signal ausgeben

BEEP [*tonart*] [,*int*] Akustisches Signal ausgeben.

tonart Mit der Angabe 0 - 5 können 6 verschiedene Signale ausgewählt werden (Standard = 0).

int Anzahl der akustischen Signale.

Positionieren aus Spalte

C *n* Datenfenster auf Spalte *n* positionieren. Satzanfang = C1.

CX'*hex*' Datenfenster auf Spalte X'*hex*' positionieren. Satzanfang = CX'0'.

Verketten von String-Variablen

CAT *send ,anz ,ziel* [,*trenn*] [**,** *intvar*]

Beginnend mit der Variablen *send* werden so viele String-Variablen, wie bei *anz* angegeben miteinander verkettet. Das Ergebnis wird in die String-Variable *ziel* übertragen.

send Erste String-Variable, die verkettet werden soll.

anz Anzahl der String-Variablen, die verkettet werden sollen.

ziel String-Variable, in die das Ergebnis der Verkettung übertragen werden soll.

trenn 'string' | x'hexastring' | *strvar*

Trennzeichen (ein oder mehrere Zeichen). Diese Zeichenfolge wird zwischen die einzelnen Zeichenfolgen eingefügt, die in ziel zusammengefügt werden. Bei fehlender Angabe wird ein Leerzeichen (x'20') eingefügt. Es kann auch eine String-Variable *str* angegeben werden.

intvar In der Integer-Variablen *intvar* wird die Anzahl der verketteten Argumente zurückgeliefert. Standard=#I0.

Hinweis:

Mit dem Kommando `CUT` (S. 186) können Zeichenfolgen einer Variablen getrennt und in einzelne String-Variablen übertragen werden.

Beispiel:

```
set #s1='Beispiel'
set #s2='für'
set #s3='eine'
set #s4='Verkettung'
cat #s1,4,#s5

status

Integer - Variables
#I00=+00000004

String - Variables
#S01(0008)=Beispiel
#S02(0003)=für
#S03(0004)=eine
#S04(0010)=Verkettung
#S05(0028)=Beispiel für eine Verkettung
```

Der Inhalt der String-Variablen #S1 bis #S4 wird in die String-Variable #S5 übertragen, wobei zwischen den einzelnen Teilfeldern ein Leerzeichen eingefügt wurde. Das Kommando `STATUS` erzeugt die angegebene Bildschirmanzeige.

Arbeitsverzeichnis/FT-Profil wechseln

CHDIR [*l:*] [*pfad*]

Wechseln des Arbeitsverzeichnisses. Nach dem Laden des EDT ist das Arbeitsverzeichnis aktiv, das in den Verknüpfungsparametern festgelegt ist bzw. das aktuelle Arbeitsverzeichnis, falls EDT in einer DOS-Box geladen wird. Das Arbeitsverzeichnis wird für die Kommandos READ oder WRITE verwendet, falls kein Pfadname angegeben ist. Das Arbeitsverzeichnis dient ebenfalls für die Dialogboxen "Öffnen ..." und "Speichern unter ... " als Einstiegspfad.

Das Arbeitsverzeichnis wird nach Ausführung des Kommandos in die Statuszeile geschrieben. Falls kein Parameter angegeben ist, wird das aktuelle Arbeitsverzeichnis in die Statuszeile geschrieben.

l

Laufwerksbuchstabe des Arbeitsverzeichnisses.

pfad

Pfadname des Arbeitsverzeichnisses. Es kann auch ein relativer Pfadname angegeben werden.

CHDIR '*ft* | *batch* R= [*prefix* | *plam-lib*]' [*,scope*]

Filetransfer-Profil als Standard für nachfolgende Kommandos `READ` und `WRITE` einstellen. Bei den Kommandos `READ` und `WRITE` ist dann nur noch die Angabe des Dateinamens (ggf. nur Suffix) bzw. des Bibliothekelements notwendig.

Filetransfer-Profil

ft

Name eines Filetransfer-Profils, das mit dem Menübefehl Extras/Filetransfer.... (S. 182) definiert werden kann. Aus den Parametern des Kommandos und den Angaben aus dem Profil wird die Schnittstelle des internen EDT-Filetransfers erzeugt und der Filetransfer gestartet.

Sind Profile vorhanden, die in mehreren INI-Dateien mit dem gleichen Namen enthalten sind, kann durch Voranstellen der Dateinummer (1, 2 oder 3) die entsprechende INI-Datei angegeben werden.

Beispiel:

`1std` = Profil `STD` aus der privaten INI-Datei
`2std` = Profil `STD` aus der INI-Datei vom Windowsverzeichnis
`3std` = Profil `STD` aus der INI-Datei vom Ladeverzeichnis.

batch — Dateiname der Batchdatei. Die Extension ".BAT" muss hierbei nicht angegeben werden. Die Batch-Prozedur wird mit folgenden Parametern gestartet:

batch-file par1 par2 .....

prefix — Prefix für den Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID im BS2000 bzw. Unix-Pfad. Dieser Prefix wird dem Dateinamen im `READ` bzw. `WRITE`-Kommando vorangestellt.

plam-lib — *lib*([*typ* /] [*prefix-elem*])

Name einer PLAM-Bibliothek. Bei nachfolgenden Kommandos `READ` bzw. `WRITE` ist nur noch der Elementname bzw. der Suffix zum Elementnamen anzugeben.

lib — Name der PLAM-Bibliothek auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID.

typ — Typ des PLAM-Elements. Es sind nur die Angaben S, M, J, P und D zulässig. Wird kein Typ angegeben, so wird der Typ "S" als Standard benutzt.

prefix-elem — Prefix zum Namen des PLAM-Elements.

Hinweis:

Beginnt der Dateiname im `READ` bzw. `WRITE`-Kommando mit einer CAT-ID (:*xxx*) oder einer User-ID ("$"), wird der Parameter *prefix* | *plam-lib* ignoriert. Beginnt der Dateiname im `READ` bzw. `WRITE`-Kommando mit einem Laufwerksbuchstaben oder dem Zeichen "\", wird das Profil ignoriert und eine lokale Datei gelesen bzw. geschrieben.

CHDIR R=OFF [,*scope*] — Die Vorgaben zum fernen `READ/WRITE` werden deaktiviert. Bei Dateinamen ohne absoluten Pfad gilt wieder das aktuelle Arbeitsverzeichnis.

CHDIR R=ON [,*scope*] — Die Vorgaben zum fernen `READ/WRITE` eines vorausgegangenen `CHDIR` werden wieder aktiviert.

CHDIR ? — Die globalen und für den aktuellen Arbeitsbereich geltenden FT-Angaben sowie das aktuelle Arbeitsverzeichnis werden in einer Messagebox angezeigt.

scope — Geltungsbereich für die FT-Angaben: CURRENT | GLOBAL | *arb* .

CURRENT — Das Kommando gilt nur für den aktuellen Arbeitsbereich.

GLOBAL — Das Kommando gilt für alle Arbeitsbereich. Diese Einstellung ist Standard.

arb — Das Kommando gilt für den angegebenen Arbeitsbereich. Existiert der Arbeitsbereich noch nicht, wird ein neues Fenster erzeugt.

Beispiele:

```
chdir 'test1 r='
read'testdat1'
```
ergänztes Kommando `READ`:
```
read'test1 r=testdat1
```

```
chdir 'test1 r=:test:$test.prefix.
read'testdat1'
write'testdat2'
```
ergänztes Kommandos
```
read'test1 r=:test:$test.prefix.testdat1
write'test1 r=:test:$test.prefix.testdat2
```

```
chdir 'joblib r=:test:$test.joblib(j/pr1.)
read'job1'
read'\test\test1
read'$test.test1
```
ergänzte Kommandos:
```
read'joblib r=:test:$test.joblib(j/pr1.job1)
read'\test\test1   Profil wird ignoriert: lokale Datei
read'joblib r=$test.test1   Lib wird ignoriert
```

```
chdir 'src r=srclib()
read'src1'
```
ergänztes Kommando `READ`:
```
read'src r=srclib(s/src1)
```

Zeilen prüfen

CHECK [ON|OFF] | [,*cl*]

Diese Anweisung ermöglicht die Protokollierung jeder Zeile, die durch die Kommandos `COLUMN, COPY, CREATE, MOVE, ON... PREFIX oder SUFFIX` verändert werden.

ON

Die Zeilenprüfung wird eingeschaltet. Die geänderten Zeilen werden im Protokollbereich angezeigt. Alle geänderten Zeilen werden mit der FIND-Markierung versehen, d.h. neben der farblichen Hervorhebung kann mit der Taste F3 bzw. mit der Schaltfläche  auf den nächsten geänderten Satz positioniert werden.

OFF

Die Zeilenprüfung wird ausgeschaltet. Die Überprüfung der Zeilenlänge bleibt davon unberührt. Soll die Überprüfung der Zeilenlänge ausgeschaltet werden, muss *cl* wieder auf die maximale Satzlänge eingestellt werden.

cl

Zeilenlänge für die Zeilenlängenüberprüfung. Wird diese Länge bei der Eingabe im Datenbereich überschritten, wird eine Warnung ausgegeben.

Format 1: Editier-Code einstellen

CODE *codevar*

Mit dem Format 1 des Kommandos wird lediglich der Code für das Editieren und die Betrachtung der Daten eingestellt. Es erfolgt keine Änderung oder Code-Umsetzung der bestehenden Daten im Arbeitsbereich.

codevar

Folgende Code-Varianten können verwendet werden:

ANSI	MS Windows-Dateien (CP-1252)
ASCII	ASCII MS-DOS (CP850)
ASCII2	ASCII MS-DOS (CP850 Variante 2)
ASCIIUNIX	ASCII für Unix
EBCDIC	EBCDIC BS2000 7-Bit (CCSN=EDF03IRV)
EBCDIC8	EBCDIC BS2000 8-Bit (CCSN=EDF041)
CP500	EBCDIC IBM S/390 (CP500)
ISO8859-1	ISO8859-1
UNICODE	UNICODE-Dateien (ISO 10646)

Es können noch weitere Code-Varianten definiert werden, siehe dazu die Beschreibung im Kapitel 17 Code-Tabellen (S. 359). Statt der Schlüsselwörter ASCII, ANSI usw. können auch die Aliasnamen, die in der Datei CODEPAGE.TXT definiert sind, verwendet werden.

Die ASCII-Codierung unterscheidet sich von der ANSI-Codierung (CP-1252) hauptsächlich in den Umlauten und Sonderzeichen, während sich die EBCDIC-Codierung in allen Zeichen von der ASCII/ANSI-Codierung unterscheidet.

Der Code IS8859-1 unterscheidet sich von ANSI in den Positionen X'82'-X'9F'. Der Code ANSI (CP-1252) hat an diesen Positionen Sonderzeichen (81 - 8E = ‚ƒ„…†‡ˆ‰Š‹ŒŽ, 91 - 9C = ‘’“”•–—˜™š›œ, 9E = ž, 9F = Ÿ), der CODE ISI8859-1 hat an diesen Positionen nicht abdruckbare Steuerzeichen.

Falls im Arbeitsbereich Daten mit einer anderen Codierung stehen, kann hier der entsprechende Editier-Code ausgewählt werden. Dies bewirkt, dass die Darstellung der abdruckbaren Zeichen umgestellt wird und dass alle Datenänderungen im ausgewählten Code durchgeführt werden.

Im Hexa-Modus (Kommando HEX ON) wird z.B. im EBCDIC-Code das Zeichen "A" mit "C1" wie im BS2000 dargestellt. Die Codierung einer Datei wird in bestimmten Fällen beim Einlesen automatisch erkannt und eingestellt (siehe Menü Optionen / Code Standard, Option Auto als Standard-Code (S. 116)).

Die aktuelle Code-Einstellung wird in der Statuszeile angezeigt.

Eine Änderung des Editier-Codes bewirkt keine Umcodierung der bereits bestehenden Daten. Die bestehenden Daten können mit dem Format 2 des Kommando CODE (S. 181) bzw. über das Menü Funktionen / Code-Umsetzung (S. 83) konvertiert werden.

Die Funktionen dieses Kommandos können auch über die Menüzeile aufgerufen werden (Ansicht / Code-Anzeige (S. 94).

Format 2: Codierung der Daten ändern

CODE *code1* TO *code2*

code1 / code2 Alter bzw. neue Code-Variante

ANSI	MS Windows-Dateien (CP-1252)
ASCII	ASCII MS-DOS (CP850)
ASCII2	ASCII MS-DOS (CP850 Variante 2)
ASCIIUNIX	ASCII für Unix
EBCDIC	EBCDIC BS2000 7-Bit (CCSN=EDF03IRV)
EBCDIC8	EBCDIC BS2000 8-Bit (CCSN=EDF041)
CP500	EBCDIC IBM S/390 (CP500)
ISO8859-1	ISO8859-1
UNICODE	UNICODE-Dateien (ISO 10646)

Es können noch weitere Code-Varianten definiert werden, siehe dazu die Beschreibung im Kapitel 17 Code-Tabellen (S. 359). Statt der Schlüsselwörter ASCII, ANSI usw. können auch die Aliasnamen, die in der Datei CODEPAGE.TXT definiert sind, verwendet werden.

Es erfolgt eine Umsetzung der Daten aus der ersten Code-Variante in die zweite. Im Gegensatz zu der oben dargestellten Form des Kommandos CODE werden hier die Daten konvertiert. Gleichzeitig wird die Darstellung auf die neue Code-Variante eingestellt.

Beispiel:

```
code ascii to ebcdic    oder  code atoe
code ansi to ebcdic     oder  code antoe
code ansi to unicode    oder  code antounicode
```

Hinweis:

Die Code-Tabellen ISO8859-1, EBCDIC8 und UNICODE enthalten den gleichen Umfang an Zeichen. Bei einer Konvertierung zwischen diesen Codes werden alle Zeichen, auch nicht abdruckbare Zeichen, eindeutig umgesetzt. Bei den anderen Codes ist dies nicht der Fall, so dass Zeichen, die im Zielcode nicht vorhanden sind, den Wert X'00' erhalten.

Die Funktionen dieses Kommandos können auch über die Menüzeile aufgerufen werden (`Funktionen / Code-Umsetzung` (S. 83)).

Zeichenfolge einfügen/austauschen/ersetzen

COLUMN *cl* **O**N { *rng*|*str-var* } **C**HANGE *str* [,**N**OSTRIP]

Text ab Spalte *col* mit Zeichenfolge *str* überschreiben. Falls der Datensatz weniger Spalten enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht, soweit die Option NOSTRIP nicht angegeben ist.

cl — Spalte, ab der Text ersetzt werden soll.

NOSTRIP — Die rechtsbündigen Leerzeichen werden nicht gelöscht.

Beispiel:

```
col81on1-100c'x'
```

In den Zeilen 1 bis 100 wird das Zeichen in Spalte 81 durch das Zeichen 'x' ersetzt. Rechtsbündige Leerzeichen werden gelöscht.

COLUMN *cl* **O**N { *rng*|*str-var* } **I**NSERT *str* [,**N**OSTRIP]

Zeichenfolge *str* ab Spalte *col* einfügen. Falls der Datensatz weniger Spalten enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht, soweit die Option NOSTRIP nicht angegeben ist.

cl — Spalte, ab der Text eingefügt werden soll.

NOSTRIP — Die rechtsbündigen Leerzeichen werden nicht gelöscht.

Beispiel: `col20on&i'abc',n`
In allen Zeilen des Arbeitsbereichs wird der String 'abc' ab Spalte 20 eingefügt. Rechtsbündige Leerzeichen werden nicht gelöscht.

COLUMN *cl1-cl2* **O**N { *rng*|*str-var* } **R**EPLACE *str* [,**N**OSTRIP]

Spaltenbereich *cl1-cl2* löschen und neue Zeichenfolge *str* einfügen. Falls ein Datensatz weniger Spalten als die Spalte "von" enthält, werden Leerzeichen eingefügt. Anschließend werden im Zeilenbereich alle rechtsbündigen Leerzeichen gelöscht, soweit die Option NOSTRIP nicht angegeben ist.

cl-cl2 — Spaltenbereich von bis, in dem der Text ersetzt werden soll.

NOSTRIP — Die rechtsbündigen Leerzeichen werden nicht gelöscht.

Beispiel: `col5-20on&r'abc'`
In allen Zeilen des Arbeitsbereichs werden die Spalten 5-20 gelöscht und anschließend vor Spalte 5 der String 'abc' eingefügt. Rechtsbündige Leerzeichen werden gelöscht.

Hinweis:

Entstehen durch das Löschen der rechtsbündigen Bytes Leerzeilen (z.B. `COL81o&c' '`), werden diese im Gegensatz zum BS2000-EDT nicht gelöscht. Falls Sie diese Leerzeilen löschen wollen, können Sie das Kommando `ON.....FIND ED` (S. **215**) verwenden.

Zwei Arbeitsbereiche vergleichen

COMPARE [*n1* WITH] *n2* [LIST *n3*] [TRANS [ON] OFF] [,NF]

Die Daten aus dem Arbeitsbereich *n1* werden mit dem Arbeitsbereich *n2* zeilenweise verglichen. Das Ergebnis wird in den Arbeitsbereich *n3* geschrieben. Wird *n3* nicht angegeben, wird das Ergebnis in den Arbeitsbereich 9 geschrieben, der nach dem Vergleich automatisch angezeigt wird. Als Ergebnis werden die Zeilennummer und der Inhalt der abweichenden Zeilen angezeigt. Alle abweichenden Zeilen in den Arbeitsbereichen *n1 und n2* werden mit der FIND-Markierung versehen, d.h. neben der farblichen Hervorhebung kann mit der Taste F3 bzw. mit der Schaltfläche  auf den nächsten abweichenden Satz positioniert werden.

n1 Nummer des ersten Arbeitsbereichs, der verglichen werden soll. Ist keine Nummer angegeben, wird der aktuelle Arbeitsbereich verwendet.

n2 Nummer des zweiten zu vergleichenden Arbeitsbereichs.

n3 Nummer des Arbeitsbereichs, in dem das Ergebnis abgelegt wird. Bei fehlender Angabe wird Arbeitsbereich 9 verwendet.

Codierung der Arbeitsbereiche

TRANS [ON] Falls die zu vergleichenden Arbeitsbereiche unterschiedlich codiert sind (ANSI/ASCII/EBCDIC/UNICODE), werden die Daten des zweiten Arbeitsbereichs vor dem Vergleich in die Code-Variante des ersten Arbeitsbereichs übersetzt bzw. falls ein UNICODE-Arbeitsbereich beteiligt ist, werden beide Arbeitsbereiche in UNICODE vergleichen. Die Übersetzung erfolgt nur intern, die Daten beider Arbeitsbereiche bleiben unverändert.

Diese Einstellung ist Standard bei Textdateien.

TRANS OFF Falls die zu vergleichenden Arbeitsbereiche unterschiedlich codiert sind (ANSI/ASCII/EBCDIC/UNICODE), werden die Daten ohne Code-Angleichung verglichen. In den Ergebnis-Bereich werden die Daten ebenfalls in der Original-Codierung übertragen.

Diese Einstellung ist Standard bei Binärdateien.

Der Translate-Modus kann auch mit dem Kommando `PAR TRANS=YES|NO` (S. **218**) eingestellt werden.

Der Ergebnis-Arbeitsbereich ist in UNICODE codiert, falls einer der zu vergleichenden Arbeitsbereiche ein UNICODE-Arbeitsbereich ist. Andernfalls ist der Ergebnis-Arbeitsbereich in ANSI codiert.

NF No Focus: Nach der Erstellung der Ergebnisdaten wird nicht der Arbeitsbereich 9 bzw. der angegebene Ergebnis-Arbeitsbereich angezeigt. Der Fokus bleibt unverändert. Diese Option ist insbesondere für Prozeduren gedacht.

Um in Prozeduren das Vergleichsergebnis abfragen zu können, wird zusätzlich der EDT-Fehlerschalter gesetzt, falls die Arbeitsbereiche nicht gleich sind (Abfragen von Fehlerschalter siehe `IF ERRORS` (S. 279)). Vor dem Vergleichen muss der EDT-Fehlerschalter mit `RESET` (S. 287) zurückgesetzt werden.

Beispiele:

`comp 1`
Aktuellen Arbeitsbereich mit Arbeitsbereich 1 vergleichen. Das Ergebnis wird im Arbeitsbereich 9 angezeigt.

`comp 0w1`
Arbeitsbereich 0 mit 1 vergleichen. Das Ergebnis wird im Arbeitsbereich 9 angezeigt.

`comp 0w1l4 trans off`
Vergleich des Arbeitsbereichs 0 mit 1. Das Ergebnis wird im Arbeitsbereich 4 angezeigt. Die Code-Angleichung wird ausgeschaltet.

```
@@reset
@@comp 1
@@if no errors goto gleich
@@write 'file1'
@@:gleich
```
In einer EDT-Prozedur aktuellen Arbeitsbereich mit Arbeitsbereich 1 vergleichen. Falls die Arbeitsbereiche gleich sind, verzweigt die Prozedur zur Sprungmarke `gleich`. Sind die Arbeitsbereiche ungleich, wird der aktuelle Arbeitsbereich in die Datei `file1` geschrieben.

Zeilen aus anderem Arbeitsbereich kopieren

COPY *rng* (*n*)
Kopiert Daten aus dem Arbeitsbereich *n* in den aktuellen Arbeitsbereich. Die Zeilennummern des Sendebereichs *n* bleiben erhalten. Es werden die Zeilen des Sendebereiches mit den originalen Zeilennummern in den aktuellen Arbeitsbereich kopiert, d.h. Zeilen im Zielbereich können überschrieben werden.

COPY { *rng*|*str-var* } [(*n*)] **TO** *ln1*[(*inc*)] [:*ln2*] [,*ln1*[(*inc*)] [:*ln2*] ,.....]
Kopiert Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich. Die kopierten Zeilen erhalten Nummern beginnend mit *ln1* in der Schrittweite *inc*. Die höchste Zeilennummer ist *ln2*. Hierbei können die Zeilen im Zielbereich überschrieben werden. Es können auch mehrere Zielbereiche, durch Komma getrennt, angegeben werden.

COPY { *rng*|*str-var* } [(*n*)] { AFTER|BEFORE } *ln*
Kopiert Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor bzw. hinter die Zeile mit der Nummer *ln*. Es werden keine Zeilen des Zielbereichs überschrieben. Dieses Kommando stellt eine **Erweiterung** zum COPY-Kommando des BS2000-EDT dar.

COPY { *rng*|*str-var* } [(*n*)] { FIRST|LAST }
Kopiert Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor die erste bzw. hinter die letzte Zeile. Es werden keine Zeilen des Zielbereichs überschrieben. Dieses Kommando stellt eine **Erweiterung** zum COPY-Kommando des BS2000-EDT dar.

Datei kopieren

COPYFILE *file1=file2* [O] Die Datei *file1* wird in die Datei *file2* kopiert. Zum Umbenennen bzw. Verschieben von Dateien können Sie das Kommando `MOVEFILE` (S. **210**) verwenden.

file1 Dateiname der Ursprungsdatei als String (S. **341**). Es sind alle Varianten, wie Stringvariable, Zeilennummer usw. zulässig.

file2 Dateiname der neuen Datei als String (S. **341**). Es sind alle Varianten, wie Stringvariable, Zeilennummer usw. zulässig.

O Overwrite: Falls die neue Datei *file2* bereits existiert, wird sie ohne Rückfrage überschrieben. Ohne diese Option wird über eine Messagebox gefragt, ob die Datei überschrieben werden soll. Im Prozedur-Modus wird die Datei auch ohne Angabe des Overwrite-Parameters immer überschrieben.

Beispiele:

```
copyfile 'datei1'='datei2' o
copyfile #s1=#s2+'neu'
copyfile #l1:1-30:=#l1:1-30:+'neu'
```

Textzeilen erzeugen

CREATE [*ln* | *str-var*] [:] *str* [, *str....*]

Schreibt eine Zeichenfolge in eine beliebige Zeile oder String-Variable. Dieses Kommando wird vor allem in EDT-Prozeduren benötigt, um neue Zeilen zu erzeugen. Eine bereits bestehende Zeile wird überschrieben. Die zweite Variante des Kommandos CREATE ist im Kapitel 6. "Kommandos für Prozedurdateien" beschrieben.

ln Eine Zeile mit dieser Nummer wird erzeugt.

str-var String-Variable, die erzeugt werden soll.

CREATE ... **READ** .. Diese Variante des CREATE-Kommandos ist im Kapitel Prozedursprache (S. **275**) beschrieben.

Beispiele:

```
create 500:'neue Zeile 500'
create #s10:'test',#s11:5-10:,500
create %-0.1:'neue Zeile vor der ersten Zeile'
create $+1:'neue Zeile nach der letzten Zeile'
```

String-Variable aufteilen

CUT *send ,anz ,ziel* [**,** *start*] [**,** *trenn*] [**,** *intvar1* **,** *intvar2*]

Der Inhalt einer String-Variablen wird in Teil-Zeichenfolgen zerlegt. Jede Teil-Zeichenfolge in eine separate String-Variable geschrieben.

send String-Variable, die aufgeteilt werden soll.

anz Anzahl der String-Variablen, in die das Ergebnis übertragen werden sollen.

ziel Erste String-Variable, in die das Ergebnis der Aufteilung übertragen werden soll.

start Nummer der ersten Teilzeichenfolge, die zu verarbeiten ist.
Standard = 1

trenn 'string' | x'hexastring' | *strvar*

Trennzeichen (ein oder mehrere Zeichen), das zur Ermittlung der Teil-Zeichenfolgen verwendet werden soll. Bei jedem einzelnen Zeichen dieser Zeichenfolge wird das Ende einer Teil-Zeichenfolge erkannt. Bei fehlender Angabe wird als Trennzeichen das Leerzeichen (x'20') und das Tabulatorzeichen (x'09') verwendet. Es kann auch eine String-Variable *str* angegeben werden.

intvar1 Ablieferung der Anzahl der tatsächlich erzeugten String-Variablen. Standard = #I0.

intvar2 Ablieferung der Anzahl der tatsächlich vorhandenen Teilstrings in der Sende-Variablen. Standard = #I1.

Hinweis:

Mit den Kommandos CAT und CREATE können Zeichenfolgen aus mehreren String-Variablen verkettet werden.

Beispiel:

```
set #s6='Beispiel für eine Aufteilung'
cut #s6,4,#s1

status

Integer - Variables
#I00=+00000004
#I01=+00000004

String - Variables
#S01(0008)=Beispiel
#S02(0003)=für
#S03(0004)=eine
#S04(0010)=Aufteilung
#S06(0028)=Beispiel für eine Aufteilung
```

Die Teil-Zeichenfolgen (Wörter) der String-Variablen #S5 werden in die String-Variable #S1 bis #S4 übertragen. Das Kommando `STATUS` erzeugt obige Bildschirmanzeige. Als Trennzeichen wird nur das Leerzeichen verwendet.

Zeilenbereich löschen

DELETE [*rngcol*] [MULTIPLE FIRST|LAST]

Löscht eine oder mehrere Zeilen bzw. Teile von Zeilen. Kommando `DELETE` ohne Parameter löscht den ganzen Arbeitsbereich. Es können auch String-Variable mit dem Kommando `DELETE` gelöscht werden. Mit dem Kommando `UNDO` (siehe S. **254**) können gelöschte Zeilen wieder zurückgeholt werden.

MULTIPLE FIRST oder MF Im angegebenen Bereich werden alle gleichen Zeilen bis auf die letzte Zeile gelöscht (z.B. Zeilen 3 bis 5 sind gleich: Es werden die Zeilen 3 und 4 gelöscht). Enthält *rngcol* einen Spaltenbereich, werden nur die angegebenen Spalten verglichen.

MULTIPLE LAST oder ML Im angegebenen Bereich werden alle gleichen Zeilen bis auf die erste Zeile gelöscht (z.B. Zeilen 3 bis 5 sind gleich: Es werden die Zeilen 4 und 5 gelöscht). Enthält *rngcol* einen Spaltenbereich, werden nur die angegebenen Spalten verglichen.

Variable löschen

DELETE **A**LL **V**ARIABLES — Löscht alle String-, Float-, Integer- und Line-Variablen.

Textbegrenzerzeichen vereinbaren

DELIMIT = R | *str1* | { +|- } *str2* | ?

Definiert eine Menge von Zeichen, die beim Suchen einer Zeichenfolge mit dem Kommando `ON` als Textbegrenzerzeichen fungieren (siehe auch Parameter *str* auf S. 344).

R — Setzt die Textbegrenzerzeichen auf die Standardzeichen. Diese besteht aus `+.!*();-/,?:'="` sowie dem Leerzeichen (X'20') und dem Tabulatorzeichen (X'09').

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 (–).

? — Zeigt die Textbegrenzerzeichen in einem Fenster an.

Wird kein Operand angegeben, ist die Menge der Textbegrenzerzeichen leer. Dies bedeutet, dass ein nachfolgendes Kommando `ON` mit Textbegrenzerzeichen-Suche nur dann Treffer findet, wenn der Satz genau aus der gesuchten Zeichenfolge besteht.

Löschen von Zeilenmarkierungen

DMA — Delete Marks. Löscht im aktuellen Arbeitsbereich die mit dem Kommando `ON...FIND` gesetzten Zeilen- und Treffermarkierungen.

DMA ON — Schaltet den Modus "Delete Mark" ein. Vor jedem Kommando `ON...FIND` werden alle bisherigen im gesamten Arbeitsbereich gelöscht. Dieser Modus hat die gleiche Wirkung, als wenn vor jedem `ON..FIND`.... das Kommando DMA eingegeben würde (z.B. `dma;on&find'TEST'`).

Menüzeile: `Optionen / Suchen` (S. 110)

DMA OFF — Schaltet den Modus "Delete Mark" aus. Standard.

Beispiele:

```
dma on
```
Der Modus "Delete Mark" wird eingeschaltet. Eventuell vorhandene Zeilenmarkierungen werden gelöscht.

```
on100-200find'TEST1'
```
Im gesamten Arbeitsbereichs werden vor der Ausführung des ON-Kommandos alle Zeilenmarkierungen gelöscht und in den Zeilen 100-200 alle Zeilen markiert, die `'TEST1'` enthalten.

```
on&find'TEST2'
```
Als Auswirkung des DMA ON-Modus werden in allen Zeilen des Arbeitsbereichs vor Ausführung des ON-Kommandos die bisherigen Zeilenmarkierungen gelöscht, d.h. nach dem ON-Kommando sind nur alle Zeilen markiert, die `'TEST2'` enthalten.

```
dma off
```
Alle bisher vorhandenen Zeilenmarkierungen bleiben erhalten. Der Modus "Delete Mark" wird ausgeschaltet.
```
dma;on100-200find'TEST1'
```
Vor der Ausführung des ON-Kommandos werden in allen Zeilen des Arbeitsbereichs die Zeilenmarkierungen gelöscht (DMA).
```
on&find'TEST2'
```
Als Auswirkung des DMA OFF-Modus werden vor der Ausführung des ON-Kommandos die bisherigen Zeilenmarkierungen ('TEST1') nicht gelöscht, so dass nach Ausführung des Kommandos alle Zeilen mit den Zeichenfolgen 'TEST1' und 'TEST2' markiert sind.

DMR Delete Marked records. Löscht alle mit dem Kommando `ON...FIND` oder Markierung `F` markierten Zeilen.

Arbeitsbereiche löschen

DROP [Q] [F] [ALL |*n*[,*n*...]] Löscht Arbeitsbereiche.

Q Query: Vor dem Löschen wird gefragt, ob der Arbeitsbereich wirklich gelöscht werden soll. Diese Option ist vor allem für Prozeduren vorgesehen.

F Das Löschen eines anderen Arbeitsbereichs ist auch möglich, wenn sich der Benutzer nicht im Arbeitsbereich 0 befindet.

ALL Die Arbeitsbereiche 1 bis 32 werden gelöscht. Im Gegensatz zum Kommando `DELETE` erfolgt keine Sicherung in den `UNDO`-Buffer, d.h. die Daten können mit dem Kommando `UNDO` nicht wieder hergestellt werden. Diese Option ist nur im Arbeitsbereich 0 oder im Zusammenhang mit dem Parameter F zulässig.

n Nummer des Arbeitsbereichs, der gelöscht werden soll. Es können beliebig viele Arbeitsbereichs-Nummern >0 angegeben werden. Diese Option ist nur im Arbeitsbereich 0 oder im Zusammenhang mit dem Parameter F zulässig.

Wenn weder die Option ALL noch ein Arbeitsbereich angegeben ist, wird der aktuelle Arbeitsbereich gelöscht.

Markierungsspalte und Daten stets editierbar

EDIT **F**ULL ON|OFF Die Markierungsspalte und das Datenfenster des aktuellen Datenbereichs werden permanent überschreibbar/nicht überschreibbar. Es kann auch das BS2000-Format `PAR EDIT FULL` ON|OFF verwendet werden. Der Modus wirkt nicht, falls mit `INDEX OFF` die Zeilennumerierung ausgeschaltet ist. Der EDIT FULL-Modus wird automatisch mit dem Kommando `EDIT WORD` eingeschaltet.

Cursorposition bei Zeilenwechsel definieren

EDIT **I**NDENT ON|OFF Cursorposition in neuer Zeile definieren. Das Kommando wirkt nur im EDIT WORD-Modus. Die Einstellung wirkt für alle Arbeitsbereiche und ist auch über den Parameter `Indent-Modus bei Eingabe` im Menüpunkt `verschiedene Optionen` des Menüs `Optionen` möglich (S. 127).

ON Wird mit der Taste `<Enter>` im Modus `EDIT WORD` in eine neu erstellte Zeile positioniert, so steht der Cursor in der Spalte, in der die Daten ungleich Leerzeichen in der vorhergehenden Zeile beginnend. Dies kann z.B. beim Editieren von Primärprogrammen hilfreich sein.

OFF — In einer neuen Zeile wird der Cursor immer auf Spalte 1 positioniert.

EDIT-LONG-Modus ein/ausschalten

EDIT **L**ONG ON|OFF — Datensätze werden in der vollen Länge (EL) oder nur in der Länge des Datenfensters angezeigt (ELO).

EDIT-SEQUENTIAL-Modus ein/ausschalten

EDIT **S**EQUENTIAL ON | OFF

ON — Der Operand beeinflusst das Hochzählen der aktuellen Zeilennummer. Im Regelfall wird 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.

OFF — Wird der Parameter OFF oder das Kommando EDIT ohne SEQUENTIAL angegeben, schaltet die Anweisung diese Funktion wieder ab.

EDIT-WORD-Modus ein/ausschalten

EDIT **W**ORD ON — In diesem Modus reagiert der EDT auf Tasteneingaben ähnlich wie die andere Editier- und Textverarbeitungsprogramme (z.B. WINWORD). Im Einzelnen bewirkt das Kommando folgendes:

- Falls sich der Cursor im Datenfenster befindet, bewirkt die Taste `<Enter>`, dass eine neue Zeile begonnen wird. Befindet sich der Cursor am Beginn oder am Ende einer Zeile, so bewirkt dies das Einfügen einer neuen Zeile. Befindet sich der Cursor innerhalb der Zeile, so bewirkt dies das Aufteilen der Zeile an dieser Stelle (split). Mit der Taste `<Esc>` gelangt man vom Datenfenster zur Kommandozeile.
- Befindet sich der Cursor am Ende einer Zeile, so wird mit der Taste `<Entf>` die aktuelle Zeile mit der nächsten Zeile verkettet.
- Befindet sich der Cursor in der ersten Spalte, so wird mit der Taste `<Back>` die aktuelle Zeile mit der vorhergehenden verkettet.
- Der `EDIT FULL`-Modus wird eingeschaltet, d.h. das Editierfenster ist immer überschreibbar (Markierung X oder Taste `F2` ist nicht notwendig).

EDIT **W**ORD OFF — Der `EDIT WORD`-Modus wird ausgeschaltet. Der EDT verhält sich wie im BS2000:

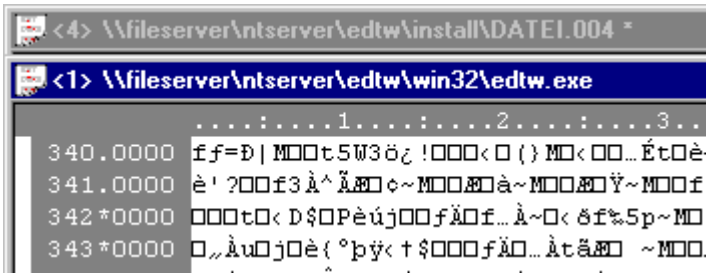
- Die Taste `<Enter>` bewirkt im Datenfenster, dass der Cursor in die Kommandozeile positioniert und evtl. ein Kommando ausgeführt wird. In die nächste Zeile des Datenfensters gelangt man nur mit der Taste `<Tab_Right>` oder `<Cursor_down>`.
- Der `EDIT FULL`-Modus wird auf den Status gesetzt, der mit dem Kommando `EDIT FULL` eingeschaltet wurde.

  Die Einstellung ist auch über das Menü `Optionen - WORD-Modus` möglich.

Zeilen-Trennzeichen einfügen/löschen

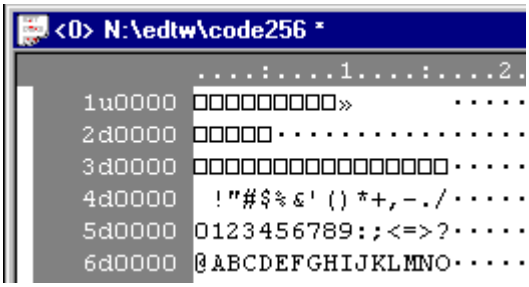
ERS [ON | OFF | RDOS | RUNIX [*rng*]]

Ändern des Satzformats. Zu jeder Zeile sind Informationen zum Format eines Satzes gespeichert:



Ein Stern in der Zeilennummer bedeutet, dass diese Zeile beim Zurückschreiben in eine Datei nicht um ein Satz-Trennzeichen ergänzt wird. Es handelt sich entweder um eine Binärdatei oder der Satz in einer satzstrukturierten Datei ist länger als die maximale Satzlänge. Die Zeile wird mit der nächsten Zeile verkettet.

Ein Punkt bedeutet, dass diese Zeile beim Zurückschreiben in eine Datei um ein Satz-Trennzeichen ergänzt wird. Die Art des Satz-Trennzeichens (`X'0D0A'` oder `X'0A'`) richtet sich danach, welche Dateistruktur (S. **40**) beim Einlesen festgestellt wurde.



Ein "u" oder "d" bedeutet, dass es sich um ein gemischtes Format handelt und diese Zeile beim Zurückschreiben in eine Datei um das Satztrennzeichen X'0D0A' (bei d) bzw. X'0A' (bei u) ergänzt wird.

Mit dem Kommando ERS kann nun dieses Merkmal für alle Zeilen oder einen ausgewählten Zeilenbereich des gesamten Arbeitsbereichs gesetzt oder gelöscht werden. Siehe hierzu auch die Ausführungen zu den Kommandos `REFORMAT` (S. **234**) und `UNFORMAT` (S. **255**).

Satztrennzeichen pro Satz

ON — Satztrennzeichen soll geschrieben werden, Punkt wird in Zeilennummer gesetzt (z.B. `1000.00`).

OFF — Satztrennzeichen soll nicht geschrieben werden, Stern wird in Zeilennummer gesetzt (z.B. `1000*00`).

RDOS — Als Satztrennzeichen für jeden Satz des angegebenen Bereichs bzw. für alle Sätze des Arbeitsbereiches soll `X'0D0A'` (MS-DOS/Windows) geschrieben werden.

RUNIX — Als Satztrennzeichen für jeden Satz des angegebenen Bereichs bzw. für alle Sätze des Arbeitsbereiches soll `X'0A'` (Unix) geschrieben werden.

ERS WDOS | WUNIX | WMIXED]

Satzformat gesamter Arbeitsbereich

Neben den einzelnen Sätzen gibt es auch eine Format-Kennzeichnung des gesamten Arbeitsbereichs.

WDOS — Für den gesamten Arbeitsbereich gilt `X'0D0A'` (MS-DOS/Windows). In der Zeilennummer wird als Satz-Trennzeichen für alle Sätze nur "." bzw. in Ausnahmefällen "*" angezeigt.

WUNIX — Für den gesamten Arbeitsbereich gilt `X'0A'` (Unix). In der Zeilennummer wird als Satz-Trennzeichen für alle Sätze nur "." bzw. in Ausnahmefällen "*" angezeigt.

WMIXED — Für jede einzelne Zeile gilt das gespeicherte Satz-Trennzeichen (`X'0A'` oder `X'0D0A'`). In der Zeilennummer wird als Satz-Trennzeichen für alle Sätze nur "d" oder "u" bzw. in Ausnahmefällen "*" angezeigt.

Mit der Markierung E kann für eine einzelne Zeile das Merkmal zum Schreiben des Satztrennzeichens gesetzt und wieder gelöscht werden.

Beispiele:

`ERS ON`	Satzende für alle Zeilen setzen
`ERS OFF`	Satzende für alle Zeile löschen
`ERS RDOS 1-2,14-15`	DOS-Format Zeilen 1-2 und 14-15
`ERS WMIXED`	gemischtes Format gesamter Arbeitsbereich.

Dateinamen voreinstellen

FILE *'file'* | *str-var'*

Voreinstellung des Dateinamens für die Kommandos `READ` und `WRITE`. Bei diesen Kommandos kann dann die Angabe des Dateinamens entfallen. Die Einstellung gilt nur für den aktuellen Arbeitsbereich.

Vorbelegung für Filetransfer bzw. Folgeverarbeitung

FILE '[*ft* | *batch*] R=*remote-file* [L=*local-file*] [M=*mode*] [*authorisation*] [FP=*fpass*|?]
[*mvs-fileattr*] [*par1 par2*] '

FILE *str-var* — Alle Parameter stehen in einer Stringvariablen.

Mit diesem Format des FILE-Kommandos kann eine Vorbelegung für das Lesen und Schreiben mit Filetransfer bzw. Folgeverarbeitung erfolgen. Der Filetransfer kann entweder von einer Batch-Datei oder vom internen EDT-Filetransfer durchgeführt werden.

Filetransfer-Profil

ft — Name eines Filetransfer-Profils, das mit dem Menübefehl `Extras/Filetransfer...` (S. 136) definiert werden kann. Aus den Parametern des Kommandos und den Angaben aus dem Profil wird die Schnittstelle des internen EDT-Filetransfers erzeugt und der Filetransfer gestartet.

Sind Profile vorhanden, die in mehreren INI-Dateien mit dem gleichen Namen enthalten sind, kann durch Voranstellen der Dateinummer (1, 2 oder 3) die entsprechende INI-Datei angegeben werden.

Beispiel:

`1std` = Profil `STD` aus der privaten INI-Datei
`2std` = Profil `STD` aus der INI-Datei vom Windowsverzeichnis
`3std` = Profil `STD` aus der INI-Datei vom Ladeverzeichnis.

batch — Dateiname der Batchdatei. Die Extension ".BAT" muss hierbei nicht angegeben werden. Die Batch-Prozedur wird mit folgenden Parametern gestartet:

batch-file par1 par2 .....

Lokale Datei

local-file — Dateiname der zu erzeugenden Datei auf dem lokalen Rechner. Ist dieser Parameter nicht angegeben, wird eine temporäre Datei erzeugt, die nach dem Filetransfer gelöscht wird.

Entfernte Datei

remote-file — *bs2-file* | *lib*([*typ*/]*elem*[/*version*]) | */posix-file* | *mvs-file*

bs2-file — Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID. Es kann auch eine Generation aus einer Dateigenerationsgruppe in der BS2000-Syntax angegeben werden (z.B. `fgg(*0003)`).

lib — Name der PLAM-Bibliothek auf dem fernen Rechner.

typ — Typ des PLAM-Elements. Es sind nur die Angaben S, M, J, P und D zulässig. Wird kein Typ angegeben, so wird der Typ "S" als Standard benutzt.

elem — Name des PLAM-Elements. Delta-geführte Elemente können vom openFT-BS2000 nur gelesen werden.

version — Version des PLAM-Elements. Fehlt die Version, wird die höchste Version des Elements benutzt.

/posix-file — Beginnt der Dateinamen mit dem Zeichen "/", so wird die Datei automatisch von dem entsprechenden POSIX-Dateisystem gelesen. Es ist immer der Dateiname ab ROOT anzugeben. Beim Filetransfer mit FTP muss im Filetransfer-Profil der Typ FTP-POSIX ausgewählt werden.

mvs-file — *data-set* | *pds(member)* | '*hlq*.*data-set*' | '*hlq*.*pds(member)*'

Bei der Variante `read'profile r=mvsfile'` müssen die Hochkommas verdoppelt werden, weil der ganze Parameter bereits ein String in Hochkommas ist. Soweit alle Angaben in einer Stringvariablen stehen, dürfen die Hochkommas nicht verdoppelt werden. Soll z.B. eine Datei aus der FSTAT-Liste eingelesen werden, so muss lediglich der Inhalt aus der Spalte 1-256 in eine Stringvariable übertragen werden. Dann kann die Datei mit dem Befehl `read` *strvar* eingelesen werden.

Um die Angabe der doppelten Hochkommas zu umgehen, können auch folgende alternative Formate angegeben werden:

`mvs:`*hlq*.*data-set* | `mvs:`*hlq*.*pds(member)* oder
`$`*hlq*.*data-set* | `$`*hlq*.*pds(member)*

data-set — Dateiname auf dem fernen Rechner ohne `HLQ` (high level qualifier). Vom Host-System wird der Dataset-Name automatisch um den HLQ ergänzt.

pds(member)

Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended) ohne HLQ. Vom Host-System wird der Member-Name automatisch um den HLQ ergänzt.

'*hlq.data-set*' | mvs:*hlq.data-set* | $*hlq.data-set*

Vollqualifizierter Dateiname auf dem fernen Rechner. Aus diesem String wird für den Filetransfer der Dateiname '*hlq.data-set*' erzeugt.

'*hlq.pds(member)*' | mvs:*hlq.pds(member)* | $*hlq.pds(member)*

Vollqualifiziertes Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended). Aus diesem String wird für den Filetransfer der Dateiname '*hlq.pds(member)*' erzeugt.

Zugangsdaten

Die Zugangsdaten können im Filetransfer-Profil (`Extras/Filetransfer...` (S. 136)) definiert werden oder bei den Kommandos `FILE` (S. 192), `READ` (S. 226) und `WRITE` (S. 264) angegeben werden. Es ist auch möglich, einen Teile der Zugangsdaten im Profil und einen Teil bei den Kommandos anzugeben.

authorisation

[LH=*host*] [LU=*user*] [LA=*account*] [LP=*lpass*|?] oder TA=*ta*

host | ?

Host-Name für den Zugang zum entfernten Rechner. Der Hostname kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

user | ?

User-ID für den Zugang zum entfernten Rechner. Die User-ID kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

acount | ?

Abrechnungs-Nummer für den Zugang zum entfernten Rechner. Die Abrechnungs-Nummer kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

lpass | ?

Logon-Passwort für den Zugang zum entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *lpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

ta | ?

Transfer-Admission des FTAC-Profils für den Zugang zum entfernten Rechner. FTAC-Profile können mit der Software FTAC erstellt werden. Die Transfer-Admission muss mindestens 8 Stellen und darf höchstens 32 Stellen lang sein. Enthält der Name Blanks, muss er in Hochkommas eingegeben werden. Die Transfer-Admission kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

fpass|?

File-Passwort im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

Übertragungsmodus

mode

Transfer-Modus. Diese Angabe besteht aus den 2 Information:

a) Übertragungsmodus Text/Binär:

Text-Modus bedeutet, dass die Daten mit der Standard-Translatetabelle bzw. der Translatetabelle, die im FT-Profil angegeben ist, übersetzt werden (ANSI <--> EBCDIC/ASCII).

Im **Binär-Modus** werden die Daten transparent ohne Übersetzung übertragen. Es sind drei verschiedene Modi zu unterscheiden:

a) Binär satzstrukturiert für openFT-BS2000 (M=B)
b) Binär undefiniert für openFT-BS2000 (M=U)
c) Binär für FTP (M=B)

Nach dem Einlesen der Datei in den Arbeitsbereich wird die Codierung automatisch auf den entsprechenden Code umgeschaltet (wie Kommando `CODE EBCDIC` (S. 181)).

Gilt nur für openFT: In der lokalen Datei (Name wird entweder vom EDT vergeben oder Name aus "L=*file*") werden die Daten mit 4 Byte Satzlängenfeld ohne Satzende-Kennzeichen gespeichert. Dadurch ist es möglich, dass in einem Satz auch das BS2000-Satzende-Kennzeichen (X'15') oder das Unix-Satzende-Kennzeichen (X'0A') vorkommen kann.

Im Filetransfer-Profil kann der Übertragungsmodus ebenfalls eingestellt werden. Wird beim Kommando `READ` nichts angegeben, so gilt die Einstellung des FT-Profils. Wird beim Kommando `WRITE` nichts angegeben, so gilt entweder der Übertragungsmodus des vorhergehenden `READ` oder die Einstellung des FT-Profils, falls eine lokale oder neue Datei ohne vorhergehenden `READ` auf den fernen Rechner übertragen wird.

b) Feste/Variable Satzlänge. Soll eine Datei mit fester Satzlänge erzeugt werden, müssen alle Zeilen im Arbeitsbereich exakt die richtige Länge haben.

Folgende Werte sind zulässig bei **openFT_BS2-Profilen**:

T Text-Modus (Standardeinstellung). Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Beim Schreiben ohne weitere Formatangabe wird die ferne Datei allerdings mit dem Format "Variable Satzlänge" erzeugt.

TF*nnn*
Text-Modus feste Satzlänge. Die Datei wird mit RECFORM=F und RECSIZE=*nnn* gelesen bzw. erstellt. Wurde diese Option beim Lesen angegeben, so wird beim Schreiben ohne Parameter automatisch das gleiche Dateiformat erzeugt.

TV Text-Modus, variable Satzlänge (gilt nur für openFT-BS2000). Diese Angabe ist nur beim Schreiben notwendig, falls nach dem Lesen einer Datei mit fester Satzlänge eine Datei mit variabler Satzlänge erzeugt werden soll.

B Binär-Modus. Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Die Daten werden im Arbeitsbereich im EBCDIC-Code satzstrukturiert dargestellt. Ist die Datei im Binär-Modus gelesen worden und wird sie mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, wird sie automatisch wieder im Binär-Modus transferiert, allerdings immer mit variabler Satzlänge.

BF*nnn*
Binär-Modus feste Satzlänge. Die Datei wird wie bei M=B binär mit RECFORM=F und RECSIZE=*nnn* gelesen bzw. erstellt. Die Daten werden im Arbeitsbereich im EBCDIC-Code satzstrukturiert dargestellt. Wurde diese Option beim Lesen angegeben, so wird beim Schreiben ohne Parameter automatisch das gleiche Dateiformat erzeugt.

BV Binär-Modus, variable Satzlänge. Diese Angabe ist nur beim Schreiben notwendig, falls nach dem Lesen einer Datei mit fester Satzlänge eine Datei mit variabler Satzlänge erzeugt werden soll.

U Binär-Modus undefined (ohne Satzstruktur). Mit dieser Option können Dateien ohne Satzstruktur gelesen und geschrieben werden. Dieser Modus ist z.B. notwendig, wenn eine POSIX-Datei aus einem ASCII-Filesystem gelesen werden soll. Nach dem Einlesen kann der Arbeitsbereich mit dem Kommando `REFORMAT UNIX` und `CODE ASCII` satzstrukturiert dargestellt werden. Vor dem Schreiben muss der Arbeitsbereich mit dem Kommando `UNFORMAT` wieder in das Binärformat umgewandelt werden.

Folgende Werte sind zulässig bei **FTP-Profilen**:

T Text-Modus (Standardeinstellung). Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Dateien, die im BS2000 mit fester Satzlänge gespeichert sind, werden beim Schreiben in die gleiche Datei automatisch im richtigen Format erzeugt. Neue Dateien können nur mit variabler Satzlänge erzeugt werden.

B Binär-Modus. Die Daten werden transparent ohne Übersetzung übertragen und im Arbeitsbereich ohne Satzstruktur dargestellt. Soweit es sich um BS2000-Dateien mit variabler Satzlänge (RECFORM=V) oder Unix/Windows-Textdateien handelt, enthalten die Daten folgende Satzende-Kennzeichen: Windows=X'0D0A', Unix = X'0A' und BS2000 = X'15'. Soweit es sich um BS2000-Dateien mit fester Satzlänge (RECFORM=F) handelt, enthalten die Daten kein Satzende-Kennzeichen. Die Codierung kann im Profil schon auf den Wert ANSI, ASCIIUNIX oder EBCDIC7 eingestellt werden.

Nach dem Einlesen kann die Satzstruktur des fernen Rechners mit dem Kommando `REFORMAT` (S. **234**) wieder hergestellt werden.

Beispiel:

`REFORMAT BS2`	(BS2000, variable Satzlänge)
`REFORMAT RS80`	(BS2000, feste Satzlänge 80)
`REFORMAT UNIX`	(Unix-Datei)
`REFORMAT DOS`	(Windows-Datei)

Vor dem binären Schreiben eines solchen Arbeitsbereichs muss mit dem Kommando `UNFORMAT` wieder das Binärformat erzeugt werden.

Wird der Arbeitsbereich mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, werden die Daten automatisch wieder im Binär-Modus transferiert und die Datei auf dem fernen Rechner mit den bestehenden Dateiattributen erstellt.

Datei-Passwort

fpass|?

Passwort für den Zugriffsschutz der Datei auf dem entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort in einer Dialogbox unsichtbar angefordert.

Datei-Attribute für MVS

mvs-fileattr

RECFM=*recfm* LRECL=*lrecl* [BLKSIZE=*blksize*]

recfm

Record format: F | FB | FBA | V | VB | VBA | U

lrecl

Record length: Satzlänge, max. 32767

blksize

Block size: Blocklänge, max. 32767. Fehlt die Angabe der Blocklänge, wird vom MVS-System ein optimaler Wert ermittelt. In der Regel kann deshalb dieser Parameter entfallen.

Parameter für die Folgeverarbeitung

par1 par2

Beliebige Parameter, die als Parameter beim Aufruf an die Batch-Prozedur bzw. die Prozedur für die Folgeverarbeitung übergeben werden.

Dialogbox für die Passwort-Eingabe

Das Datei- bzw. Logon-Passwort ist in der Form cccc oder x'xxxxxxxx' anzugeben. Die Hochkommas müssen in der Dialogbox nicht verdoppelt werden.

Dateinamen und Attribute in Arbeitsbereich anzeigen

FSTAT '[*drive*:] [\] [*path*\] *file*' [TO *ln*[*(inc)*]] [F|S|L|M] [R|**SUB**DIR] [ADS|ADSO] [**,** NF]

FSTAT '[*drive*:] [\] [*path*\] *zipfile*' **,A** [**T**O *ln*[*(inc)*]] [F | L] [**,** NF]

Die nach der Syntax des MS-DOS-Kommandos DIR ermittelten Dateien bzw. die Elemente eines ZIP-Archivs werden in den aktuellen Arbeitsbereich ab der Zeile *ln* oder in den Arbeitsbereich 9 geschrieben. Verzeichnisse werden in der Dateienliste farblich hervorgehoben. Durch besondere Eingaben in der Markierungsspalte (S. **204**) können die Dateien aus der Dateienliste des FSTAT-Kommandos in Arbeitsbereiche eingelesen oder gelöscht werden.

Nach Ausführung des FSTAT-Kommandos wird der Arbeitsbereich angezeigt, in dem die Liste der Dateien übertragen wurde (Arbeitsbereich 9 oder aktueller Arbeitsbereich). Dies gilt auch, falls das Kommando in einer EDT-Prozedur aufgerufen wurde. Das Anzeigen des Ergebnis-Arbeitsbereichs kann nur mit der Option NF verhindert werden.

drive

Laufwerk.

path

Pfad.

file

Dateiname bzw. Teil eines Dateinamens. Die Angabe *file* kann nach MS-DOS Syntax die Zeichen "*" und "?" enthalten, z.B. *.* oder *.INI oder *.PR?.

zipfile

Dateiname eines ZIP-Archives. Die im Archiv enthaltenen Elemente werden in den aktuellen Arbeitsbereich ab der Zeile *ln* oder in den Arbeitsbereich 9 geschrieben, falls die Option A (Archiv) angegeben wird.

ln Zeilennummer, ab der die Datei-Informationen in den aktuellen Arbeitsbereich geschrieben werden. Ist *ln* nicht angegeben, werden die Informationen in den Arbeitsbereich 9 geschrieben, der vorher gelöscht wird.

inc Schrittweite, aus der die auf *ln* folgenden Zeilennummern gebildet werden sollen. Wird *inc* nicht angegeben, so wird die implizit gegebene Schrittweite von *ln* verwendet (z.B. Schrittweite 0.01 bei der Zeilenangabe 10.50).

A Archiv: im aktuellen Arbeitsbereich ab der Zeile *ln* oder in den Arbeitsbereich werden die Elemente eines ZIP-Archivs aufgelistet.

ADS Alternate Data Streams (ADS): Neben "normalen Dateien" werden auch ADS aufgelistet. ADS werden nur von Windows NT/2000 mit NTFS-Dateisystem unterstützt. Die ADS sind für den Explorer "unsichtbar".

ADSO Alternate Data Streams (ADS) only: Es werden nur Dateien mit ADS aufgelistet.

F | M | L | S Mit dieser Option wird der Umfang der Ausgabe gesteuert.

F (Full)	mit Laufwerk und Pfad und Dateiattributen
L (Long)	mit Laufwerk und Pfad ohne Dateiattribute
M (Medium)	ohne Laufwerk und Pfad mit Dateiattributen
S (Short)	ohne Laufwerk und Pfad ohne Dateiattribute

Wurde keine der Optionen M, F, L oder S angegeben, so gilt die Option F. Zusätzlich wird im Dialog-Modus die Spalte mit den Dateinamen von 256 auf die Länge des längsten Dateinamens + 5 verkürzt, damit die folgenden Spalten in der Regel sichtbar sind. Im Prozedur-Modus (Kommando DO oder INPUT bzw. Schalter -i beim Laden des EDT) wird die Spalte nicht verkürzt.

F Full. Es werden folgende Informationen ausgegeben:

Spalte 001-256 vollständiger Dateiname einschl. des Profilnamens bei Remote-Dateien, z.B.

C:\test\testdat.txt	(Windows)
profil r=/home/test/testdat.txt	(Unix)
profil r=:catid:$TEST.TESTDAT	(BS2000)
profil r= ' hlq.dataset '	(MVS)
profil r= ' hlq.pds(member) '	(MVS-PDS)

Spalte 257-267 Dateigröße in Bytes

bei BS2000:	Pam-Pages,
bei MVS Datasets:	Tracks,
bei MVS PO-Elementen	Zeilen

Spalte 270-274 Attribute;
Spalte 270 - H = Hidden;
Spalte 271 - S = Systemdatei;
Spalte 272 - R = Read only;
Spalte 273 - A = Archive (Datei ist archiviert);
Spalte 274 - D = Directory (Diese Einträge sind markiert);
Spalte 277-286 Datum der letzten Änderung;
bei MVS Datasets: Datum letzter Zugriff
Spalte 289-296 Uhrzeit der letzten Änderung.
bei MVS-Dateien leer

Gilt nur für Unix-Dateienliste:

Spalte 299-308 Attribute `rwxrwxrwx`
Spalte 310-316 Anzahl der Links
Spalte 318-325 Eigentümer der Datei
Spalte 327-334 Gruppe der Datei

Gilt nur für BS2000-Dateienliste mit FTP:

Spalte 299-308	Cat-ID
Spalte 310-319	User-ID
Spalte 321-368	Dateiname (ohne Cat-ID und User-ID)

Gilt nur für BS2000-Dateienliste mit openFT:

Spalte 299	`R:` Read-Passwort da
Spalte 300	`W:` Write-Passwort da
Spalte 301	`X:` Execute-Passwort da
Spalte 302	`S:` Share
Spalte 304-308	FCBTYPE SAM PLAM PAM' SAM *typ:* Typ bei LIB-Elementen
Spalte 310	Recform V / F / U
Spalte 312-316	Recsize
Spalte 318-325	CCSNAME
Spalte 327-334	Owner

Gilt nur für MVS-Dateienliste (data-set):

Spalte 300-307	Volume
Spalte 309-312	Unit
Spalte 314-318	Anzahl Extensions
Spalte 320-324	Recform
Spalte 326-331	Lrecl (Satzlänge)
Spalte 333-340	Blocklänge
Spalte 342-346	DSORG (Dataset-Organisation)

Gilt nur für MVS-Dateienliste (PDS-member):

Spalte 299-308	Datum der Dateierzeugung
Spalte 310-314	VV.MM
Spalte 316-325	Anzahl der Zeilen bei Dateierzeugung
Spalte 327-331	Mod
Spalte 333-340	Benutzer

Gilt nur für ZIP-Archive:

Spalte 298-307	komprimierte Dateigröße

Die Option F ist nur möglich, wenn als max. Satzlänge im Menü `Optionen / verschiedene Optionen` (S. **127**) mindestens 512 eingestellt ist.

M — wie F, die erste Spalte enthält jedoch nur den Dateinamen ohne Laufwerk und Pfad.

L — Long. Pro Datei werden ab Spalte 1 das Laufwerk, der Pfadname und der Dateiname ausgegeben (z.B. `C:\edt\datei.dat`).

S — Short. Pro Datei werden ab Spalte 1 nur der Dateiname und die Erweiterung (z.B. `datei.dat`) ausgegeben.

R oder **SUB**DIR

Neben dem angegebenen Verzeichnis werden auch alle Unterverzeichnisse rekursiv durchsucht, d.h. mit dem Kommando
`FSTAT 'C:\*.BAT'R` erhalten Sie z.B. alle BAT-Dateien aller Verzeichnisse des Laufwerks C. Ohne die Option R wird nur das Root-Verzeichnis durchsucht.

NF

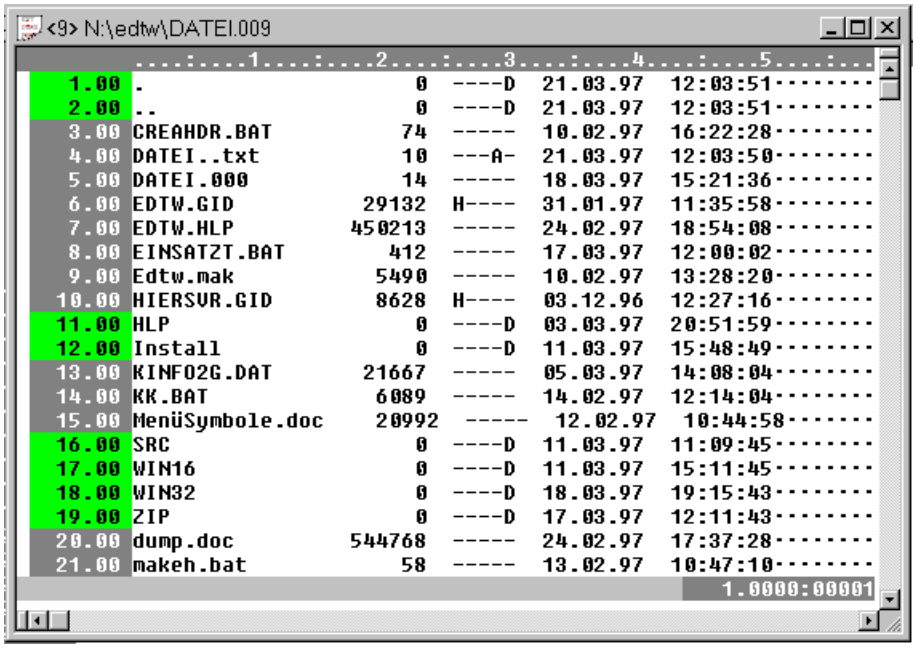
No Focus: Nach der Erstellung der Ergebnisdaten wird nicht der Arbeitsbereich 9 bzw. die angegebene Zeile im aktuellen Arbeitsbereich angezeigt. Der Fokus bleibt unverändert. Diese Option ist insbesondere für Prozeduren gedacht.

Hinweis:

Die FSTAT-Liste kann durch Eingaben in der Markierungsspalte und in der Kommandozeile weiterverarbeitet werden (Einlesen von Dateien und FSTAT mit Unterverzeichnissen, siehe S. 204).

Beispiel:

```
fs '\*.*'
```



FSTAT '[*ft* | *batch*] R=*remote-file* [*authorisation*] [FP=*fpass*|?] [*par1 par2 ....*]' [TO *ln*[*(inc)*]] [F|S|L|M]

Eine FSTAT- oder Bibliotheks-Liste von einem anderen Rechner wird in den aktuellen Arbeitsbereich ab der Zeile *ln* oder in den Arbeitsbereich 9 eingelesen. Der notwendige Filetransfer wird entweder über die Batch-Datei *batch-file* oder über den internen EDT-Filetransfer (Verbindung zu openFT oder FTP) aufgerufen.

Die Dateien bzw. die Elemente eines MVS PDS-Archivs werden in den aktuellen Arbeitsbereich ab der Zeile *ln* oder in den Arbeitsbereich 9 geschrieben. MVS PDS-Archive werden in der Dateienliste farblich hervorgehoben. Durch besondere Eingaben in der Markierungsspalte (S. 204) können die Dateien aus der Dateienliste des FSTAT-Kommandos in Arbeitsbereiche eingelesen oder gelöscht werden.

Nach Ausführung des FSTAT-Kommandos wird der Arbeitsbereich angezeigt, in dem die Liste der Dateien übertragen wurde (Arbeitsbereich 9 oder

aktueller Arbeitsbereich). Dies gilt auch, falls das Kommando in einer EDT-Prozedur aufgerufen wurde. Das Anzeigen des Ergebnis-Arbeitsbereichs kann nur mit der Option NF verhindert werden.

Filetransfer-Profil

ft — Name eines Filetransfer-Profils, das mit dem Menübefehl `Extras/Filetransfer...` (S. 182) definiert werden kann. Aus den Parametern des Kommandos und den Angaben aus dem Profil wird die Schnittstelle des internen EDT-Filetransfers erzeugt und der Filetransfer gestartet.

Sind Profile vorhanden, die in mehreren INI-Dateien mit dem gleichen Namen enthalten sind, kann durch Voranstellen der Dateinummer (1, 2 oder 3) die entsprechende INI-Datei angegeben werden.

Beispiel:

`1std` = Profil `STD` aus der privaten INI-Datei
`2std` = Profil `STD` aus der INI-Datei vom Windowsverzeichnis
`3std` = Profil `STD` aus der INI-Datei vom Ladeverzeichnis.

batch — Dateiname der Batchdatei. Die Extension ".BAT" muss hierbei nicht angegeben werden. Die Batch-Prozedur wird mit folgenden Parametern gestartet:

batch-file F *par1_par2 .....*

Die Angabe von "F" als erster Parameter bedeutet, dass die Batchdatei von dem Kommando FSTAT aufgerufen wurde. Der Name der lokalen Datei *local-file* wird von EDT in der Form edt*sssss*.lc*n* erzeugt (*sssss* = aktuelle Sekunde, *n* = Nummer der Arbeitsdatei).
Standardmäßig wird die lokale Datei in dem Verzeichnis angelegt, in dem der EDT aufgerufen wird.

Entfernte Datei

remote-file — *bs2-file* | *lib*(*) | *lib*(*typ*/*elem*[/*version*]) | */posix-file* | *mvs-file*

bs2-file — "*" für alle Dateien einer User-ID, Dateiname oder teilqualifizierter Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID. Evtl. vorhandene PLAM-Bibliotheken werden als Verzeichnis gekennzeichnet und können mit einem Folge-FSTAT angezeigt werden.

lib — Name der PLAM-Bibliothek auf dem fernen Rechner. Sind in einer Bibliothek Elemente mit mehreren Versionen enthalten, wird immer das Element mit der höchsten Version angezeigt und ein zusätzlicher Listeneintrag mit der Versions-Nummer "*old" für einen Folge-FSTAT, bei dem dann alle älteren Versionen angezeigt werden.

lib(*) — Alle Elementtypen und Elemente einer PLAM-Bibliothek.

typ — Typ der PLAM-Elemente.

elem — Name des PLAM-Elements oder "*" für alle Elemente.

version — Version des PLAM-Elements oder "*" für alle Elemente. Fehlt die Version, wird die höchste Version des Elements benutzt.

/posix-file — Vollständiger POSIX-Dateiname oder POSIX-Verzeichnis/*. Es ist immer der Dateiname ab ROOT anzugeben.

mvs-file — *data-set* | *pds(member)* | '*hlq*.*data-set*' | '*hlq*.*pds(member)*'

Bei der Variante `read'profile r=`*mvsfile*`'` müssen die Hochkommas verdoppelt werden, weil der ganze Parameter bereits ein String in Hochkommas ist. Soweit alle Angaben in einer Stringvariablen stehen, dürfen die Hochkommas nicht verdoppelt werden. Soll z.B. eine Datei aus der FSTAT-Liste eingelesen werden, so muss lediglich der Inhalt aus der Spalte 1-256 in eine Stringvariable übertragen werden. Dann kann die Datei mit dem Befehl `read` *strvar* eingelesen werden.

Um die Angabe der doppelten Hochkommas zu umgehen, können auch folgende alternative Formate angegeben werden:

`mvs:`*hlq*`.`*data-set* | `mvs:`*hlq*`.`*pds(member)* oder
`$`*hlq*`.`*data-set* | `$`*hlq*`.`*pds(member)*

data-set Dateiname auf dem fernen Rechner ohne `HLQ` (high level qualifier). Vom Host-System wird der Dataset-Name automatisch um den HLQ ergänzt.

pds(member) Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended) ohne HLQ. Vom Host-System wird der Member-Name automatisch um den HLQ ergänzt.

`'`*hlq*`.`*data-set*`'` | `mvs:`*hlq*`.`*data-set* | `$`*hlq*`.`*data-set*

Vollqualifizierter Dateiname auf dem fernen Rechner. Aus diesem String wird für den Filetransfer der Dateiname `'`*hlq.data-set*`'` erzeugt.

`'`*hlq*`.`*pds(member)*`'` | `mvs:`*hlq*`.`*pds(member)* | `$`*hlq*`.`*pds(member)*

Member eines vollqualifizierten PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended). Aus diesem String wird für den Filetransfer der Dateiname `'`*hlq*`.`*pds(member)*`'` erzeugt.

Zugangsdaten

Die Zugangsdaten können im Filetransfer-Profil (`Extras/Filetransfer....` (S. **136**)) definiert werden oder bei den Kommandos `FILE` (S. **192**), `READ` (S. **226**) und `WRITE` (S. **264**) angegeben werden. Es ist auch möglich einen Teile der Zugangsdaten im Profil und einen Teil bei den Kommandos anzugeben.

authorisation [LH=*host*] [LU=*user*] [LA=*account*] [LP=*lpass*|?] oder TA=*ta*

host | ? Host-Name für den Zugang zum entfernten Rechner. Der Hostname kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

user | ? User-ID für den Zugang zum entfernten Rechner. Die User-ID kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

acount | ? Abrechnungs-Nummer für den Zugang zum entfernten Rechner. Die Abrechnungs-Nummer kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

lpass | ?

Logon-Passwort für den Zugang zum entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *lpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

ta | ?

Transfer-Admission des FTAC-Profils für den Zugang zum entfernten Rechner. FTAC-Profile können mit der BS2000-Software FTAC-BS2000 erstellt werden. Die Transfer-Admission muss mindestens 8 Stellen und darf höchstens 32 Stellen lang sein. Enthält der Name Blanks, muss er in Hochkommas eingegeben werden. (gilt nur für openFT-BS2000). Die Transfer-Admission kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

fpass|?

File-Passwort im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

par1 par2

Beliebige Parameter, die beim Aufruf an die Batch-Prozedur übergeben werden.

F | M | L | S

Mit dieser Option wird der Umfang der Ausgabe gesteuert wie bei dem lokalen FSTAT (S. **197**).

Hinweis:

Mit Hilfe der Batch-Datei kann ein beliebiger Arbeitsgang zur Bereitstellung einer Liste von Dateinamen ausgeführt werden.
Wie oben dargestellt, kann eine FSTAT-Liste oder das Inhaltsverzeichnis einer Bibliothek mit einem Filetransfer Programm von einem BS2000-Host abgeholt werden. Es ist aber auch möglich, die Liste aller, in einem ARJ-/ZIP-Archiv enthaltenen Dateien zu erzeugen und im EDT anzuzeigen.

Beispiele für Batch-Prozeduren zum Holen einer FSTAT- oder Bibliotheksliste von einem BS2000-Host sind in den ausgelieferten Dateien MPS.BAT (Filetransfer der Firma MPS) und LOG.BAT (Filetransfer der Firma Logics) enthalten.

Die FSTAT-Liste kann durch Eingaben in der Markierungsspalte und in der Kommandozeile weiterverarbeitet werden (Einlesen von Dateien und FSTAT mit Unterverzeichnissen, siehe S. **204**).

Mit dem Kommando `WRITE` ohne Parameter wird die bearbeitete Datei wieder in das entfernte System zurück übertragen.

Beispiele:

```
fstat'bs2000 r=$userid.'
fstat'bs2000 r=$userid.src.'
fstat'bs2000 r=*'
fstat'bs2000 r=lib(*)'
fstat'bs2000 r=lib(s/*)'
fstat'bs2000 r=lib(s/elem/*)'
fstat'bs2000 r=/*)'
fstat'bs2000 r=/home/test/*'
fstat'linux r=/home/test/src*'
fstat'mvs r=''test.src*''' to 1 f
fstat'mvs r=mvs:test.src*' m
fstat'mvs r=$test.src*' to1 f
```

Weiterverarbeitung der FSTAT-Liste

In der Markierungsspalte können die folgenden Aktionen eingegeben werden:

0 – 9 oder F

Datei: Die markierte Datei wird in den angegebenen Arbeitsbereich 0-9 bzw. in den nächsten freien Arbeitsbereich (bei F) eingelesen. Enthält der Arbeitsbereich geänderte Daten, so wird gefragt, ob der Arbeitsbereich überschrieben werden soll. Nach dem Einlesen aller Daten wird die zuletzt eingelesene Datei angezeigt.

Windows-Verzeichnis oder Unix-Verzeichnis: Für das Verzeichnis wird das Kommando `FSTAT'\`*dir*`\*.*'` bzw. `FSTAT'/`*dir*`/*.*'` aufgerufen und das Ergebnis in den angegebenen Arbeitsbereich 0-9 bzw. in den nächsten freien Arbeitsbereich (bei F) übertragen.

BS2000 PLAM-Bibliothek: Für die Bibliothek wird das Kommando `FSTAT'profil r=`*plamlib*`(*)'` aufgerufen und das Ergebnis in den angegebenen Arbeitsbereich 0-9 bzw. in den nächsten freien Arbeitsbereich (bei F) übertragen.

BS2000 Element einer PLAM-Bibliothek mit mehreren Versionen: Für die Bibliothek wird das Kommando `FSTAT'profil r=`*plamlib*`(`*typ/elem/*`*`)'` aufgerufen und das Ergebnis in den angegebenen Arbeitsbereich 0-9 bzw. in den nächsten freien Arbeitsbereich (bei F) übertragen.

ZIP-Archiv: Für das Archiv wird das Kommando `FSTAT'`*zip-archiv*`',a` aufgerufen und das Ergebnis in den angegebenen Arbeitsbereich 0-9 bzw. in den nächsten freien Arbeitsbereich (bei F) übertragen.

Bei einer MVS PDS-Lib wird das Kommando `FSTAT'profil r=''polib(*)'''` aufgerufen und das Ergebnis in den angegebenen Arbeitsbereich 0-9 bzw. in den nächsten freien Arbeitsbereich (bei F) übertragen.

A

Append. Diese Angabe ist nur bei einem Verzeichnis, ZIP-Archiv, einer BS2000 PLAM-Bibliothek (nur openFT) oder einem MVS PDS-Archiv zulässig. Für das Verzeichnis, die Bibliothek bzw. das Archiv wird das Kommando `FSTAT` aufgerufen und das Ergebnis an die bestehende Liste angehängt.

N

New Params. Diese Angabe ist nur bei einem Verzeichnis, ZIP-Archiv, einer BS2000 PLAM-Bibliothek (nur openFT) oder einem MVS PDS-Archiv zulässig. Für das Verzeichnis, die Bibliothek bzw. das Archiv wird das Kommando `FSTAT` aufgerufen und das Ergebnis anstelle der bisherigen Liste angezeigt.

E Erase. Diese Angabe ist nur bei einer lokalen Datei oder einer mit einem FTP-Profil eingelesenen Datei zulässig. Die Datei wird sowohl aus der FSTAT-Liste als auch auf dem Datenträger gelöscht.

Die folgenden Markierungsbefehle wirken wie in normalen Arbeitsbereichen:

D Delete. Löscht diese Zeile aus der FSTAT-Liste.

K Die markierte Zeile wird in der Länge des Dateinamens in die Kommandozeile übernommen. Die Zeile kann auch länger als der sichtbare Teil der Kommandozeile sein.

X Gibt die Zeile zum Überschreiben frei.

Kommandozeile

In der Kommandozeile können die folgenden Kommandos eingegeben werden:

READ [*rng*] (*n*) Alle Dateien, deren Dateinamen in dem Zeilenbereich *rng* stehen, werden hintereinander in den Arbeitsbereich *n* eingelesen. Vor der ersten Zeile und nach der letzten Zeile einer Datei wird jeweils eine Zeile mit dem Dateinamen, der Größe der Datei und dem Datum der letzten Änderung eingefügt. Diese Zeile beginnt mit dem String "`$$$BEG###>>>--->>>---` " bzw. "`$$$END###>>>--->>>---` ". Ohne Bereichsangabe werden alle Dateien eingelesen. Siehe dazu auch die Ausführungen zum Kommando READ (S. 224).

READ *zlnr* | *linevar* (*n*) Die Datei, deren Dateiname in der Zeile *zlnr* oder *linevar* steht, wird in den Arbeitsbereich *n* eingelesen.

Zu den Eingaben für die Markierungsspalte bei normalen Arbeitsbereichen siehe S. 175

EDT beenden

HALT [**N**OUPD] [**IS**AVE] [**IN**OSAVE] [*n* | *int-var*]
Das Kommando beendet den EDTW.

NOUPD Die Option bewirkt im Dialogmodus, dass die Dialogbox "Änderung in Datei speichern" nicht ausgegeben wird. Auch bei Änderungen der Daten wird das Programm sofort beendet. Im Prozedurmodus wird diese Option automatisch als Standard benutzt.

ISAVE Die Option bewirkt, dass evtl. geänderte Einstellungen, unabhängig von den Einstellungen im Menü `Optionen / Sichern` (S. 131), in der INI-Datei gespeichert werden.

INOSAVE Die Option bewirkt, dass evtl. geänderte Einstellungen, unabhängig von den Einstellungen im Menü `Optionen / Sichern` (S. 131), **nicht** in der INI-Datei gespeichert werden.

n | *int-var* Falls das Programm EDT in einer Batch-Datei aufgerufen wurde, kann hier ein Errorlevel gesetzt werden. Es kann ein beliebiger numerischer Wert zwischen 0 und 2.147.483.648 direkt oder in Form einer Integer-Variablen angegeben werden.

In der Batch-Datei kann der Errorlevel mit dem Kommando "`IF %ERRORLEVEL% EQ 0 goto ok`" abgefragt werden.

Hexadezimale Darstellung

HEX ON | 2 | 4 | OFF — Hexadezimale Darstellung der Daten ein-/ausschalten.

ON oder 2 — 2-zeilige Anzeige. Die Daten werden hexadezimal angezeigt, d.h. nach jeder Zeile im Character-Format wird für jedes Halbbyte eine zusätzliche Zeile angezeigt. Auch bei eingeschaltetem Long-Modus wird nur eine Zeile pro Datensatz angezeigt. Änderungen können sowohl in der Character-Zeile als auch in den Hexa-Zeilen durchgeführt werden. Die Änderungen werden sofort sowohl im Character-Format als auch hexadezimal dargestellt.

4 — 4-zeilige Anzeige für UNICODE-Arbeitsbereiche. Nach jeder Zeile im Character-Format wird für jedes der 4 Halbbytes eines UNICODE-Zeichens eine zusätzliche Zeile angezeigt. Diese Variante ist nur erlaubt, wenn es sich um einen UNICODE Arbeitsbereich handelt.

OFF — Die Daten werden alphanumerisch dargestellt.

Horizontal-Tabulator definieren

HT [[*c*] [, *nnn*]]] — Horizontal-Tabulator. Definition eines Zeichens für den Horizontal-Tabulator. Dieses Zeichen wird in das Tabulatorzeichen x'09' umgesetzt. Die Tabulator-Taste bewirkt im EDT die Positionierung auf das nächste Eingabefeld (nächste Zeile oder Kommandozeile). Deshalb muss dieses Zeichen eingegeben werden, wenn das Tab-Zeichen (x'09') erstellt werden soll.

Mit der HT-Funktion wird die Wirkungsweise der Tabulatortaste nachgebildet. Werden unterschiedliche Schrittweiten benötigt, so muss hierzu das Kommando `TABS` verwendet werden. Der Horizontal-Tabulator wird nicht unterstützt, falls für den aktuellen Arbeitsbereich ein EBCDIC-Code eingestellt ist.

c — Tabulator Ersatzzeichen.

n — Schrittweite in Zeichen. Bei fehlender Angabe wird die Schrittweite mit 8 angenommen.
HT ohne Parameter: Die Tabulator-Umsetzung wird ausgeschaltet.

Anzeige der Zeilennummern ein/aus

INDEX ON| 6 | 8 | LONG | OFF

Im EDT-Bildschirm werden die Zeilennummern angezeigt / nicht angezeigt.

ON oder 6 — Die 6-stellige Zeilennummer wird angezeigt.

LONG oder 8 — Die 8-Stellige Zeilennummer wird angezeigt.

OFF — Es wird keine Zeilennummer angezeigt.

Kommandodatei ausführen

INPUT '*file*' [(*params*)] [PRINT]

Starten einer Prozedur-Datei. In der Prozedurdatei können alle EDT-Kommandos einschließlich der Kommandos der Prozedursprache angegeben werden.

Die Prozedurdatei wird zur Verarbeitung in den internen Arbeitsbereich 34 eingelesen und mit dem Kommando DO aufgerufen. Dadurch ist es möglich, dass in INPUT-Prozeduren wieder weitere INPUT-Kommandos vorkommen können. Außerdem können auch in der Ebene 0 die Kommandos IF und GOTO verwendet werden.

Nach dem Ende einer Prozedur wird wieder der Arbeitsbereich angezeigt, von dem aus die Prozedur gestartet wurde bzw. der Arbeitsbereich, der evtl. mit dem Kommando SETF (S. **246**) in der Prozedur aktiviert wurde.

file — Name der Prozedurdatei. Ist kein Pfadname angegeben, wird die Datei in folgenden Pfaden gesucht:

1. aktuelles Verzeichnis
2. Installationsverzeichnis von EDTW.EXE
3. alle Pfade der Umgebungsvariablen PATH
4. alle Pfade der Umgebungsvariablen EDTPATHI

Die Umgebungsvariable EDTPATHI ist aufgebaut wie die Variable PATH (*path* [**;** *path***;** *path*.....]).

params — Parameter für die Prozedurdatei. Es können bis zu 32 Parameter, jeweils durch Komma getrennt, für die Prozedurdatei angegeben werden. Die Parameter müssen in der PARAMS–Anweisung (erste Zeile in der Prozedurdatei) definiert werden.

PRINT — Anzeigen aller Kommandos am Bildschirm.

Zeilennummern und Arbeitsbereich Status ausgeben

LIMITS [**S**IZE] — Zeilennummern ausgeben. In einem Fenster werden folgende Informationen für den aktuellen Arbeitsbereich angezeigt:

- die Anzahl der Zeilen
- die niedrigste und die höchste Zeilennummer
- aktuelle Zeilennummer und Schrittweite
- die Code-Variante für die Darstellung
- die Struktur (Satz-Format DOS / Satz-Format UNIX / Satz-Format MIXED / Binär), siehe auch Kommando REFORMAT (S. **234**), ERS (S. **191**) und Thema Dateistruktur (S. **40**).

SIZE — Die Gesamtanzahl der Bytes wird ermittelt und als zusätzlicher Wert ausgegeben. Die Angabe dieses Parameters kann bei sehr großen Arbeitsbereichen eine gewisse Zeit beanspruchen.

Mit dem Kommando PROC (ohne Operanden) wird angezeigt, welche Arbeitsbereiche belegt und welche Arbeitsbereiche frei sind.

Zeilenbereich ausdrucken

LIST [*rngcol*|*str-var*[*-str-var*]] [N] [H] [P] [X] [C] [A] [S] [L] [F]

Drucken eines Zeilenbereichs. Standardmäßig wird auf jeder Seite eine Kopfzeile mit Dateinamen, Arbeitsbereich, Datum, Uhrzeit und Seitennummer ausgedruckt. Vor jeder Zeile wird eine Zeilennummer ausgegeben. Mit den Optionen H (Header) und N (Nummer) können diese Informationen unterdrückt werden.

rngcol — Zeilen- und Spaltenbereich. Bei fehlender Angabe wird der gesamte Arbeitsbereich ausgedruckt.

str-var — String-Variable.

N — Die Zeilennummern werden unterdrückt.

H — Header: Die Kopfzeile wird nicht ausgedruckt.

P — Preview: Es wird die Druckvorschau angezeigt.

X — Der Ausdruck erfolgt in hexadezimaler Form.

C — Das erste Zeichen steuert das Ausdrucken, wird selbst aber nicht ausgedruckt. Die Optionen X, S, H und F werden ignoriert. Die Optionen N und H werden automatisch eingeschaltet.

Inhalt des ersten Zeichens Siemens-Steuerzeichen in EBCDIC:

X'C1'	Seitenvorschub vor dem Drucken
X'81'	Seitenvorschub nach dem Drucken
X'40' bis X'4F':	0 bis 15 Zeilen vor dem Drucken
X'00' bis X'0F':	0 bis 15 Zeilen nach dem Drucken

X'00' überdruckt die vorhergehende Zeile.

Falls es sich nicht um einen EBCDIC-Arbeitsbereich handelt, wird für die Auswertung des Steuerzeichens das 1. Byte nach EBCDIC konvertiert.

A — Das erste Zeichen steuert das Ausdrucken, wird selbst aber nicht ausgedruckt. Die Optionen X, S, H und F werden ignoriert. Die Optionen N und H werden automatisch eingeschaltet.

Inhalt des ersten Zeichens ASA-Steuerzeichen:

1	Seitenvorschub vor dem Drucken
blank	1 Zeile Vorschub vor dem Drucken
0	2 Zeilen Vorschub vor dem Drucken
-	3 Zeilen Vorschub vor dem Drucken
+	überdruckt die vorhergehende Zeile.

S — Scale: Nach der Kopfzeile wird ein Zeilenlineal ausgedruckt.

L — Long: Falls eine Zeile länger ist als die Druckbreite, wird die Zeile in voller Länge mit entsprechend vielen Fortsetzungszeilen ausgedruckt. Ohne diese Option werden Zeilen nur in der Länge der Seitenbreite ausgedruckt. Die gemeinsame Angabe von Long und Hexa bewirkt, dass alle Zeilen sowohl im Character-Format als auch hexadezimal gedruckt werden.

F — Form feed: Kommt in den Daten am Zeilenanfang oder am Zeilenende das Zeichen X'0C' oder die Zeichenfolge `'<newpage>'` vor, wird an dieser Stelle eine Seitenvorschub eingefügt.

Kleinbuchstaben bei Eingabe in Großbuchstaben umwandeln

LOWER [ON|OFF [,DISP OFF]]

LOWER DISP ON|OFF — Neu eingegebene Kleinbuchstaben in Großbuchstaben umwandeln oder unverändert als Kleinbuchstaben belassen.

LOWER ON — Neu eingegebene Kleinbuchstaben werden unverändert in den Arbeitsbereich übertragen und nicht in Großbuchstaben umgewandelt.

LOWER OFF — Neu eingegebene Kleinbuchstaben werden in Großbuchstaben umgewandelt. Bestehende Kleinbuchstaben werden angezeigt.

LOWER OFF,DISP OFF — Gleiche Wirkung wie Kommando LOW OFF. Bestehende Kleinbuchstaben werden wie im BS2000-EDT am Bildschirm als Schmierzeichen dargestellt.

LOWER DISP ON — Diese Variante des Kommandos setzt den Modus LOWER OFF,DISP OFF auf den Standardwert zurück.

LOWER OFF,DISP ON — Gleiche Wirkung wie Kommando LOW OFF. Bestehende Kleinbuchstaben werden im Gegensatz zum BS2000-EDT am Bildschirm dargestellt.

Großbuchstaben in Kleinbuchstaben umwandeln

LOW *rngcol*|*str-var*[- *str-var*]

Alle Großbuchstaben im Bereich *rngcol* bzw. in der oder den String-Variablen *str-var* werden in Kleinbuchstaben umgewandelt. Es werden auch alle Umlaute und sprachabhängige Sonderzeichen, wie z. B. â, á à usw. umgesetzt. Die Umsetzung erfolgt in Abhängigkeit von den Definitionen in der Datei `CODEPAGE.TXT` (siehe Kapitel 17 Code-Tabellen (S. 359)). Mit dem Kommando `UPPER` können Kleinbuchstaben in Großbuchstaben umgewandelt werden.

Zeilen kennzeichnen / hervorheben

MARK *ln* , *type* [,*'text'*]

ln — Die zu markierende Zeilennummer.

type — Art der Markierung

N	Note	(Gelb)
W	Warning	(Orange)
E	Error	(Rot)
S	Structure-Error	(Dunkelrot)

text — Der optional angegebene Hinweis-/Fehler-Text wird beim Anklicken des Satzes in der Statuszeile des EDTW eingeblendet. Durch die markierten Zeilen kann mit F3 (nächste) und F4 (vorherige) geblättert werden.

Verzeichnis erstellen

MK**D**IR *path* [C] — Das Dateiverzeichnis *path* wird erstellt. Ggf. wird ein Verzeichnisbaum erstellt, falls die darüberliegenden Verzeichnisse nicht vorhanden sind.

path — Verzeichnis als String (S. 341). Es sind alle Varianten, wie Stringvariable, Zeilennummer usw. zulässig.

C — Create*:* Ist ein übergeordnetes Verzeichnis nicht vorhanden, werden die fehlenden Verzeichnisse ohne Rückfrage angelegt. Ohne diese Option wird im Dialogmodus gefragt, ob das Verzeichnis angelegt werden soll. Im Prozedurmodus wird ein fehlendes Verzeichnis immer angelegt.

Beispiel:

```
mkddir 'verz1' c
mkdir #s1
md #s1+'neu'
md #l1:20-50:
```

Kopieren und Löschen aus einem anderen Arbeitsbereich

MOVE *rng* (*n*) — Übertragen von Daten aus einem anderen Arbeitsbereich (*n*). Die Zeilen des Sendebereiches werden mit den originalen Zeilennummern in den Ziel-

bereich übertragen, d.h. im Zielbereich können Zeilen überschrieben werden. Die Daten im Sendebereich werden nach der Übertragung gelöscht.

MOVE *rng* [(*n*)] TO *ln1*[(*inc*)] [:*ln2*] [,*ln1*[(*inc*)] [:*ln2*] ,.....]

Übertragen von Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich. Die Zeilen erhalten die Nummern von *ln1* in der Schrittweite von *inc*. Die Daten im Sendebereich werden gelöscht. Die maximale Zeilennummer im Empfangsbereich ist *ln2*. Hierbei können Zeilen im Zielbereich überschrieben werden.

MOVE *rng* [(*n*)] { AFTER|BEFORE } *ln*

Übertragen Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor (B) bzw. nach (A) der Zeile mit den Nummer *ln*. Die Daten im Sendebereich werden gelöscht. Dieses Kommando stellt eine **Erweiterung** zum Kommando `MOVE` des EDT im BS2000 dar. Im Zielbereich können keine Zeilen überschrieben werden.

MOVE *rng* [(*n*)] { FIRST|LAST }

Kopieren Daten aus einem anderen (*n*) oder aus dem aktuellen Bereich in den aktuellen Arbeitsbereich vor dem ersten (F) bzw. nach der letzten Zeile (L). Die Daten im Sendebereich werden gelöscht. Dieses Kommando stellt eine **Erweiterung** zum Kommando `MOVE` des EDT im BS2000 dar. Im Zielbereich können keine Zeilen überschrieben werden.

Datei verschieben / umbenennen

MOVEFILE *file1=file2* [O] Die Datei *file1* wird in die Datei *file2* umbenannt. Sind die Dateien auf verschiedenen Dateisystemen bzw. Laufwerken, wird die Datei *file1* in die Datei file2 kopiert und anschließend die Datei *file1* gelöscht. Zum Kopieren von Dateien können Sie das Kommando `COPYFILE` (S. **186**) verwenden.

file1 Dateiname der Ursprungsdatei als String (S. **341**). Es sind alle Varianten, wie Stringvariable, Zeilennummer usw. zulässig.

file2 Dateiname der neuen Datei als String (S. **341**). Es sind alle Varianten, wie Stringvariable, Zeilennummer usw. zulässig.

O Overwrite: Falls die neue Datei *file2* bereits existiert, wird sie ohne Rückfrage überschrieben. Ohne diese Option wird über eine Messagebox gefragt, ob die Datei überschrieben werden soll. Im Prozedur-Modus wird die Datei immer überschrieben.

Beispiele:

```
movefile 'datei1'='datei2' o
movefile #s1='datei2'
movefile #s1=#s2
```

ON Dateibearbeitung mit Suchbegriff

ON überprüft einen Zeilenbereich auf das Vorhandensein eines Suchbegriffs und führt eine der folgenden Aktionen aus:

Ändern des Suchbegriffs;
Löschen des Suchbegriffs;
Positionieren auf Zeile mit Suchbegriff und markieren;
Kopieren der Zeilen mit Suchbegriff;
Kopieren der Zeilen mit Markierung;

Löschen des Zeileninhalts vor oder nach dem Suchbegriff;
Einfügen einer Zeichenfolge vor oder nach dem Suchbegriff;
Löschen Zeilen mit Suchbegriff;
Löschen aller markierten Zeilen;
Löschen aller leeren Zeilen;
Zurücksetzen der Markierung;
Auflisten der Zeilen mit den Suchbegriffen.

Verwendung von Musterzeichen im Suchbegriff

Neben konstanten Zeichen können auch variable, sogenannte Musterzeichen, angegeben werden. Es gibt zwei Musterzeichen:

* ersetzt eine beliebig lange, auch leere Zeichenfolge. Die Angabe von mehreren "*" nebeneinander ist nicht zulässig. Ebenfalls ist die Angabe "/*" oder "*/" nicht zulässig.

/ ersetzt genau ein Zeichen.

Die Zeichen können umdefiniert werden (Menüzeile: `Optionen / Sonderzeichen` (S. 119) oder Kommando `SYMBOLS` (S. 252)). Die Musterzeichen wirken jedoch nur, wenn der Zusatz `PATTERN` vor dem Suchstring angegeben wird.

Negatives Suchen

Durch die Angabe des Schlüsselwortes `NOT` werden die Sätze gesucht, die den Suchbegriff nicht enthalten.

Festhalten eines Treffers

Der EDT hält fest, ob ein Treffer festgestellt wurde oder nicht.
Im Dialog sind die Suchbegriffe in den Trefferzeilen hervorgehoben. Mit der Taste `F3` kann zur nächsten Trefferzeile positioniert werden. Mit der Taste `F4` wird zur vorhergehenden Trefferzeile positioniert. Mit der Taste `F9` kann auf die Trefferzeile ab der nächsten Bildschirmseite positioniert werden.

In Prozeduren kann mit dem Kommando `IF` (Format 3) abgefragt werden, ob eine Trefferzeile vorhanden ist.

Die Zeilennummer des ersten Treffers wird in der Line-Variablen #L0 festgehalten. Die Nummer der Spalte, in der beim ersten festgestellten Treffer der Suchbegriff beginnt, wird in der Integer-Variablen #I0 gespeichert. Die Spalte, in der der Suchbegriff endet, wird in der Integer-Variablen #I1 gespeichert.

Falls es sich um einen Arbeitsbereich mit mehreren Dateien (S. 224) handelt (Menü `Datei/Öffnen mehrfach` oder Kommando `READ '*.ext'`), wird der Dateiname der Datei mit dem Treffer in die String-Variable #S0 geschrieben. Statt der Variablen #S0 kann mit dem Kommando `PAR MULTIREAD` (S. 218) auch eine andere String-Variable zugewiesen werden.

Wird als Suchbereich ein Bereich von mehreren String-Variablen angegeben, z.B. `on#s1-#s20 find 'xx'`, so wird in die Integer-Variable #I99 die Nummer der String-Variablen mit dem Suchbegriff gespeichert. So kann z.B. mit der Angabe `#S0+#I99` auf diese String-Variable zugegriffen werden.

Groß-Kleinschreibung

Über die Dialogbox `Optionen / Suchen` (S. **110**) bzw. mit dem Kommando `SEARCH-OPTION` (S. **244**) kann eingestellt werden, ob die Groß-/Kleinschreibung bei der Suche beachtet werden soll. Zusätzlich kann diese Option in der Dialogbox `Bearbeiten/Suchen` (S. **67**) oder durch die entsprechende Variante des Suchstrings (V'string' oder L'String') zum Kommando `ON` (S. **210**) und Kommando S (S. **241**) gewählt werden.

Falls die Option LOW OFF (Kleinbuchstaben in Großbuchstaben umwandeln) aktiv ist, wird die Zeichenfolge unabhängig von der Sucheinstellung in Großbuchstaben umgewandelt.

Rückfrage vor Änderung

Durch die Angabe des Operanden Q (Query) wird vor jeder Aktion (CHANGE, PRINT, DELETE, COPY) in einer Dialogbox gefragt, ob die Aktion durchgeführt werden soll. Folgende Eingaben sind zulässig:

oder J: Aktion für diese Zeile ausführen;

oder N: Aktion für diese Zeile nicht ausführen;

oder A: Aktion für alle Zeilen ausführen.

oder E: Kommando abbrechen.

Suchen und Ersetzen von Zeichenfolgen

ON *rngcol* **C**HANGE [**A**LL] [**F**IRST] [**R**] [**P**ATTERN] [**Q**] *str1* [,*int*] **T**O *str2*

In allen gefundenen Zeilen wird die Zeichenfolge *str1* durch die Zeichenfolge *str*2 ersetzt. Statt des Schlüsselworts `TO` kann auch das Zeichen "=" angegeben werden. Dieses Kommando entspricht dem Format 7 des ON-Kommandos im BS2000-EDT.

Suchbegriff löschen

ON *rngcol* **D**ELETE [**A**LL] [**F**IRST] [**R**] [**P**ATTERN] [**Q**] *str* [,*int*]

Der Suchbegriff wird in allen gefundenen Zeilen gelöscht. Der restliche Zeileninhalt bleibt erhalten. Dieses Kommando entspricht dem Format 10 des ON-Kommandos im BS2000-EDT.

Suchen und Markieren der Trefferzeilen

ON *rngcol* **F**IND [FIRST] [ONCE | ALL] [NOT] [PATTERN] *str* [,*int*]

ON *rng* **S**EARCH *searchstr*

Suchen Zeilen mit der Zeichenfolge *str* bzw. *searchstr*, markieren der Trefferzeilen und positionieren auf den ersten Treffer. Dieses Kommando entspricht dem Format 4 des ON-Kommandos im EDT unter BS2000.

SEARCH — Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar. Die Syntax entspricht dem Suche-Kommando im CFS.

searchstr — Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*) miteinander verknüpft sind. Für jedes Suchargument kann eine Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe S. **346**.

ON *rng* **F**IND **L**ENGTH *len* — Suchen Zeilen mit der Satzlänge *len*, markieren der Trefferzeilen und positionieren auf den ersten Treffer. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

len — *nnn* | =*nnn* | >*nnn* | <*nnn* | *nnn* - *mmm*

nnn — Satzlänge bzw. "Satzlänge von".

mmm — "Satzlänge bis".

ON *rng* **F**IND *DOS | *UNIX | *NO

Suchen Zeilen mit bestimmten Satzende-Kennzeichen in Arbeitsbereichen, in denen verschiedene Satzende-Kennzeichen vorkommen. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

*DOS — Suchen von Sätzen mit dem DOS- bzw. Windows- Satzende-Kennzeichen (X'0D0A').

*UNIX — Suchen von Sätzen mit dem UNIX- Satzende-Kennzeichen (X'0A').

*NO — Suchen von Sätzen ohne Satzende-Kennzeichen. Dabei handelt es sich um Sätze, die entweder länger als die maximale Satzlänge sind oder um den letzten Satz, falls die Datei nicht mit einem Satzende-Kennzeichen abgeschlossen ist.

Suchen und Kopieren der Trefferzeilen

ON *rngcol* **F**IND [**F**IRST] [**N**OT] [Q] MARK [**C**OPY TO] (*n*) [KEEP|OLD|KEEPOLD|F|L|A *ln*|B *ln*]

Alle markierten Zeilen in den Arbeitsbereich *n* kopieren. Dieses Kommando entspricht dem Format 5 des ON-Kommandos im BS2000.

KEEP — Der Zielarbeitsbereich wird vor dem Übertragen gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten. Wird KEEP nicht angegeben, werden die Zeilen im Zielarbeitsbereich ab der aktuellen Zeile mit der aktuellen Schrittweite erzeugt.

KEEP OLD oder OLD KEEP — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten.

OLD — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilen werden im Zielarbeitsbereich ab der aktuellen Zeilennummer mit der aktuellen Schrittweite erzeugt.

F — Die Trefferzeilen werden vor der ersten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

L — Die Trefferzeilen werden nach der letzten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

A *ln* — Die Trefferzeilen werden nach der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

B *ln* — Die Trefferzeilen werden vor der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

ON *rngcol* **F**IND [**F**IRST] [**N**OT] [**P**ATTERN] [Q] *str* [,*int*] [**C**OPY TO] (*n*) [**K**EEP|**O**LD|KEEPOLD|F|L|A*ln*| B*ln*]

ON *rng* **S**EARCH *searchstr* [**C**OPY TO] (*n*) [**K**EEP|**O**LD|KEEPOLD|FIRST|LAST|A*ln*|B*ln*]

Kopieren aller gefundenen Zeilen in den Arbeitsbereich *n*. Entspricht dem Format 6 des ON-Kommandos im EDT unter BS2000.

KEEP — Der Zielarbeitsbereich wird vor dem Übertragen gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten. Wird KEEP nicht angegeben, werden die Zeilen im Zielarbeitsbereich ab der aktuellen Zeile mit der aktuellen Schrittweite erzeugt.

KEEP OLD oder OLD KEEP — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilennummer des Sendebereichs bleibt erhalten.

OLD — Der Zielarbeitsbereich wird vor dem Übertragen nicht gelöscht. Die Zeilen werden im Zielarbeitsbereich ab der aktuellen Zeilennummer mit der aktuellen Schrittweite erzeugt.

F — Die Trefferzeilen werden vor der ersten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

L — Die Trefferzeilen werden nach der letzten Zeile im Zielarbeitsbereich kopiert. Diese Option stellt eine Erweiterung zum EDT im BS2000 dar.

A *ln* — Die Trefferzeilen werden nach der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

B *ln* — Die Trefferzeilen werden vor der Zeile mit der Zeilennummer *ln* im Zielarbeitsbereich kopiert. Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar.

SEARCH — Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar. Die Syntax entspricht dem Suche-Kommando im CFS.

searchstr — Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*****) miteinander verknüpft sind. Für jedes Suchargument kann eine Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe S. **346**.

Einfügen / Löschen vor oder nach dem Suchbegriff

ON *rngcol* **F**IND [ALL] [**F**IRST] [R] [**P**ATTERN] [Q] *str1* [,*int*] { **C**HANGE|**I**NSERT } **P**REFIX *str2*

In allen Trefferzeilen wird *str2* vor dem Suchbegriff *str1* ersetzt bzw. eingefügt. Dieses Kommando entspricht dem Format 8 des ON-Kommandos im BS2000-EDT.

ON *rngcol* **F**IND [**A**LL] [**F**IRST] [R] [**P**ATTERN] [Q] *str1* [,*int*] { **C**HANGE|**I**NSERT } **S**UFFIX *str2*

In allen Trefferzeilen wird *str2* nach dem Suchbegriff *str1* ersetzt bzw. eingefügt. Dieses Kommando entspricht dem Format 9 des ON-Kommandos im BS2000-EDT.

ON *rngcol* **F**IND [**A**LL] [**F**IRST] [R] [**P**ATTERN] [Q] *str* [,*int*] **D**ELETE **P**REFIX

Der Inhalt vor dem Suchbegriff wird in den Trefferzeilen gelöscht. Dieses Kommando entspricht dem Format 11 des ON-Kommandos im BS2000-EDT.

ON *rngcol* **F**IND [**A**LL] [**F**IRST] [R] [**P**ATTERN] [Q] *str* [,*int*] **D**ELETE **S**UFFIX

Der Inhalt nach dem Suchbegriff wird in den Trefferzeilen gelöscht. Dieses Kommando entspricht dem Format 12 des ON-Kommandos im BS2000-EDT.

Suchen und Löschen der Trefferzeilen

ON *rngcol* **F**IND [**A**LL] [**F**IRST] [R] [**P**ATTERN] [Q] *str* [,*int*] **D**ELETE

ON *rng* **S**EARCH [Q] *searchstr* **D**ELETE

Löschen aller gefundenen Zeilen. Dieses Kommando entspricht dem Format 13 des ON-Kommandos im BS2000-EDT.

SEARCH — Diese Option stellt eine **Erweiterung** zum EDT im BS2000 dar. Die Syntax entspricht dem Suche-Kommando im CFS.

searchstr — Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*****) miteinander verknüpft sind. Für jedes Suchargument kann ein Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe S. **346**.

ON *rngcol* **F**IND [**F**IRST] [NOT] [Q] **M**ARK **D**ELETE

Alle markierten Zeilen werden gelöscht. Dieses Format stellt eine **Erweiterung** zum BS2000-EDT dar.

ON *rngcol* **F**IND [FIRST] [Q] **E**MPTY **R**ECORDS] **D**ELETE
ON *rngcol* **FERD**
ON *rngcol* **FED**

Alle leeren Zeilen werden gelöscht. Dieses Format stellt eine **Erweiterung** zum BS2000-EDT dar.

Zurücksetzen von Markierungen

ON *rngcol* **F**IND [FIRST] **M**ARKS **R**ESET

Im angegebenen Bereich wird bei allen Zeilen die FIND-Markierung zurückgesetzt. Dies ist ein **zusätzliches Format** zum BS2000-EDT.

Suchen und Anzeigen der Trefferzeilen

ON *rngcol* **P**RINT [**A**LL] [**F**IRST] [R] [**P**ATTERN] [Q] *str* [,*int*] [N]

Auflisten der gefundenen Trefferzeilen (N: ohne Zeilennummern). Dieses Kommando entspricht dem Format 1 des ON-Kommandos im BS2000-EDT.

Markierungsspalte und Daten stets editierbar

PAR EDIT FULL [=ON | OFF]

Die Markierungsspalte und das Datenfenster werden permanent überschreibbar/nicht überschreibbar. Anstelle von PAR EDIT FULL= ON|OFF kann auch die Kurzform EDIT FULL ON|OFF angegeben werden.

PAR EDIT LONG [=ON|OFF]

Ganze Zeile anzeigen: siehe Kommando EDIT LONG (S. **190**).

PAR EDIT INDENT [=ON|OFF]

Automatisch einrücken: siehe Kommando EDIT INDENT (S. **189**).

PAR EDIT WORD [=ON|OFF]

Word-Modus: siehe Kommando EDIT WORD (S. **190**)

Protokollierung von Fehlermeldungen

PAR ERRMSG=N | Y [, 1 | 2 | 3]

Mit dieser Anweisung kann der Umfang der Protokollierung im Prozedurmodus (Kommando DO (S. **277**) oder INPUT (S. **206**) bzw. Schalter -i (S. **35**) beim Laden) gesteuert werden.

N — Fehlermeldungen werden unterdrückt.

Y — Fehlermeldungen werden ausgegeben (Standard).

1 — Es wird nur die Fehlermeldung und das fehlerhafte Kommando (einschl. Zeilennummer und der Arbeitsbereich des Kommandos) ausgegeben.

2 — Es werden zusätzlich die Inhalte aller String-, Integer-, Float- und Line-Variablen sowie der Inhalt des Schleifensymbols und der symbolischen Zeilennummern (*,%,$,?) aller Arbeitsbereiche ausgegeben.

Beispiel:

```
ERR: Ungültige oder fehlende Parameter
>>> 0001.0000(1) @print !xxx
Line - Variables
#L01=0050.4500
Integer - Variables
#I01=+0000000005 #I02=+0000000006
String - Variables
#S01(0004)=var1
#S02(0004)=var2
Procedure - Variables
(1) loop: act=0005.0000 beg=0001.0000 end=0101.0000
                                      step=+00010000
(1) *=0001.0000   %=0001.0000   $=0101.0000 ?=0000.0000
(0) *=0001.0000   %=0001.0000   $=0201.0000 ?=0000.0000
```

3 — Die erweiterte Ausgabe (Inhalt der Variablen wie zu 2) gilt auch für die Protokollierung des Kommando `DO PRINT` (S. **277**)

Positionieren nach Kommando ON....FIND

PAR FPOS=[-|+]*n* | Y | N

[-|+]*n*
Nach dem Kommando `ON....FIND` wird als erste Zeile die Zeile mit dem Suchbegriff +/- *n* Zeilen angezeigt.

N
Nach dem Kommando `ON....FIND` bleibt die Anzeige unverändert. Insbesondere in Prozeduren, die von der EDT-Kommandozeile (DO oder INPUT) gestartet werden, kann dies sinnvoll sein.

Y
Die Positionierung nach dem Kommando `ON....FIND` wird wieder eingeschaltet. Es gilt der vor dem Kommando `PAR FPOS=N` geltende Wert für die Zeilenanzahl.

Menüzeile: `Optionen / Suchen` (S. **110**)

Help-Modus einstellen

PAR HELPMODE= HLP | CHM

HLP
HELP-Informationen aus der Datei EDTW.HLP mit WINHELP anzeigen. Dieser Modus wird nur noch aus kompatibilitätsgründen unterstützt und kann eingestellt werden, falls die Anzeige der CHM-Datei Probleme verursacht. Ab der Windows Version 7 ist allerdings das Programm WINHELP nicht mehr als Standardprogramm installiert. Ggf. muss dieses Programm erst installiert werden.

CHM
HELP-Informationen aus der Datei EDTW.CHM anzeigen.

Da zum Anzeigen der HTMLHELP-Datei EDTW.CHM intern Funktionen des Internet-Explorers benutzt werden, kann es bei Netzlaufwerken zu Problemen kommen, wenn aufgrund von Sicherheitseinstellungen der Zugriff blokkiert wird. Die Themen werden dann entweder nicht angezeigt oder es kommt eine Fehlermeldung. Um diese Probleme zu vermeiden, wird die Datei EDTW.CHM, wie die Datei EDTW.INI, in das lokale Verzeichnis %APPDATA% übertragen und von dort gelesen.

Fehlermeldung "Zeilennummer nicht vorhanden" ignorieren

PAR IGNORE-LINE-ERROR= Y | N

N
Beim Zugriff auf eine nicht vorhandene Zeilennummer wird eine Fehlermeldung ausgegeben.

Y
Beim Zugriff auf eine nicht vorhandene Zeilennummer wird **keine** Fehlermeldung ausgegeben.

Warten nach Bildschirmausgaben im Zeilenmodus

PAR KEYWAIT=Y | N

In Prozeduren können in bestimmten Situationen Dialogabfragen unterdrückt werden.

N
Ist für die Ausgabe von Meldungen in den Protokoll-Arbeitsbereich 32 die Option "Bestätigung bei voller Seite" eingeschaltet (S. **123**), so wird keine Bestätigung angefordert. Sind bei Programm-Ende Daten noch nicht gesichert, wird das Programm ohne Sicherungsabfrage beendet.

Y — Die Rückfragen bei Seitenüberlauf und Programm-Ende werden nicht unterdrückt (Standardeinstellung).

Protokollierung von Hinweisen

PAR LOGMSG=N | Y

Mit dieser Anweisung kann der Umfang der Protokollierung im Prozedurmodus (Kommando `DO` (S. **277**) oder `INPUT` (S. **206**) bzw. Schalter -i (S. **35**) beim Laden) gesteuert werden.

N — Hinweismeldungen werden unterdrückt.

Y — Hinweismeldungen werden ausgegeben (Standard).

Standard-Variablen für Mehrfach Lesen ändern

PAR MULTIREAD=#S*n*,#I*n*,#I*n*

Mit dieser Anweisung können die Standard-Variablen, die von der Funktion Mehrfach-Lesen verwendet werden, geändert werden.

#S*n* — String-Variable, in der beim Kommando `ON&FIND` (S. **210**) der aktuelle Dateinamen übertragen wird. Standard = `#S0`.

#I*n* — Integer-Variable, in der nach dem Einlesen von mehreren Dateien in einen Arbeitsbereich die Anzahl der fehlerfrei gelesenen Dateien übertragen wird. Standard = `#I98`.

#I*n* — Integer-Variable, in der nach dem Einlesen von mehreren Dateien in einen Arbeitsbereich die Anzahl der fehlerhaft gelesenen Dateien übertragen wird. Standard = `#I99`.

Translate-Modus

PAR TRANS=YES|NO — Werden Daten von einem Arbeitsbereich in einen anderen übertragen (z.B. Kommandos `COPY`, `MOVE`, Markierungen C, M, R, A, B in der Markierungsspalte, Kopieren über die Zwischenablage, Drag & Drop usw.) oder 2 Arbeitsbereiche verglichen (Kommando `COMP`) und die Arbeitsbereich sind unterschiedlich codiert, kann wahlweise pro Arbeitsbereich die Übersetzung in die Codierung des Empfangs-Arbeitsbereichs ein- oder ausgeschaltet werden. Für das Kommando `COMP` (S. **184**) enthält zusätzlich eine Option `TRANS`, mit der die Übersetzung gesteuert werden kann.

YES — Die Daten werden in die Codierung des Empfangs-Arbeitsbereichs übersetzt.

NO — Die Daten werden nicht übersetzt.

Als Standard gilt bei Textdateien `TRANS=YES` und bei Binärdateien `TRANS=NO`.

True / False-Abfrage

PAR TRUE-FALSE=GLOBAL | LOCAL

Die Abfrage mit `IF .TRUE./.FALSE.` (S. 282) bezog sich bis zur Version 2.24 auf das letzte ON-Kommando im aktuellen Arbeitsbereich. Im BS2000-EDT bezieht sich die `.TRUE.`- bzw. `.FALSE.`-Abfrage global auf das letzte ON-Kommando. Mit dieser Option kann das Verhalten des BS2000-EDT eingestellt werden kann.

LOCAL — Funktion wie EDT für Windows bis V 2.24: Die True-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando im aktuellen Arbeitsbereichs ein Treffer gefunden wurde. Die False-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando im aktuellen Arbeitsbereich kein Treffer gefunden wurde oder im aktuellen Arbeitsbereich noch kein ON-Kommando aufgerufen wurde.

GLOBAL — Funktion wie EDT für BS2000: Die True-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando in einem beliebigen Arbeitsbereich ein Treffer gefunden wurde. Die False-Bedingung ist erfüllt, wenn mit dem letzten ON-Kommando in einem beliebigen Arbeitsbereich kein Treffer gefunden wurde oder in dieser Session noch kein ON-Kommando aufgerufen wurde.

Hinweis:

Mit dem Menü `OPTIONEN / EINSTELLUNGEN / VERSCHIEDENE` (S. 127) kann der Modus für das Kommando IF ebenfalls eingestellt werden. Das Kommando `IF .TRUE./.FALSE.` (S. 282) wurde um die Möglichkeit erweitert, einen Arbeitsbereich anzugeben. Damit kann auch das Ergebnis eines ON-Kommandos in einem anderen Arbeitsbereich geprüft werden, ohne dass das Verhalten des BS2000-EDT eingeschaltet wird.

Variablen-Substitution

PAR VARSUBST=YES | NO

In Zeichenfolgen werden spezielle EDT-System-Variablen, wie z.B. Dateinamen, Datum, Uhrzeit usw., substituiert. Das Einleitungszeichen für die Variablennamen kann geändert werden (Menü `Optionen / Einstellungen / Sonderzeichen2` (S. 121) bzw. Kommando `QUOTE` (S. 220)) Standard = "!". Weitere Informationen siehe S. 344.

YES — Die Variablen-Substitution wird eingeschaltet.

NO — Die Variablen-Substitution wird ausgeschaltet.

Als Standard gilt VARSUBST=NO.

Windows-Fenster im Prozedurmodus aktualisieren

PAR WINUPD=YES | NO

Windows-Fenster im Prozedurmodus aktualisieren/nicht aktualisieren. In länger laufenden EDT-Prozeduren kann das Ausschalten der Fenster-Aktualisierung den Ablauf wesentlich beschleunigen.

Zeichenfolge am Zeilenanfang einfügen

PREFIX *rng* WITH *str*

In allen Zeilen des Bereichs *rng* wird am Zeilenanfang die Zeichenfolge *str* eingefügt.

Zeilenbereich anzeigen

PRINT [*rngcol*| *str-var*[*-str-var*] | *string*] [N]

Anzeigen eines Zeilenbereichs. Es wird eine temporäre Datei erzeugt, die in einem eigenen Fenster angezeigt wird. Die temporäre Datei wird nach der Anzeige gelöscht.

rngcol Zeilen- und Spaltenbereich. Bei fehlender Angabe wird der gesamte Arbeitsbereich angezeigt.

str-var String-Variable.

string Beliebiger String in der Form '....' oder X'....' (in der Regel nur in Prozeduren sinnvoll).

N Die Zeilennummern werden unterdrückt.

Begrenzersymbol für Zeichenfolgen umdefinieren

QUOTE [*spec*] [*,char*] [*,esc*] Wenn in einer Anweisung eine Zeichenfolge anzugeben ist (mit search, string, file usw.), ist diese in Hochkommas einzuschließen. QUOTE definiert Zeichen, die diese Hochkommas ersetzen.

spec Sonderzeichen, ersetzt das Hochkomma (’).

char Zeichen, ersetzt das Anführungszeichen (").
spec und *char* müssen verschieden sein.

Anführungszeichen finden ausschließlich beim Kommando `ON` Verwendung, und zwar im Zusammenhang mit Textbegrenzern (siehe `DELIMIT` und `ON`). Hochkommas werden ebenfalls bei ON ... angewandt, darüber hinaus aber auch in anderen Anweisungen.
Anwendungsfälle für diese Anweisung ergeben sich beispielsweise, wenn bei ON ... eine Suchzeichenfolge aufgefunden werden soll, die ein Hochkomma oder ein Anführungszeichen enthält.

esc Einleitungszeichen für die Namen von EDT-System-Variablen (S. **344**). Die Substitution erfolgt nur, wenn sie mit der Option "Ersetzung in Zeichenfolgen" im Menü `Optionen / Einstellungen / Verschiedenes` (S. **127**) oder dem Kommando `PAR VARSUBST` aktiviert wird. Das Einleitungszeichen kann auch mit dem Menü `Optionen / Einstellungen / Sonderzeichen2` (S. **121**) definiert werden.

Datei in Arbeitsbereich einlesen

READ ['*file*[**:***ads*]' | '*zip(* [*path/*] *elem)*' | *str-var* | ?] [,] [*rngcol*]] [**B**INARY [*rngbyte*] | **R**ECORD | **S**TRIP] [**T**ABULATOR] [**K**EY] [**O**PEN] [*code*]

READ STDIN

Dateiname

Es wird eine Datei in den aktuellen Arbeitsbereich eingelesen.
Befinden sich in dem Arbeitsbereich bereits Daten, so wird die Datei an das Ende der bisherigen Daten angefügt.

Beim Einlesen von Dateien werden die Zeichen CR/LF (`X'0D0A'`) gesucht und als Endekriterium für einen Satz gewertet. Dateien, die die Zeichen CR/LF nicht oder in einem Abstand der maximalen Satzlänge aufweisen, werden als Binärdaten ohne Satzstruktur angesehen und im EDT in einem besonderen Format dargestellt. Dazu wird der bisher eingelesene Teil des Arbeitsbereichs gelöscht und die Datei neu gelesen. Die Datei wird im Arbeitsbereich in Teilen zu jeweils 64 Byte dargestellt. Die Tatsache, dass hier keine echten Zeilenende-Kennzeichen vorhanden sind, ist an dem Zeichen "*" in den EDT-Zeilennummern zu erkennen. Bei satzstrukturierten Dateien wird hier ein Punkt "." angezeigt.

file — Dateiname. Der Name muss in Hochkommas eingeschlossen werden. Enthält der Dateinamen keinen Pfadnamen, wird als Pfad das Arbeitsverzeichnis benutzt (siehe auch Kommando `CHDIR` (S. **179**)).

ads — Alternate Data Streams (ADS): Der ADS zu der angegebenen Datei *file* wird eingelesen. Diese Erweiterung wird nur von Windows NT/2000 mit NTFS-Dateisystem unterstützt. Die ADS sind für den Explorer "unsichtbar". Eine Liste der ADS können Sie mit dem Kommando `FSTAT` (S. **197**) erzeugen.

zip([path/]elem) — Element eines ZIP-Archivs. Der Name muss in Hochkommas eingeschlossen werden. Enthält der Dateinamen des ZIP-Archivs keinen Pfadnamen, wird als Pfad das Arbeitsverzeichnis benutzt (siehe auch Kommando `CHDIR` (S. **179**)). Vor dem Elementnamen kann wahlweise ein Verzeichnisname innerhalb des ZIP-Archivs angegeben werden. Das Element wird aus dem ZIP-Archiv in das TEMP-Verzeichnis extrahiert, eingelesen und sofort wieder gelöscht.

? — Ist mit dem Kommando `FILE` ein Dateiname definiert worden, so wird beim Kommando `READ` ohne Dateinamen die voreingestellte Datei eingelesen.

str-var — String-Variable, die den Dateinamen enthält.

STDIN — Die Daten werden von der Systemdatei STDIN gelesen. Das Einlesen von STDIN ist nur möglich, wenn beim Laden eine STDIN - Datei mittels des Pipe-Zeichens zugewiesen wird. In der Regel wird diese Variante nur bei EDT-Prozeduren zum Einsatz kommen. Das Einlesen der STDIN-Datei kann auch mit dem Schalter `-stdin` beim Laden des EDTW erfolgen.

Beispiel: `prog | edtw.exe -iprocfile.edt`.

Format und Sätze und Spalten aus Datei auswählen

rngcol — Zeilen - und/oder Spaltenbereich der zu lesenden Zeilen. Die Zeilennummer bezieht sich auf die logischen Zeilennummern. Beispielsweise werden mit `0.0001-0.0005` die ersten 5 Zeilen der Datei eingelesen. Mit `0.0001-0.0005:1-3:` werden die Spalten 1-3 der ersten 5 Zeilen eingelesen. Wenn statt des Dateinamens eine String-Variable angegeben wurde, ist die String-Variable von der Bereichsangabe *rngcol* durch ein Komma zu trennen. Es können mehrere Zeilen- und oder Spaltenbereiche durch Komma getrennt angegeben werden. Die Zeilen können auch durch Line-Variable und die Spalten durch Integer-Variable angegeben werden. Die Bereichsangabe wird nicht ausgewertet, wenn die Datei binär eingelesen wird.

BINARY — Die Datei wird unabhängig von evtl. vorhandenen Satzstruktur im Format einer Binärdatei (siehe oben) in den Arbeitsbereich eingelesen.

rngbyte *rng* [,*rng*.....]

rng { *int* | *intvar* | $-*int* } - { *int* | *intvar* | $[-*int*] }

Bereich der einzulesenden Bytes im Binärmodus. Die Bereichsangabe kann direkt als Integer, als Integer-Variable oder als "Dateiende - Anzahl" angegeben werden. Die Bereiche können sich auch überlappen. Die Daten werden insoweit mehrfach eingelesen.

int Erstes bzw. letztes einzulesendes Byte als Ganzzahl.

intvar Integer-Variable mit der Nummer des Bytes

$-*int* Distanz zum Dateiende.

Beispiele:

```
@set#i1=1
@set#i2=10000
1-10000,1-100   Bytes 1-10000 und 1-100
#i1-#i2         Bytes 1-10000
$-10000-$       Die letzten 10000 Bytes der Datei
#i2-20000       Bytes 10000 bis 20000
$-#i2-$         Die letzten 10000 Bytes der Datei
```

RECORD Die Datei ist satzstrukturiert, enthält jedoch Sätze mit einer Länge größer als die maximale Satzlänge. In diesem Modus werden die Sätze, die länger sind, in Teilsätzen mit der maximalen Satzlänge im Arbeitsbereich dargestellt. Beim Zurückschreiben werden diese Teile wieder zu einem langen Satz zusammengefügt.

Daten-Aufbereitung

STRIP Löscht die Leerzeichen am Ende jeder eingelesenen Zeile. Besteht eine Zeile nur aus Leerzeichen, werden alle Leerzeichen gelöscht, die Zeile bleibt jedoch erhalten. Diese Option ist nur bei satzstrukturierten Dateien möglich. Enthält die Datei Sätze größer als die maximale Satzlänge, wird die gleiche Satzbehandlung wie mit der Option `R` durchgeführt.

KEY Die ersten 8 Zeichen jedes einzulesenden Satzes werden als Schlüssel interpretiert. Die Sätze, die in den Arbeitsbereich eingelesen werden, erhalten diesen Schlüssel als Zeilennummer und nicht als Teil des Zeileninhalts. Dabei wird geprüft, ob in den ersten 8 Zeichen jeder Zeile ein gültiger Schlüssel steht Ziffern 0 bis 9).

TABULATOR Nach dem Einlesen wird das Kommando `STT` (S. **251**) ausgeführt. Dadurch werden Leerzeichen durch ein Tabulatorzeichen ersetzt. Vor dem Schreiben mit dem Kommando `WRITE` ohne Angabe eines Dateinamens wird automatisch das Kommando `TTS` (S. **254**) ausgeführt, so dass die Tabulatorzeichen wieder in Leerzeichen umgewandelt werden. Dadurch wird erreicht, dass während des Editierens die Tabulatorfunktion genutzt werden kann, ohne dass Tabulatorzeichen gespeichert werden müssen.

Diese Option kann z.B. dazu genutzt werden, um BS2000-Assembler-Sourcen zu bearbeiten. Durch Einstellung einer Tabulatorschrittweite (S. **108**) von 71 wird erreicht, dass alle Leerstellen vor der Spalte 71 durch ein Tabulatorzeichen ersetzt werden. Die Spalten 72-80 (Fortsetzungszeichen und Numerierung) werden dann nicht verschoben, wenn in den Spalten 1-71 ein Zeichen gelöscht oder eingefügt wird. Der Tabulator, der über das Kommando `TABS` (S. **253**) eingestellt wird, kann zusätzlich für die Spalten 1-71 genutzt werden.

OPEN-Modus

OPEN

Die Daten werden nicht in den Arbeitsspeicher eingelesen. Im Speicher wird nur ein Verzeichnis der Datensätze mit den Plattenadressen angelegt. Benötigte Daten der Eingabedatei werden bei Bedarf direkt von der Platte gelesen. Neue bzw. geänderte Datensätze werde in eine temporäre Datei geschrieben. In diesem Modus kann eine Datei mit max. 1.024 GB editiert werden.

Beim Zurückschreiben der geänderten Datei mit dem Kommando `WRITE` werden die Daten aus der Eingabedatei und der temporären Datei zusammengefügt und in die Ausgabedatei geschrieben. Soll die eingelesene Datei überschrieben werden, so werden die Daten zuerst in eine temporäre Datei geschrieben. Danach wird die Originaldatei gelöscht und die temporäre Datei umbenannt.

Die Eingabedatei bleibt währen der Bearbeitung geöffnet und kann somit in dieser Zeit von einem anderen Prozess nicht gelesen oder überschrieben werden. Im Open-Modus gibt es keine Einschränkungen beim Editieren. Die einzige Ausnahme besteht darin, dass keine AUTOSAVE-Datei und beim Schreiben keine BACKUP-Datei (S. 106) erstellt wird. Auch UNDO- und Redo-Aktionen sind wie im Normalmodus möglich.

Codierung (UNICODE-Dateien)

code

CHAR | UTF | UCB | UCL

Falls einzulesende Dateien einen UNICODE-Header (Byte Order Mark) oder ein XML-Header enthalten, wird automatisch die richtige Codierung verwendet. Die Angabe der Codierung ist dann nicht notwendig. Ist ein Header vorhanden und weicht das angegebene Format von den Angaben im Header ab, wird eine Fehlermeldung ausgegeben.

UTF16-Dateien (UCB oder UCL) ohne UNICODE-Header werden in der Regel anhand des Satzende-Kennzeichens innerhalb der ersten 5000 Bytes erkannt und werden ebenfalls automatisch richtig eingelesen.

UTF8-Dateien können anhand von UTF8-Zeichenfolgen innerhalb der ersten 5000 Bytes automatisch erkannt werden, falls die entsprechende Option im Menü `Optionen / Code / Dateien ohne Unicode-Header` (S. 117) aktiviert ist.

CHAR

Datei im Ein-Byte-Format. Dieses Format wird automatisch benutzt, falls kein UNICODE-Header vorhanden ist (siehe hierzu auch Menü `Optionen / Code / Dateien ohne Unicode-Header` (S. 117)). Die Option CHAR wird auch bei UNICODE-Dateien mit Header akzeptiert. In diesem Fall wird ein evtl. vorhandener Header ebenfalls als Daten eingelesen und jedes UNICODE-Zeichen als zwei Zeichen dargestellt.

UTF

Datei im UTF-8-Format. UTF-8 (Abk. für 8-Bit Unicode Transformation Format) ist die am weitesten verbreitete Codierung für Unicode-Zeichen. Dabei wird jedem Unicode-Zeichen eine speziell codierte Bytekette von variabler Länge zugeordnet. UTF-8 unterstützt bis zu vier Byte, auf die sich wie bei allen UTF-Formaten alle 1.114.112 Unicode-Zeichen abbilden lassen. Das UTF-8-Header enthält `X'EFBBBF'`

UCB

UTF-16BE Unicode Big-Endian (höherwertiges Byte zuerst). UTF-16 (Abk. für 16-bit Unicode Transformation Format) ist eine Codierung, bei der jedes Unicode-Zeichen mind. 16 Bit belegt. Unicode-Zeichen, deren Code sich nicht mit 16 Bit darstellen läßt, belegen zwei 16-Bit-Wörter (code units). Das UTF-16-Header enthält `X'FEFF'`.

Das höherwertige Byte steht in den ersten 8 Bits, z.B. Die Zahl "0" (`X'30'`) erhält den Wert `X'0030'`.

UCL

UTF-16LE Unicode Little-Endian (niederwertiges Byte zuerst). Das UTF-16-Header enthält `X'FFFE'`.

Das niederwertige Byte steht in den ersten 8 Bits, z.B. Die Zahl "0" (`X'30'`) erhält den Wert `X'3000'`.

Beispiele:

`read'datei1'`
Die Datei `datei1` wird vollständig eingelesen

`read'datei1'o`
Die Datei `datei1` wird im Open-Modus vollständig eingelesen.

`read'datei2'0.0001-1,5-6:21-30,1-10:`
Von der Datei `datei2` werden die Sätze 1 bis 10.000, und 50.000 bis 60.000 eingelesen. Von jedem Satz werden nur die Spalten 20-30 und 1-10 in den Arbeitsbereich übertragen. Die Spalten werden so umorganisiert, dass in den ersten 10 Spalten des Arbeitsbereichs die Spalten 21-30 der Datei und in den Spalten 11-20 des Arbeitsbereichs die Spalten 1-10 der Datei stehen.

`read'datei2'b$-10000-$`
Die Datei wird im Binärformat eingelesen. Von der Datei `datei2` werden nur die letzten 10.000 gelesen.

`read'datei3'b1-10000,50000-60000`
Die Datei wird im Binärformat eingelesen. Von der Datei `datei3` werden nur die Bytes 10.000 bis 20.000 und 50.000 bis 60.000 gelesen.

```
par varsubst=y
read 'datei1..!user'
```
`!user` wird durch den Benutzernamen (S. **344**) ersetzt (nur Win-NT), z.B. `datei1.benutzer1`

`read#s1,0.0001-1`
Von der Datei, dessen Name in der Variablen #s1 steht, werden die ersten 10.000 Sätze eingelesen.

`read test.zip(verz1/test1.src)`
Das Element `verz1/test1.src` aus dem ZIP-Archiv `test.zip` wird in den aktuellen Arbeitsbereich eingelesen.

Mehrere Dateien in einen Arbeitsbereich einlesen

READ '*files*' [**B**INARY | **R**ECORD | **S**TRIP] | '(*filelist*)' | [*rng*] (*n*) [**SUB**DIR] [**O**PEN]

files

Kommen im Dateinamen die Jokerzeichen "?" (ersetzt genau ein Zeichen) oder "*" (ersetzt eine beliebig lange, auch leere Zeichenfolge) vor, werden alle Dateien, die das Suchmuster erfüllen, hintereinander in den aktuellen Arbeitsbereich eingelesen. Das gleiche gilt, falls ein Dateiname ohne Jokerzeichen in Verbindung mit der Option `SUBDIR` angegeben wird. Das Kommando kann auch mehrmals in einem Arbeitsbereich aufgerufen werden. Die zusätzlich ausgewählten Dateien werden dann am Schluss des Arbeitsbereichs angehängt.

(*filelist*)

Dateiname einer Datei, in der für jede einzulesende Datei ein Satz mit dem Dateinamen steht. Die Dateinamen können wahlweise auch durch Anführungszeichen eingeschlossen sein (`"TESTDATEI"`). Die Option `SUBDIR` ist bei dieser Variante nicht zulässig.

[*rng*] (*n*)

Voraussetzung für diese Option ist, dass die auszuwertenden Zeilen jeweils einen Dateinamen enthalten. Eine solche Liste von Dateinamen kann z.B. mit dem Kommando `FSTAT` erzeugt werden. Alle Dateien, deren Dateinamen in dem Zeilenbereich *rng* stehen, werden hintereinander in den Arbeitsbereich *n* eingelesen. Die Option `SUBDIR` ist bei dieser Variante nicht zulässig.

OPEN

Die Daten werden nicht in den Arbeitsspeicher, sondern in eine temporäre Datei eingelesen. Im Speicher wird nur ein Verzeichnis der Datensätze mit den Plattenadressen angelegt. Neue bzw. geänderte Datensätze werde ebenfalls in die temporäre Datei geschrieben. Die Gesamtgröße aller einzulesenden Dateien darf max. 1024 GB betragen.

Beim Zurückschreiben der geänderten Datei mit dem Kommando `REWRITE` werden die Daten aus der temporären Datei in die entsprechenden Ausgabedateien geschrieben.

SUBDIR

Neben dem angegebenen Verzeichnis bzw. dem aktuellen Verzeichnis werden auch alle Unterverzeichnisse rekursiv durchsucht, d.h. mit dem Kommando `READ 'C:\*.BAT'SUB` erhalten Sie z.B. alle BAT-Dateien aller Verzeichnisse des Laufwerks C. Ohne die Option `SUB` wird nur das Root-Verzeichnis durchsucht. Diese Option ist nicht zulässig bei den Varianten (*filelist)* und [*rng*] (*n*)

Hinweise:

Bei Betätigen der rechten Maus-Taste wird ein zusätzliches Untermenü (S. **53**) für die Bearbeitung der aktuellen Datei zur Verfügung gestellt.

Das Einlesen von mehreren Dateien kann auch über das Menü `Datei/Öffnen mehrfach` (S. **58**) **...** erfolgen.

Vor dem Beginn einer neuen Datei wird eine Zeile mit dem Dateinamen, der Größe der Datei und dem Datum der letzten Änderung eingefügt. Diese Zeile beginnt mit dem String "$$$***>>>>>>>>>> " (3 x $, 3 x * und 10 x >). Nach der letzten Zeile einer Datei wird ein Endezeile eingefügt, der mit dem String "<<<<<<<<<<***$$$" beginnt und ebenfalls die Statusangaben der Datei enthält.

In der Statuszeile des EDT wird in diesem Modus der Dateiname der jeweils aktuellen Datei angezeigt. Mit dem Kommando `SHOW DIR` (S. **247**) kann ein Inhaltsverzeichnis aller Dateien des Arbeitsbereichs angezeigt werden.

Alle im Arbeitsbereich enthaltenen Dateien können gemeinsam geändert werden, z.B. mit dem Kommando `ON...CHANGE`, und danach mit dem Kommando `REWRITE` (S. **237**) bzw. dem Menübefehl `Datei/Speichern mehrfach` (S. **63**) in die Ursprungsdateien oder in neue Einzeldateien in einem anderen Verzeichnis geschrieben werden.

Nach dem Einlesen werden folgende Integer-Variablen versorgt:

#I98 enthält die Anzahl der fehlerfrei gelesenen Dateien.
#I99 enthält die Anzahl der Dateien, bei denen ein Fehler aufgetreten ist.

Wird in einer EDT-Prozedur nach dem Einlesen das Kommando ON... benutzt, so wird der Dateinamen der Trefferzeile in die String-Variable #S0 geschrieben.

Die Einlesephase kann mit der Schaltfläche  oder mit dem Menü `Extras/Prozedur oder WAIT` (S. 159) abgebrochen werden.

Die vom EDT erzeugte erste und letzte Zeile, die zur Identifizierung der enthaltenen Dateien dienen, dürfen nicht gelöscht oder geändert werden, da dies bei einer Folgeverarbeitung mit dem Kommando `REWRITE` zu Fehlern führen kann.

Beispiel:

`read '*.ini'` alle Dateien mit Extension `ini` in den aktuellen Arbeitsbereich einlesen.

`fstat'*.*'r` Liste mit allen Dateien einschl. Dateien der Unterverzeichnisse erstellen. Nach dem FSTAT wird automatisch der Arbeitsbereich 9 angezeigt.

`read &(2)` Alle Dateien der FSTAT-Liste in den Arbeitsbereich 2 einlesen oder.

`read 1-900(2)` Die ersten 900 Dateien der FSTAT-Liste in den Arbeitsbereich 2 einlesen.

`read '(list)'` Es werden alle Dateien eingelesen, deren Dateinamen in der Datei `list` enthalten sind (`datei1, datei2, datei3`).
Inhalt der Datei `list`:
```
datei1
datei2
datei3
```

```
read 'c:\*.ini' SUBDIR
```
Es werden alle `INI`-Dateien eingelesen, die sich auf dem Laufwerk C befinden

```
read 'c:\edtw.ini' SUBDIR
```
Es werden alle Dateien mit dem Namen `EDTW.INI` eingelesen, die sich auf dem Laufwerk C befinden

Dateien von anderen Rechnern einlesen

READ '*ft* | *batch* R=*remote-file* [L=*local-file*] [M=*mode*] [*authorisation*] [FP=*fpass*]
[*par1 par2 ....*]' [**O**PEN] [*rngcol*] [**B**INARY [*rngbyte*]]

READ *str-var* [**O**PEN] [*rngcol*] [**B**INARY [*rngbyte*]]

Mit diesem Format des READ-Kommandos wird zuerst mit dem internen Filetransfer des EDT eine Datei von einem entfernten Rechner übertragen bzw. mit einer Batch-Prozedur verarbeitet. Diese lokale Datei wird dann in den Arbeitsbereich eingelesen. Für den Filetransfer ist keine zusätzliche Software auf dem PC notwendig.

Beim Lesen der höchsten Version eines Bibliothekelements wird die Versionsnummer im Reiter angezeigt und auch für eventuell folgende Schreibaktionen verwendet. Handelt es sich bei der höchsten Version um die Standardversion (X'FF' bzw. @), wird aus Kompatibilitätsgründen kein @ angezeigt

str-var String-Variable, die alle Parameter innerhalb der Hochkommas enthält.

Filetransfer-Profil

ft

Name eines Filetransfer-Profils, das mit dem Menübefehl `Extras/Filetransfer...` (S. 182) definiert werden kann. Aus den Parametern des Kommandos und den Angaben aus dem Profil wird die Schnittstelle des internen EDT-Filetransfers erzeugt und der Filetransfer gestartet.

Sind Profile vorhanden, die in mehreren INI-Dateien mit dem gleichen Namen enthalten sind, kann durch Voranstellen der Dateinummer (1, 2 oder 3) die entsprechende INI-Datei angegeben werden.

Beispiel:

`1std` = Profil `STD` aus der privaten INI-Datei
`2std` = Profil `STD` aus der INI-Datei vom Windowsverzeichnis
`3std` = Profil `STD` aus der INI-Datei vom Ladeverzeichnis.

batch

Dateiname der Batchdatei. Die Extension ".BAT" muss hierbei nicht angegeben werden. Die Batch-Prozedur wird mit folgenden Parametern gestartet:

batch-file par1 par2 .....

Zusätzlich werden folgende Umgebungs-Variablen vor dem Aufruf der Batch-Prozedur gesetzt:

`call:`	`"R"`
`locfile:`	Name der lokalen Datei
`remfile:`	Name der entfernten Datei
`lpass:`	Logon-Passwort
`fpass:`	Datei-Passwort

Die Angabe von "R" in der Variablen `call` bedeutet, dass die Batch-file von dem Kommando `READ` aufgerufen wurde.

Lokale Datei

local-file

Dateiname der zu erzeugenden Datei auf dem lokalen Rechner. Ist dieser Parameter nicht angegeben, wird eine temporäre Datei erzeugt, die nach dem Filetransfer gelöscht wird.

Entfernte Datei

remote-file *bs2-file* | *lib*([*typ*/]*elem*[/*version*]) | */posix-file* | *mvs-file*

bs2-file

Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID. Es kann auch eine Generation aus einer Dateigenerationsgruppe in der BS2000-Syntax angegeben werden (z.B. `fgg(*0003))`.

lib

Name der PLAM-Bibliothek auf dem fernen Rechner.

typ

Typ des PLAM-Elements. Es sind nur die Angaben S, M, J, P und D zulässig. Wird kein Typ angegeben, so wird der Typ "S" als Standard benutzt.

elem

Name des PLAM-Elements. Delta-geführte Elemente können vom openFT-BS2000 nur gelesen werden.

version

Version des PLAM-Elements. Fehlt die Version, wird die höchste Version des Elements benutzt.

/posix-file

Beginnt der Dateinamen mit dem Zeichen "/", so wird die Datei automatisch von dem entsprechenden POSIX-Dateisystem gelesen. Es ist immer der Dateiname ab ROOT anzugeben. Beim Filetransfer mit FTP muss im Filetransfer-Profil der Typ FTP-POSIX ausgewählt werden.

mvs-file *data-set* | *pds(member)* | *'hlq.data-set'* | *'hlq.pds(member)'*

Bei der Variante `read'profile r=mvsfile'` müssen die Hochkommas verdoppelt werden, weil der ganze Parameter bereits ein String in Hochkommas ist. Soweit alle Angaben in einer Stringvariablen stehen, dürfen die Hochkommas nicht verdoppelt werden. Soll z.B. eine Datei aus der FSTAT-Liste eingelesen werden, so muss lediglich der Inhalt aus der Spalte 1-256 in eine Stringvariable übertragen werden. Dann kann die Datei mit dem Befehl `read` *strvar* eingelesen werden.

Um die Angabe der doppelten Hochkommas zu umgehen, können auch folgende alternative Formate angegeben werden:

mvs:hlq.data-set | *mvs:hlq.pds(member)* oder
$hlq.data-set | *$hlq.pds(member)*

data-set Dateiname auf dem fernen Rechner ohne `HLQ` (high level qualifier). Vom Host-System wird der Dataset-Name automatisch um den HLQ ergänzt.

pds(member) Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended) ohne HLQ. Vom Host-System wird der Member-Name automatisch um den HLQ ergänzt.

`'hlq.data-set'` | *mvs:hlq.data-set* | *$hlq.data-set*

Vollqualifizierter Dateiname auf dem fernen Rechner. Aus diesem String wird für den Filetransfer der Dateiname `'hlq.data-set'` erzeugt.

`'hlq.pds(member)'` | *mvs:hlq.pds(member)* | *$hlq.pds(member)*

Vollqualifiziertes Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended). Aus diesem String wird für den Filetransfer der Dateiname `'hlq.pds(member)'erzeugt.`

Zugangsdaten

Die Zugangsdaten können im Filetransfer-Profil (Extras/Filetransfer.... (S. 182)) definiert werden oder bei den Kommandos `FILE` (S. 252), `READ` (S. 302) und `WRITE` (S. 353) angegeben werden. Es ist auch möglich, einen Teile der Zugangsdaten im Profil und einen Teil bei den Kommandos anzugeben.

authorisation [LH=host] [LU=user] [LA=account] [LP=lpass|?] oder TA=ta

host | ? Host-Name für den Zugang zum entfernten Rechner. Der Hostname kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

user | ? User-ID für den Zugang zum entfernten Rechner. Die User-ID kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

acount | ? Abrechnungs-Nummer für den Zugang zum entfernten Rechner. Die Abrechnungs-Nummer kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

lpass | ?

Logon-Passwort für den Zugang zum entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *lpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

ta | ?

Transfer-Admission des FTAC-Profils für den Zugang zum entfernten Rechner. FTAC-Profile können mit der Software FTAC erstellt werden. Die Transfer-Admission muss mindestens 8 Stellen und darf höchstens 32 Stellen lang sein. Enthält der Name Blanks, muss er in Hochkommas eingegeben werden. Die Transfer-Admission kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

fpass|*?*

File-Passwort im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

Übertragungsmodus

mode

Transfer-Modus. Diese Angabe besteht aus den 2 Information:

a) Übertragungsmodus Text/Binär:

Text-Modus bedeutet, dass die Daten mit der Standard-Translatetabelle bzw. der Translatetabelle, die im FT-Profil angegeben ist, übersetzt werden (ANSI <--> EBCDIC/ASCII).

Im **Binär-Modus** werden die Daten transparent ohne Übersetzung übertragen. Es sind drei verschiedene Modi zu unterscheiden:

a) Binär satzstrukturiert für openFT-BS2000 (M=B)
b) Binär undefiniert für openFT-BS2000 (M=U)
c) Binär für FTP (M=B)

Nach dem Einlesen der Datei in den Arbeitsbereich wird die Codierung automatisch auf den entsprechenden Code umgeschaltet (wie Kommando `EBCDIC` (S. 235)).

Gilt nur für openFT: In der lokalen Datei (Name wird entweder vom EDT vergeben oder Name aus "L=file") werden die Daten mit 4 Byte Satzlängenfeld ohne Satzende-Kennzeichen gespeichert. Dadurch ist es möglich, dass in einem Satz auch das BS2000-Satzende-Kennzeichen (X'15') oder das Unix-Satzende-Kennzeichen (X'0A') vorkommen kann.

Im Filetransfer-Profil kann der Übertragungsmodus ebenfalls eingestellt werden. Wird beim Kommando `READ` nichts angegeben, so gilt die Einstellung des FT-Profils. Wird beim Kommando `WRITE` nichts angegeben, so gilt entweder der Übertragungsmodus des vorhergehenden `READ` oder die Einstellung des FT-Profils, falls eine lokale oder neue Datei ohne vorhergehenden `READ` auf den fernen Rechner übertragen wird.

b) Feste/Variable Satzlänge. Soll eine Datei mit fester Satzlänge erzeugt werden, müssen alle Zeilen im Arbeitsbereich exakt die richtige Länge haben.

Folgende Werte sind zulässig bei **openFT_BS2-Profilen:**

T Text-Modus (Standardeinstellung). Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Beim Schreiben ohne weitere Formatangabe wird die ferne Datei allerdings mit dem Format "Variable Satzlänge" erzeugt.

TFnnn
Text-Modus feste Satzlänge. Die Datei wird mit RECFORM=F und RECSIZE=nnn gelesen bzw. erstellt. Wurde diese Option beim Lesen angegeben, so wird beim Schreiben ohne Parameter automatisch das gleiche Dateiformat erzeugt.

TV Text-Modus, variable Satzlänge (gilt nur für openFT-BS2000). Diese Angabe ist nur beim Schreiben notwendig, falls nach dem Lesen einer Datei mit fester Satzlänge eine Datei mit variabler Satzlänge erzeugt werden soll.

B Binär-Modus. Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Die Daten werden im Arbeitsbereich im EBCDIC-Code satzstrukturiert dargestellt. Ist die Datei im Binär-Modus gelesen worden und wird sie mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, wird sie automatisch wieder im Binär-Modus transferiert, allerdings immer mit variabler Satzlänge.

BFnnn
Binär-Modus feste Satzlänge. Die Datei wird wie bei M=B binär mit RECFORM=F und RECSIZE=nnn gelesen bzw. erstellt. Die Daten werden im Arbeitsbereich im EBCDIC-Code satzstrukturiert dargestellt. Wurde diese Option beim Lesen angegeben, so wird beim Schreiben ohne Parameter automatisch das gleiche Dateiformat erzeugt.

BV Binär-Modus, variable Satzlänge. Diese Angabe ist nur beim Schreiben notwendig, falls nach dem Lesen einer Datei mit fester Satzlänge eine Datei mit variabler Satzlänge erzeugt werden soll.

U Binär-Modus undefined (ohne Satzstruktur). Mit dieser Option können Dateien ohne Satzstruktur gelesen und geschrieben werden. Dieser Modus ist z.B. notwendig, wenn eine POSIX-Datei aus einem ASCII-Filesystem gelesen werden soll. Nach dem Einlesen kann der Arbeitsbereich mit dem Kommando `REFORMAT UNIX` und `CODE ASCII` satzstrukturiert dargestellt werden. Vor dem Schreiben muss der Arbeitsbereich mit dem Kommando `UNFORMAT` wieder in das Binärformat umgewandelt werden.

Folgende Werte sind zulässig bei **FTP-Profilen:**

T Text-Modus (Standardeinstellung). Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Dateien, die im BS2000 mit fester Satzlänge gespeichert sind, werden beim Schreiben in die gleiche Datei automatisch im richtigen Format erzeugt. Neue Dateien können nur mit variabler Satzlänge erzeugt werden.

B Binär-Modus. Die Daten werden transparent ohne Übersetzung übertragen und im Arbeitsbereich ohne Satzstruktur dargestellt. Soweit es sich um BS2000-Dateien mit variabler Satzlänge (RECFORM=V) oder Unix/Windows-Textdateien handelt, enthalten die Daten folgende Satzende-Kennzeichen: Windows=X'0D0A', Unix = X'0A' und BS2000 = X'15'. Soweit es sich um BS2000-Dateien mit fester Satzlänge (RECFORM=F) handelt, enthalten die Daten kein Satzende-Kennzeichen. Die Codierung kann im Profil schon auf den Wert ANSI, ASCIIUNIX oder EBCDIC7 eingestellt werden.

Nach dem Einlesen kann die Satzstruktur des fernen Rechners mit dem Kommando `REFORMAT` (S. 312) wieder hergestellt werden.

Beispiel:

```
REFORMAT BS2      (BS2000, variable Satzlänge)
REFORMAT RS80     (BS2000, feste Satzlänge 80)
REFORMAT UNIX     (Unix-Datei)
REFORMAT DOS      (Windows-Datei)
```

Vor dem binären Schreiben eines solchen Arbeitsbereichs muss mit dem Kommando `UNFORMAT` wieder das Binärformat erzeugt werden.

Wird der Arbeitsbereich mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, werden die Daten automatisch wieder im Binär-Modus transferiert und die Datei auf dem fernen Rechner mit den bestehenden Dateiattributen erstellt.

Datei-Passwort

fpass|*?*

Passwort für den Zugriffsschutz der Datei auf dem entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort in einer Dialogbox unsichtbar angefordert.

Datei-Attribute für MVS

mvs-fileattr

RECFM=recfm LRECL=lrecl [BLKSIZE=blksize]

recfm

Record format: F | FB | FBA | V | VB | VBA | U

lrecl

Record length: Satzlänge, max. 32767

blksize

Block size: Blocklänge, max. 32767. Fehlt die Angabe der Blocklänge, wird vom MVS-System ein optimaler Wert ermittelt. In der Regel kann deshalb dieser Parameter entfallen.

Parameter für die Folgeverarbeitung

par1 par2

Beliebige Parameter, die als Parameter beim Aufruf an die Batch-Prozedur bzw. die Prozedur für die Folgeverarbeitung übergeben werden.

OPEN

Die Daten der lokalen Datei werden nicht in den Arbeitsspeicher eingelesen. Im Speicher wird nur ein Verzeichnis der Datensätze mit den Plattenadressen angelegt. Benötigte Daten der Eingabedatei werden bei Bedarf direkt von der Platte gelesen. Neue bzw. geänderte Datensätze werde in eine temporäre Datei geschrieben. In diesem Modus kann eine Datei mit max. 1.024 GB eingelesen werden.

Beim Zurückschreiben der geänderten Datei mit dem Kommando `WRITE` werden die Daten aus der lokalen Eingabedatei und der temporären Datei zusammengefügt und in die Ausgabedatei geschrieben.

Die lokale Eingabedatei bleibt währen der Bearbeitung geöffnet und kann somit in dieser Zeit nicht überschrieben werden. Im Open-Modus gibt es keine Einschränkungen beim Editieren. Auch UNDO- und REDO-Aktionen sind wie im Normalmodus möglich.

rngcol Zeilen - und/oder Spaltenbereich der zu lesenden Zeilen. Die Zeilennummer bezieht sich auf die logischen Zeilennummern. Beispielsweise werden mit `0.0001-0.0005` die ersten 5 Zeilen der Datei eingelesen. Mit `0.0001-0.0005:1-3:` werden die Spalten 1-3 der ersten 5 Zeilen eingelesen. Wenn statt des Dateinamens eine String-Variable angegeben wurde, ist die String-Variable von der Bereichsangabe *rngcol* durch ein Komma zu trennen. Es können mehrere Zeilen- und oder Spaltenbereiche durch Komma getrennt angegeben werden. Die Zeilen können auch durch Line-Variable und die Spalten durch Integer-Variable angegeben werden. Die Bereichsangabe wird nicht ausgewertet, wenn die Datei binär eingelesen wird.

BINARY Die Datei wird unabhängig von evtl. vorhandener Satzstruktur im Format einer Binärdatei (siehe oben) in den Arbeitsbereich eingelesen.

rngbyte *rng* [,*rng*.....]

rng { *int* | *intvar* | $-*int* } – { *int* | *intvar* | $[-*int*] }

Bereich der einzulesenden Bytes im Binärmodus. Die Bereichsangabe kann direkt als Integer, als Integer-Variable oder als "Dateiende - Anzahl" angegeben werden. Die Bereiche können sich auch überlappen. Die Daten werden insoweit mehrfach eingelesen.

int Erstes bzw. letztes einzulesendes Byte als Ganzzahl.

intvar Integer-Variable mit der Nummer des Bytes

$-*int* Distanz zum Dateiende.

Beispiele:

```
@set#i1=1
@set#i2=10000
1-10000,1-100     Bytes 1-10000 und 1-100
#i1-#i2           Bytes 1-10000
$-10000-$         Die letzten 10000 Bytes der Datei
#i2-20000         Bytes 10000 bis 20000
$-#i2-$           Die letzten 10000 Bytes der Datei
```

Dialogbox für die Passwort-Eingabe

Das Datei- bzw. Logon-Passwort ist in der Form cccc oder x'xxxxxxxx' anzugeben. Die Hochkommas müssen in der Dialogbox nicht verdoppelt werden.

Beispiel READ mit internem Filetransfer:

```
read'std r=src1.ass'
```
Die Datei `src1.ass` wird vom fernen Rechner gelesen. Es werden die FT-Angaben aus dem FT-Profil `std` benutzt.

```
read'std r=datei1'o
```
Die Datei `datei1` wird vom fernen Rechner gelesen und im Open-Modus bearbeitet. Es werden die FT-Angaben aus dem FT-Profil `std` benutzt.

Beispiel READ mit Batchdatei (aktueller Arbeitsbereich = 0):

```
read'host1 r=test1 l=test1.src user1 acc1'
```

Starten der Batch-Datei host1.bat über eine übergeordnete temporäre Batch-file:

```
set locfile=test1.src
set remfile=test1
set call=R
call host1.bat user1 acc1
```

Inhalt der Datei host1.bat:

```
if R==%call% goto read
if W==%call% goto write
goto ende
:read
ncopy host1@%remfile% %locfile% %1,%2
goto ende
:write
ncopy %locfile% host1@%remfile% %1,%2
if not errorlevel 0 goto ende
del %locfile%
:ende
```

Folgendes Kommando wird in der Batch-Datei zum Lesen der Daten ausgeführt:

```
ncopy host1@source.test1 test1.src user1,acc1
```

Hinweis:

Die Angaben *ft* | *batch* R=*remote-file* können entfallen, falls sie mit dem Kommando CHDIR (S. 179) als Standard definiert sind.

Mit Hilfe der Batch-Datei kann ein beliebiger Arbeitsgang zur Bereitstellung der zu editierenden Datei ausgeführt werden. Wie oben dargestellt kann eine Datei mit einem Filetransfer-Programm von einem BS2000-Host abgeholt werden. Es ist aber auch möglich, eine in einem ARJ-/ZIP-Archiv komprimierte MS-DOS Datei zu entpacken und im EDT zu bearbeiten.

Wiederherstellen

Vorhergehende UNDO-Aktion wieder rückgängig machen.

Shortcuts:

Symbol:
Tasten: <Strg>+Y

Fremde Dateistruktur/UTF-8 in ASCII-Format umwandeln

REFORMAT { RS *n* | LF | LF2 | LF2+[*hhhh*] | LF4 | DOS | UNIX | BS2 | ST *str* } [(*n*)]

REFORMAT UTF

EDT verarbeitet standardmäßig satzstrukturierte ASCII-Dateien, wobei jede Zeile mit CR/LF abgeschlossen ist. Enthält eine Datei keine Zeilenende-Kennzeichen, so kann sie trotzdem bearbeitet werden. EDT liest dann die Datei in Stücken zu je 64 Bytes pro Zeile ein. Solche Zeilen erkennt man daran, dass anstelle des Punktes in der Zeilennummer ein Stern angezeigt wird. Die Datei ist dann im sog. Binär-Modus eingelesen. Dieser Modus kann auch für eine satzstrukturierte Datei beim Kommando READ durch die Option B nach dem Dateinamen erzwungen werden.

Satzstrukturierte Dateien aus anderen Systemen, wie z.B. UNIX oder BS2000, haben einen anderen Aufbau. Um auch solche Dateien im Record-Modus bearbeiten zu können, müssen sie mit dem Kommando REFORMAT analysiert und neu formatiert werden. Die Daten können dann wie eine ASCII-Datei satzweise bearbeitet werden. Die Daten können vor dem Zurückschreiben in die Datei entweder wieder in das ursprüngliche Format umgewandelt werden (Kommando UNFORMAT, siehe S. 255 oder im ASCII-Format gespeichert werden.

Datei mit Sätzen fester Länge

RS *n* Record-Size. Nach dem Schlüsselwort RS ist die Satzlänge anzugeben. Die Daten des gesamten Arbeitsbereichs werden in Sätze mit der Länge *n* aufgeteilt. Der letzte Satz kann kürzer sein, falls die Länge des gesamten Arbeitsbereichs kein Vielfaches der angegebenen Satzlänge ist.

Datei mit variabler Satzlänge und Satzlängenfeld

LF2

Length-Field - Länge 2. Die Datei ist so organisiert, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Ein Satz mit einem Längenfeld von X'0005' enthält z.B. noch drei Nutzbytes. Sätze dieses Formats werden von verschiedenen Textsystemen verwendet (DCA-Format). Enthält das Satzlängenfeld einen Wert < 2, wird die Konvertierung abgebrochen und der gesamte Arbeitsbereich wieder auf den Binärmodus zurückgesetzt. Enthält die Satzlänge einen Wert > max. Satzlänge (Standard 2048, erweiterbar auf 32.767, siehe Menüzeile `Optionen / verschiedene Optionen` (S. **127**)), wird dieser Satz weiter im Binärmodus angezeigt. Alle anderen Sätze werden konvertiert.

LF

Length-Field - Länge 2. "LF" hat die gleiche Bedeutung wie LF2 und wird aus Gründen der Kompatibilität weiterhin unterstützt.

LF2+[*hhhh*]

Length-Field - Länge 2 + 2 Byte X'*hhhh*'. Die Datei ist so organisiert, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Danach folgen 2 Bytes mit dem Wert X'*hhhh*'. Fehlt die Angabe *hhhh*, wird X'4040' benutzt.

LF4

Length-Field - Länge 4. Die Datei ist so organisiert, dass in den ersten vier Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Dieses Format wird z.B. in BS2000-ZIP-Archiven benutzt, die nicht im kompatiblen Format erzeugt wurden.

Hinweis:

Wurde die Datei mit dem Kommando `UNFORMAT LF2H`, vom EDT-Filetransfer oder vom URLServer erzeugt, wird das Format der Datei anhand des Header-Satzes ("`EDTWFILELENGTHFIELD`") erkannt und das Kommando `REFORMAT LF` sowie ggf. das Kommando CODE automatisch ausgeführt.

Datei mit anderem Satz-Trennzeichen

ST *str*

Das Satz-Trennzeichen der Datei ist nach dem Schlüsselwort ST anzugeben. Die Zeichenfolge kann als Character-String, als Hexadezimal-String oder als String-Variable angegeben werden. Der Arbeitsbereich wird nach der Zeichenfolge *str* durchsucht und in Sätze aufgeteilt. Das Satz-Trennzeichen wird gelöscht. Falls ein Satz länger ist als die maximale Satzlänge (Standard 2048, erweiterbar auf 32.767, siehe Menüzeile `Optionen / verschiedene Optionen` (S. **127**)), wird er in 64 Byte langen Teilsätzen dargestellt (zu erkennen an dem Stern in der Zeilennummer).

(*n*)

Arbeitsbereich für die Konvertierung. Ist kein Arbeitsbereich angegeben, wird der Arbeitsbereich 9 benutzt.

UNIX

Der aktuelle Arbeitsbereich enthält eine Datei im UNIX-Format mit den Satz-Trennzeichen `X'0A'`. Die Satz-Trennzeichen werden gelöscht. Mit dem Kommando `UNFORMAT` kann das ursprüngliche UNIX-Format wieder hergestellt werden. Diese Option hat die gleiche Wirkung wie `ST X'0A'`.

BS2

Der aktuelle Arbeitsbereich enthält eine Datei im POSIX-Format des BS2000 mit den Satz-Trennzeichen X'15'. Die Satz-Trennzeichen werden gelöscht. Mit dem Kommando `UNFORMAT` kann das ursprüngliche UNIX-Format wieder hergestellt werden. Diese Option hat die gleiche Wirkung wie `ST X'15'`.

DOS

Der aktuelle Arbeitsbereich enthält eine Datei im MS-DOS-Format mit den Satz-Trennzeichen `X'0D0A'`. Die Satz-Trennzeichen werden gelöscht. Mit dem Kommando `UNFORMAT` kann das ursprüngliche MS-DOS-Format wieder hergestellt werden. Diese Option hat die gleiche Wirkung wie `ST X'0D0A'`.

UTF-8 Daten in UNICODE umwandeln

UTF

Konvertierung eines Arbeitsbereichs mit UTF-8-Daten, falls die Daten wegen des fehlenden UNICODE-Headers im Character-Format eingelesen wurden. UTF-8 kann z.B. daran erkannt werden, dass Deutsche Umlaute als "Ã" (X'C3') mit einem nachfolgend nicht abdruckbaren Zeichen dargestellt werden. Die Daten werden vom variablen Format nach UNICODE umgewandelt. Beim Schreiben eines solchen Arbeitsbereichs wird automatisch wieder eine Datei im UTF-8-Format ohne Header erstellt.

UTF-8 (Abk. für 8-bit Unicode Transformation Format) ist die verbreitetste Codierung für Unicode-Zeichen. Damit werden die meisten Alphabete und Schriftzeichensysteme umfasst, die weltweit derzeit genutzt werden, also nicht nur die lateinischen Buchstaben und arabischen Zahlen, sondern zum Beispiel auch die arabische, griechische, kyrillische, koreanische oder thailändische Schrift. Dabei wird jedem Unicode-Zeichen eine speziell codierte Bytekette von variabler Länge zugeordnet. UTF-8 unterstützt bis zu 4 Byte, auf die sich wie bei allen UTF-Formaten alle 1.114.112 Unicode-Zeichen abbilden lassen.

Beispiel:

```
reformat rs55
reformat lf
reformat st '**]'
reformat st x'0509' (22)
reformat unix
reformat utf
```

Zeilen neu numerieren

RENUMBER [*ln* [(*inc*)]]

Die Zeilen im aktuellen Arbeitsbereich werden neu numeriert.

ln

Anfangswert

inc

Schrittweite der Numerierung.

Standardwert = 1(1)

Wird bei der Neunumerierung der Maximalwert von 9999.9999 überschritten, gehen keine Zeilen verloren! Lediglich die Numerierung kann nicht mehr korrekt angezeigt werden.

Die Standardwerte können im Menü `Optionen / Einstellungen / Standard-Werte` (S. **130**) geändert werden.

Einfügemodus zurücksetzen

RESINS [ON|OFF]

Zurücksetzen des Einfügemodus.

ON

Der durch die Taste `<Einfg>` aktivierte Einfügemodus, wird wie im EDT unter BS2000 nach dem Betätigen der Taste `<Enter>` wieder zurückgesetzt.

OFF

Der Einfügemodus bleibt solange erhalten, bis er durch nochmaliges Betätigen der Taste <Einfg> wieder zurückgesetzt wird.

Menüzeile: Optionen / verschiedene Optionen (S. 127)

Mehrere Dateien zurückschreiben

REWRITE ['*path*' [,S] | *str-var* [,S]] [(*arb*)]

Wurden in einen Arbeitsbereich mehrere Dateien mit dem Kommando READ *rng* (*n*) (siehe auch Kommando FSTAT bzw. READ'..*..') eingelesen, werden alle Dateien zurückgeschrieben. Nach Ausführung des Kommandos wird ein Protokoll ausgegeben, das für jede Datei den vollständigen alten und evtl. neuen Dateinamen und die Anzahl der Sätze enthält.

Wenn kein Pfadname angegeben ist, werden die Originaldateien ohne Warnung überschrieben. Es werden nur die Dateien zurückgeschrieben, die tatsächlich geändert wurden.

path

Die Dateien werden in das angegebene Verzeichnis geschrieben. Der Name des Verzeichnisses muss in Hochkommas eingeschlossen werden. Ist eine Datei bereits vorhanden, erfolgt eine Rückfrage, ob die Datei überschrieben werden soll. Mit dieser Option werden alle Dateien in das angegebene Verzeichnis geschrieben, unabhängig von einer Modifizierung.

S

Subdirectories. Der volle Pfad (ohne Laufwerksbezeichnung) wird an die Pfadangabe *path* angehängt, so dass die ursprüngliche Verzeichnisstruktur erhalten bleibt.

Beispiel:

```
rewrite 'c:\NEU',s
```

Der Arbeitsbereich enthält die Dateien
`C:\TEST1\TEST` und
`C:\TEST2\TEST`

Diese Dateien werden gespeichert unter:
`C:\NEU\TEST1\TEST` und
`C:\NEU\TEST2\TEST`

(*arb*)

Falls beim Schreiben von Dateien Fehler auftreten, werden die Daten der fehlerhaften Dateien einschl. des Beginn- und Endesatzes in den angegebenen Arbeitsbereich geschrieben, d.h. es wird ein neuer REWRITE-Arbeitsbereich erzeugt. Nach Beheben der Fehler (z.B. Netzlaufwerk nicht erreichbar oder Schreibschutz), kann in diesem Arbeitsbereich eine neue REWRITE-Aktion gestartet werden. Der aktuelle Arbeitsbereich bleibt unverändert.

str-var

String-Variable, die den Pfadnamen enthält.

Verzeichnis löschen

RM**D**IR *path*

Das Verzeichnis *path* wird gelöscht. Ein Verzeichnis kann nur gelöscht werden, wenn es leer ist.

path

Verzeichnisname als String (S. 341). Es sind alle Varianten, wie Stringvariable, Zeilennummer usw. zulässig.

Beispiel:

```
rmdir 'verz1'
rmdir #s1
rd #s1+'neu'
rd #l1:20-50:
```

Benutzerroutine aufrufen

RUN ENTRY=*string-entry* [, **M**ODLIB=*string-dll*] [,*par1* [,*par2*,....]]

Mit der Anweisung @RUN wird eine vom Anwender geschriebene Routine (Anwenderroutine) ausgeführt.

string-entry

entry_name[,CDECL]

Der String kann entweder direkt in Hochkommas, als Stringvariable oder als zusammengesetzter String angegeben werden.

entry_name Name der Benutzerroutine als String.

CDECL Aufrufkonvention. Als Standard wird die Methode STDCALL verwendet. Soll in Ausnahmefällen die Methode CDECL verwendet werden, kann die Option CDECL durch Komma getrennt hinter dem Namen der Benutzerroutine angegeben werden.

Beispiel: `ENTRY='TEST,CDECL',.....`

string-dll

Name der DLL als String, die den Einsprungpunkt enthält. Der String kann entweder direkt in Hochkommas, als Stringvariable oder als zusammengesetzter String angegeben werden. Der Name der DLL kann mit einer relativen oder absoluten Pfadangabe beginnen

Wird keine DLL angegeben, wird die DLL mit dem Namen EDTUSER.DLL im Installationsverzeichnis verwendet.

Wird kein Pfad angegeben, wird die DLL zuerst im Installationsverzeichnis, danach in dem vom jeweiligen System vorgegebenen Verzeichnissen gesucht.

par1 [,*par2*,.....]

Ein oder mehrere Parameter für die Funktion. Maximal sind 32 Parameter möglich. Soweit Stringvariable oder Integervariable verwendet werden, kann die Funktion auch Werte zurückgeben. Es sind folgende Parameter zulässig:

a) Beliebige UTF16-Zeichenfolgen. Soweit der Arbeitsbereich nicht in Unicode codiert ist, werden die Daten von dem Arbeitsbereichs-Code in Unicode UTF16 umgewandelt.

   Es sind alle Varianten eines Strings erlaubt, also direkte Angabe einer Zeichenfolge, Stringvariablen, Zeilennummer-Variablen, direkte Angabe einer Zeilennummer, Spaltenangabe und mit "+" verknüpfte Zeichenfolgen.

Die gesamte Zeichenfolge kann max. 32.768 Zeichen lang sein. Die Zeichenfolge kann auch das String-Endezeichen X'0000' enthalten, da die Parameter mit Längenfeld übergeben werden.

Soweit die Funktion in einen Parameter Werte zurückgeben will, muß die Zeichenfolge als Stringvariable angegeben werden, damit der Rückgabewert danach verarbeitet werden kann. Da die Stringvariablen immer in Unicode codiert sind, werden die Rückgabewerte ohne Umcodierung in die Stringvariable übertragen. Erst bei der Verwendung der Stringvariablen werden gegebenenfalls die Daten von UTF16 in den Arbeitsbereichs-Code umgewandelt.

b) Integervariable. Eine Integer-Variable kann einen Wert zwischen -4,61^{18} und +4,61^{18} enthalten.

Ein Integer darf nicht direkt angegeben werden, weil die Zahl nicht von einer Zeilennummer unterschieden werden kann.

Rückgabewerte:

a) Returncode und Integervariable #i99:

Falls der Returncode einen Wert ungleich 0 enthält, wird der EDT Fehlerschalter gesetzt, der mit dem Kommando IF ERROR abgefragt und mit dem Kommando RESET gelöscht werden kann. Der Returncode wird zusätzlich in die Integervariable #i99 übertragen.

b) Meldung und Stringvariable #s99:

Wird im Kontrollblock eine Meldung in UTF16 zurückgegeben, so wird im Prozedurmodus die Meldung in den Arbeitsbereich 32 geschrieben und im Dialogmodus in einer Messagebox ausgegeben. Die Meldung wird zusätzlich in die Stringvariable #s99 übertragen. In der Regel wird hier eine Fehlermeldung zurückgegeben. Es kann jedoch auch im OK-Fall eine Meldung ausgegeben werden.

c) Rückgabewerte in den Parametern:

Alle Parameter, die als Stringvariable oder als Integervariable angegeben wurden, können mit einem Rückgabewert überschrieben werden, der nach dem Rücksprung wieder in die entsprechenden Variablen zurück übertragen wird. Das Format und die Nummer der Variable darf nicht geändert werden. Die Strings sind in UTF16 kodiert.

Beispiele:

```
@run entry='test1',modlib=#s1+'.dll','par1',#s2,#i1
@run entry=#s1,'p1',#l1,#i1,1:50-100:
    #l1=Inhalt der Zeile der Zeilennummer-Variablen #l1
    1:50-100:=Inhalt der Zeile 1, Spalte 50-100
@run e='test2',m='test.dll','p1',25:4-5:,#i1
    25:4-5:=Inhalt der Zeile 25 Spalte 4 bis 5
@run e='test3',m='test.dll','par1'+#s1+'x',#s20,#i1
@if #i99 = 1 goto err1
@if #i99 = 2 goto err2
@if #i1  = 1 goto verarbeitung1
```

Beschreibung der Unterprogramm-Schnittstelle:

Funktionsaufruf:

benutzerroutine (*kontrollblock*, *parmeterliste*)

Alle Bereiche werden vom EDTW in der max. Länge bereitgestellt. Der Benutzermodul muß also keine statischen Felder für die Rückgabewerte definieren.

Kontrollblock:

	Länge	Beschreibung
0	4	Version, Format Integer, aktuelle Version 1
4	4	Adresse Meldung in UTF16, Abschluß String mit X'0000', max. Länge 2048 Zeichen. Das Feld wird vor dem Ansprung mit der Adresse eines Bereiches mit 4KB versorgt. Der Bereich enthält einen Leerstring (X'0000'). Die Meldung wird im Dialogmodus in einer Messagebox, im Batch-Modus in den Arbeitsbereich 32 ausgegeben. keine Meldung = Leerstring (X'0000').
	1	Typ Arbeitsbereich: U = Unicode UTF16 C = 8-Bit-Code Character, jedes Zeichen belegt 2 Bytes, das linke Byte enthält X'00' Beispiel für das Zeichen €: C = X'0080' U = X'20AC' Unabhängig davon werden die Zeichenfolgen immer in UTF16 codiert übergeben!
9	3	reserviert f. Erweiterungen
12	32	Zeichenfolge Code Arbeitsbereich, z.B. "ANSI"

Parameterliste:

Distanz	Länge	Beschreibung
4	var	für jeden Parameter des Kommandos RUN: Adresse des Parameterbereichs.
var	4	Adresse NULL als Ende der Parameterliste

Bereich für jeden Parameter:

Distanz	Länge	Beschreibung
0	2	Format: Unicode-Zeichen `'I'` (Integer) Ganzzahl in der Länge 8 (Longlong) Unicode-Zeichen `'S'` (String) in UTF16 kodiert, jedes Zeichen belegt 2 Bytes.
2	2	Rückgabewert erlaubt: `' '` = kein Rückgabewert erlaubt `'R'` = Rückgabewert erlaubt (#Inn oder #Snn)
4	2	Nummer der Integer- oder String-Variable für den Rückgabewert (0-99), keine Variable = -1
6	2	reserviert
8	4	Länge Daten (ohne Längenfeld): Integer: immer 8, Zeichenfolge: Anzahl der UTF16-Zeichen (0 bis 32.768).
12	4	reserviert
16	8	falls Format 'I' Integer mit Vorzeichen, Der Rückgabewert muss ebenfalls in diesem Feld bereitgestellt werden.
24	4	falls Format 'S': Adresse eines 64KB großen Datenbereichs für die Zeichenfolge, der vom EDTW bereitgestellt wird. Die Zeichenfolge ist immer in UTF16 codiert. Der Rückgabewert muss in diesen Bereich kopiert werden.
28	4	Länge Daten Rückgabewert: Integer: immer 8, Zeichenfolge: Anzahl der UTF16--Zeichen (0 bis 32.768) -1 = keine Rückgabe. Das Feld wird vor dem Ansprung mit -1 gelöscht. Die Rückgabewerte müssen in das Integer-Feld (Distanz 16) bzw. in den Datenbereich (Adr. Distanz 24) übertragen werden.

Returncode:

0	OK, kein Fehler
>0	Fehler. Der Fehlerschalter des EDTW wird gesetzt. Enthält der Kontrollblock keine Fehlermeldung, wird eine allgemeine Fehlermeldung ausgegeben. Die Rückgabewerte werden nicht übertragen.

Hinweis:

Im Installationsverzeichnis ist ein Beispiel-Programm in der Sprache C unter dem Namen EXAMPLE_RUN.C zu finden.

Suchen von Zeichenfolgen (einfaches Suchargument)

S[*n*] [-] ,[*col*] [*r*] *searchstr*

Das Kommando entspricht der CFS-Syntax. Diese Syntax kann auch beim Kommando `ON` *rng* `S` (S. **212**) angegeben werden.

searchstr Suchbegriff mit beliebig vielen Suchargumenten, die mit und (**+**), oder (**,**) bzw. Wildcard (*****) miteinander verknüpft sind. Für jedes Suchargument kann ein Spaltenbereich und ein Vergleichs-Operator (>|<|-) angegeben werden. Ausführliche Beschreibung siehe S. **346**.

Vom der ersten im Sichtfenster angezeigten Zeile bis zum Ende des Arbeitsbereichs wird nach der angegebenen Zeichenfolge gesucht. Das Sichtfenster wird auf die Zeile positioniert, die den ersten Treffer gebracht hat. Im Kommandofeld wird ein Suche-Kommando zum Auffinden des nächsten Treffers vorgegeben.
Durch Drücken der ENTER-Taste (Absenden des Eingabevorschlags) wird die Suche fortgesetzt.

- Rückwärtssuche: Die Suche erfolgt von der ersten im Sichtfenster angezeigten Zeile in Richtung Arbeitsbereich-Anfang
Standard: Suche in Richtung Arbeitsbereich-Ende.

n Anzahl der Zeilen, in denen nach dem Suchargument gesucht wird.
Standard: unbegrenzt viele Zeilen.

col Spaltenbereich, in dem die gesuchte Zeichenfolge beginnen muss.

r > | < | -
> Suche nach einer Zeichenfolge > *str*
< Suche nach einer Zeichenfolge < *str*
- Suche nach einer Zeichenfolge ungleich item
Standard: Suche nach einer Zeichenfolge = *str*.

S Suche von der ersten angezeigten Zeile bis zum Ende des Arbeitsbereichs nach dem zuletzt definierten Suchargument.

S- Suche von der ersten angezeigten Zeile bis zum Anfang des Arbeitsbereichs nach dem zuletzt definierten Suchargument.

S-- Positionieren auf den ersten Treffer.

S++ Positionieren auf den letzten Treffer.

Beispiele:

```
s,'xyz'
```
Suche ab der ersten angezeigten Zeile bis zum Ende des Arbeitsbereichs die Zeichenfolge 'xyz'.

```
s10,:100-200:x'47'
```
Sucht in den nächsten 10 Zeilen jeweils im Spaltenbereich 100 - 200 nach X'47'.

Suchen von Zeichenfolgen (mehrere Suchargumente)

Jedes Suchargument wird durch einen Operator *vk* mit dem jeweils nächsten Suchargument verknüpft. Die Anzahl der zu verknüpfenden Suchargumente ist beliebig.

S[*n*] [-] , *such* [*vk such*] [...] .....

such [*col*] [*r*] *str*
einfaches Suchargument wie im vorhergehenden Abschnitt "Suchen von Zeichenfolgen (einfaches Suchargument)" beschrieben. Ausführliche Beschreibung siehe S. 346.

vk , | + | *
Verknüpfungsoperator mit dem vorausgegangenen Suchargument *such*.

, Suche in der aktuellen Zeile das vorausgegangene **oder** das nachfolgende Suchargument. Die Suchbedingung gilt als erfüllt, wenn zumindest eines der beiden Such-Items in der Zeile enthalten ist.

+ Suche in der aktuellen Zeile das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente in der Zeile enthalten sind. Die Reihenfolge der Suchargumente in der Zeile ist ohne Bedeutung.

***** Suche in der aktuellen Zeile das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente in der Zeile enthalten sind. Die Suchargumente müssen in der gleichen Reihenfolge auftreten, wie im Suche-Kommando angegeben.

Es können beliebig viele Konstrukte der Art *vk such* aneinandergereiht werden.

Hinweise:

Die Reihe der angegebenen Suchargumente und Verknüpfungsoperatoren wird linear abgearbeitet. Falls mehrere mit "+" bzw. "*" verknüpfte Suchargumente angegeben wurden und eines von ihnen nicht in der Zeilen enthalten ist, so wird der Suchvorgang beendet bzw. beim nächsten, mit oder "," verknüpften Such-Item fortgesetzt.

Beispiele:

```
s,'='*'(','DC '*'('*')'
```
Es werden alle Zeilen gesucht, die eine der beiden Bedingungen erfüllen:

- Zeichen '=' und irgendwann danach Zeichen '('. z.B. '=A(...)', '=V(...)'
- Zeichenfolge 'DC ' und irgendwo danach die Zeichen '(' und ')'. z.B. 'DC A(...)', 'DC Y(...)'

```
s,-'a'+-'b'+-'c'
```
Es werden alle Zeilen gesucht, die keinen der Kleinbuchstaben a, b oder c enthalten.

```
s,'a',>'a'+<'z','z'
```
Es werden alle Zeilen gesucht, die mindestens einen Kleinbuchstaben enthalten.

Suchen von Zeichenfolgen (Suchargumente in Datei)

S,(*searchfile*) Name einer Datei, in der die Suchbegriffe gespeichert sind. Ausführliche Beschreibung siehe (S. 348).

SEARCH-OPTION Voreinstellung für Suchen mit ON und S

SEARCH-OPTION **CASELESS-**SEARCH { = ON | = **OF**F }
SEA CASELESS { = ON | = **OF**F }

Mit der Anweisung `SEARCH-OPTION` wird voreingestellt, ob bei der Suche nach Zeichenfolge mit dem Kommando `ON` bzw. der Dialogbox `Bearbeiten / Suchen` (S. 210) nach Groß- und Kleinbuchstaben unterschieden werden soll oder nicht. Die Einstellung verändert nicht die Voreinstellung in der Dialogbox `Optionen / Suchen` (S. 110), die auch in der Parameterdatei `EDTW.INI` gespeichert wird.

ON

Bei der Suche mit dem Kommando `ON` wird nicht unterschieden, ob die Zeichen der Suchfolge mit dem Text in der Groß-Kleinschreibung übereinstimmen, d.h., bei der Suche nach 'string' werden auch die Zeichenfolgen 'String', 'STRING' oder 'STrIng' als Treffer erkannt. Der gleiche Effekt kann auch durch Angabe der Suchzeichenfolge in der Form V'*string*' erreicht werden.

Falls die Option LOW OFF (Kleinbuchstaben in Großbuchstaben umwandeln) aktiv ist, wird die Zeichenfolge auch ohne diese Option in Großbuchstaben umgewandelt.

OFF

Die Groß-/Kleinschreibung eines Zeichens wird bei der Suche beachtet. Der gleiche Effekt kann auch durch Angabe der Suchzeichenfolge in der Form L'*string*' erreicht werden.

Spaltenlineal

SCALE [ON|OFF]

Spaltenlineal einblenden / nicht einblenden.

Menüzeile: `Ansicht / Zeilenlineal` (S. 92)

Symbol: 

SDF Syntaxcheck für BS2000 Prozeduren

SDFTEST [P=*ft-profil*]

Das Kommando SDFTEST überprüft eine BS2000-Prozedur im aktuellen Arbeitsbereich online auf Kommando- und Statement-Syntaxfehler und intern auf Strukturfehler. Die aktuelle Ebene wird dabei nicht gespeichert sondern nur überprüft. Die Prüfung der //-Statements von Fujitsu Standard-Programmen kann im Option-Dialog `Extras->Highlighting...` (S. 149) auch deaktiviert werden.

Die Syntaxprüfung findet bei einer Remote-Datei standardmäßig über das beim Einlesen verwendete FT-Profil statt. Handelt es sich nicht um eine Remote-Datei oder ist das verwendete FT-Profil nicht vom Typ openFT_BS2, so wird die Syntaxprüfung über das Standardprofil SDFTEST durchgeführt.

Es sollte also unter `Extras->Filetransfer...` (S. 136) ein openFT Profil mit dem Namen SDFTEST eingerichtet sein.

Die fehlerhaften Zeilen werden intern mit @MARK gekennzeichnet und mit den entsprechenden Fehlermeldungen und Hinweisen versehen.

ft-profil

Über den Parameter P=... kann das zu verwendende FT-Profil explizit angegeben werden.

Zahlenfolge erzeugen

SEQUENCE [*rng*] [: *cl* :] [*n*(*i*)]

In jede Zeile des Zeilenbereichs wird an einer bestimmten Spalte eine Zahl geschrieben. Diese Zahlen bilden eine aufsteigende Folge.

rng Zeilenbereich.

cl Spalte, in der die erste Ziffer stehen soll (Standard 73).

n Ganze Dezimalzahl für die erste Zeile (Standard = 00000100). *n* hat maximal 8 Stellen. Die Zahlen, die in die folgenden Zeilen geschrieben werden, haben die gleiche Stellenanzahl. Der Wert von *n* ist beliebig. Fehlt *n*, schreibt der EDT in die erste Zeile die Zahl 00000100.

i Schrittweite zur Bildung der folgenden Zeilennummern.
Standard = 100.

Die Standardwerte können im Menü `Optionen / Einstellungen / Standard-Werte` (S. **130**) geändert werden.

SEQUENCE [*rng*] [: *cl* :] LINE

In jede Zeile wird die dazugehörige Zeilen-Nummer geschrieben. Die Zeilennummer wird als 8-stellige Zahl ohne Dezimalpunkt eingefügt.

rng Zeilenbereich.

cl Spalte, in der die erste Ziffer stehen soll (Standard 73).

SEQUENCE [*rng*] [: *cl* :] CHECK [*int*]

Der Inhalt des Spaltenbereichs wird auf aufsteigende Reihenfolge geprüft. Alle Zeilen, in denen der Spalteninhalt gleich oder kleiner der vorhergehenden Zeile ist, werden im Protokollbereich ausgegeben. Alle falschen Zeilen werden mit der FIND-Markierung versehen, d.h. neben der farblichen Hervorhebung kann mit der Taste F3 bzw. mit der Schaltfläche  auf den nächsten falschen Satz positioniert werden.

rng Zeilenbereich.

cl Spalte, in der das erste zu überprüfende Zeichen steht (Standard 73). Die Doppelpunkte vor und nach der Spaltenangabe sind immer anzugeben, auch wenn die Standardeinstellung 73 benutzt wird (z.B. `SEQ ::C`).

int Anzahl der Spalten (Standard 8).

Setzen und Anzeigen von Umgebungsvariablen

Eine Umgebungsvariable kann nur für den aktuellen Prozess und für neu erzeugte Prozesse verwendet werden, d.h. die Variable kann nur innerhalb des EDT und in vom EDT erzeugten Prozessen (z.B. Kommando SYS, Local Success Procedure im FT) ausgewertet werden. Es ist nicht möglich eine Umgebungsvariable an den aufrufenden Prozess, z.B. an eine BAT-Prozedur, in der das Programm EDT aufgerufen wird, weiterzugeben bzw. zurückzugeben.

Der Inhalt von Umgebungsvariablen, die vom aufrufenden Prozess oder innerhalb des EDT erzeugt wurden, kann ohne besonderes Kommando in allen Strings verwendet werden. Dazu ist das Einleitungszeichen von EDT-System-Variablen (S. 344) und der Name der Umgebungsvariablen, eingeschlossen in %, anzugeben.

Beispiele (Die Umgebungsvariable TMP enthält "C:\TEMP"):

```
@crea1:'!%TMP%'          C:\TEMP
@crea2:'!%TMP%\tmp1'     C:\TEMP\tmp1
@pre&w'!%TMP%\'          alle Zeilen werden um den
                         Prefix "C:\TEMP\" ergänzt.
```

SETENV Eine Liste aller Umgebungsvariablen wird im Arbeitsbereich 32 erzeugt. Die Liste enthält jeweils in einer Zeile den Namen und den Wert einer Variablen.

SETENV *env=#string#* Setzen von Umgebungsvariablen

env Name einer Umgebungsvariablen.

string Zeichenfolge, die der Umgebungsvariablen zugewiesen werden soll. Es können alle Varianten von Strings (S. 341) verwendet werden, wie z.B. Stringvariable, Zeilennummern und Zeilennummer-Variable.

Sichtfenster positionieren

SETF (*arb*) [*line* | *vpos*] [*:col:* | *hpos*]

Mit diesem Kommando kann die Anzeige auf einen anderen Arbeitsbereich umgeschaltet werden. Zusätzlich ist die Positionierung auf eine Zeile und Spalte möglich.

Die Kommandos FSTAT und COMP wechseln automatisch in den Ergebnis-Arbeitsbereich, falls nicht der Parameter NF (No focus) angegeben wurde. Dies gilt auch in Prozeduren.

Das Kommando kann im Gegensatz zum BS2000-EDT auch in einer Prozedur angegeben werden. Damit kann bestimmt werden, welche Daten während des Ablaufs einer Prozedur und am Ende der Prozedur angezeigt werden. Es ist zu beachten, dass das Kommando PROC die Anzeige nicht wechselt.

arb Arbeitsdatei (0 - 31), in der positioniert wird. Es wird vor der Positionierung in die Arbeitsdatei verzweigt. Wird nur (*arb*) angegeben, wird nur die aktuelle Arbeitsdatei gewechselt. Es kann auch eine Integer-Variable angegeben werden.

line Absolutangabe der Zeilennummer, entweder direkt oder als Line-Variable.

vpos Relative vertikale Positionieranweisung. Es können +*n*, +*intvar*, ++, –*n*, –*intvar*, sowie – – angegeben werden.

col Absolutangabe der horizontalen Position. Spaltennummer, die als erste Spalte im angegebenen Arbeitsbereich gelten soll. Werte zwischen 1 und 32.767 sind zulässig. Es kann auch eine Integer-Variable angegeben werden.

hpos Relative horizontale Positionieranweisung. Es können >*n*, >*intvar*, <*n*, <*intvar*, >> und << angegeben werden.

Beispiele:

```
setf (5) 10:50:
set #i1=2
set #l1=5
setf (#i1)
setf (5) #l1 :#i1:
setf (5) +5 >5
setf (5) +#i1 >#i1
```

Syntax-Highlighting ein/ausschalten

SHL ON | OFF | *language* — Syntax-Highlighting ein/ausschalten.

ON — Syntax-Highlighting mit der zuletzt eingestellten Sprache des aktuellen Arbeitsbereichs einschalten.

OFF — Syntax-Highlighting ausschalten.

language — Syntax-Highlighting für eine bestimmte Sprache einschalten:

COB	Cobol
ASS	Assembler
SDF	SDF-Prozedursprache
CPP	C/C++

Das Syntax-Highlighting kann auch mit der Schaltfläche in der Ansicht-Toolbar (S. 164) oder mit dem Menübefehl `Ansicht/Syntax-Highlighting` (S. 96) aktiviert oder deaktiviert werden.

Wenn die Option `Auto-Dedect` eingeschaltet ist, wird nach dem Lesen einer Datei automatisch geprüft, ob die Daten Merkmale einer der unterstützten Programmiersprachen enthalten und ggf. die entsprechende Programmiersprache aktiviert.

Siehe auch die Beschreibung der Optionen zum Syntax-Highlighting (S. 149).

Inhaltsverzeichnis eines Arbeitsbereichs ausgeben

SHOW**D**IR [*arb*]

Falls ein Arbeitsbereich mehrere Dateien enthält (siehe Kommando READ, Seite 220), wird ein Inhaltsverzeichnis aller im Arbeitsbereich enthaltenen Dateien erstellt und in den Arbeitsbereich *arb* übertragen. Fehlt der Parameter *arb*, wird der Arbeitsbereich 9 benutzt.

Darstellung des NIL-Zeichens (X'00')

SHOWNIL FONT | SPACE | POINT

Dieses Kommando ist ab der Version 4.0 weitgehend überflüssig, weil UNICODE-Daten im Original-Modus editiert werden können. Aus Kompatibilitätsgründen ist das Kommando jedoch weiterhin zulässig. Es ist jedoch zu beachten, dass die Bedeutung des Unicodes-Strings U'1234' geändert wurde: Bisher wurde ein Zeichen in ein UNICODE-Zeichen umgewandelt, z.B. `U'0'` wurde als `X'0030'` interpretiert. Ab der Version 4.0 werden in einem U-String jeweils die 4 Halbbytes eines UNICODE-Zeichens angegeben, z.B. `U'0030'` wird in das UNICODE-Zeichen `'0'` umgewandelt.

NIL-Zeichen in den Daten werden entsprechend den Vorgaben im Zeichensatz mit einem bestimmten Ersatzzeichen, in der Regel ein ausgefülltes Rechteck, dargestellt, während vom Satzende bis zum rechten Bildschirmrand ein echtes NIL-Zeichen angezeigt wird.

Falls in einem Arbeitsbereich Daten mit Unicode-Codierung enthalten sind, bei denen jedes abdruckbare Zeichen 2 Bytes belegt (1. Byte in der Regel X'00'), kann diese Darstellung störend sein.

FONT — Darstellung des Nil-Zeichens, wie im Zeichensatz. Diese Einstellung ist Standard.

SPACE — Darstellung des Nil-Zeichen als Leerzeichen.

POINT — Darstellung des Nil-Zeichens als Punkt, wie in der Parameterdatei definiert, siehe `Menü / Optionen / Sonderzeichen2` (S. 121).

Hinweis:

Die Eingabe der **Leertaste** erzeugt bei der Option SPACE und POINT ein NIL-Zeichen X'00', während die Leertaste mit der Umschalttaste ein Blank (X'20', bzw. X'40' in EBCDIC) erzeugt. Dies erleichtert die Eingabe von Unicode-Strings wesentlich.

In allen EDT-Kommandos können auch Unicode-Strings (S. 341) verwendet werden.

Beispiel: `on&find u'suchstring'.`

Arbeitsbereich sortieren

SORT [*rng*] [*sortcol* [, *sortcol.....*]] [A | D | [I] AI | DI]

SORT REVERSE

Der angegebene Zeilenbereich bzw. alle Zeilen des Arbeitsbereichs werden sortiert. Nach dem Kommando kann der ursprüngliche Zustand nicht durch `UNDO` wieder hergestellt werden.

rng — Zeilenbereich. Wird kein Zeilenbereich angegeben, werden alle Datensätze sortiert.

sortcol — :*cl1* [-*cl2*] [A| D | I | AI | DI] oder
:*cl1* [-*cl2*]: [A| D | I] AI | DI

Es sind max. 32 Spaltenbereiche zulässig. Zu jedem Spaltenbereich kann wahlweise die Sortierreihenfolge angegeben werden.

cl1 [-*cl2*] — Spaltenbereich, dessen Zeichen zur Sortierung berücksichtigt werden sollen, bestehend aus einer einzelnen Spalte (z.B. 10-10) oder einem zusammenhängenden Spaltenbereich (z.B. 15-25)

Wird nur eine Spaltennummer angegeben, werden die Zeichen ab dieser Spalte bis zum Ende der Zeile zur Sortierung berücksichtigt.

Die zweite Spaltenangabe darf nicht kleiner als die erste sein, kann aber größer sein als die tatsächliche Länge der Zeile.

Wird kein Spaltenbereich angegeben, wird die gesamte Zeile zur Sortierung berücksichtigt.

A — Ascending. Aufsteigende Sortierreihenfolge (Standardeinstellung).

D — Descending. Absteigende Sortierreihenfolge.

I — Insensitively. Die Sortierung erfolgt unabhängig von der Klein- / Großschreibung, d. h. Klein- und Großbuchstaben werden für die Ermittlung der Sortierreihenfolge gleich behandelt. Falls keine Sortierreihenfolge angegeben wird gilt A (aufsteigend).

AI — Kombination A und I.

DI — Kombination D und I.

Beispiele:

```
sort i
sort &:1-5
sort :1-5
sort &:#i1-#i2,:5-7di,:2-3
sort #l1.-#l2:5-10d,:13-15i,:20-25d
```

SORT REVERSE — Alle Zeilen im Arbeitsbereich werden in umgekehrter Reihenfolge sortiert, d.h. die letzte Zeile ist die erste Zeile, die vorletzte Zeile ist die 2. Zeile usw.

Hinweis:

Falls Sie Zeilen mit dem gleichen Inhalt bzw. Zeilen, die nur in bestimmten Spalten den gleichen Inhalt enthalten, löschen wollen, können Sie das Kommando `DELETE ... MULTIPLE` (S. **187**) verwenden.

Datenfenster aufteilen

SPLIT *arb2* [*/ view.*] [, *arb3* [*/view*]] [H | V]

Im EDT-Datenfenster können die Arbeitsbereiche in beliebiger Reihenfolge und Größe positioniert werden. In bestimmten Situationen kann es notwendig sein, dass zwei oder drei Arbeitsbereiche, z.B. zum Vergleichen, nebeneinander oder untereinander positioniert werden sollen. Dies ist auch mit dem Menübefehl `Fenster / Nebeneinander` oder `Fenster / Untereinander` möglich. Befinden sich aber noch andere Arbeitsbereiche im Datenfenster, werden alle Arbeitsbereiche nebeneinander oder untereinander positioniert. In diesen Fällen können mit dem Kommando SPLIT 2 oder 3 Arbeitsbereiche für die Positionierung ausgewählt werden. Der aktuelle Arbeitsbereich wird automatisch mit einbezogen.

arb2 [*/ view.*] — Zweiter Arbeitsbereich, ggf. mit Bestimmung einer View (S. **260**).

arb3 [*/ view.*] — Dritter Arbeitsbereich, ggf. mit Bestimmung einer View (S. **260**).

H — Das Datenfenster wird horizontal aufgeteilt (Standard).

V — Das Datenfenster wird vertikal aufgeteilt.

Beispiel:

```
split 2
```

das Datenfenster wird horizontal (Standard) in zwei gleich große Teile aufgeteilt, in denen der aktueller Arbeitsbereich und der Arbeitsbereich 2 dargestellt werden.

```
split 3,4/2v
```

das Datenfenster wird vertikal in drei gleich große Teile aufgeteilt, in denen der aktuelle Arbeitsbereich, sowie die Arbeitsbereiche 3 und von Arbeitsbereich 4 die View 2 dargestellt werden.

Inhalt der Variablen anzeigen / speichern

STATUS [I | F | S | L | P | M] [,F=*file* | M=*memoryset*]

I | F | S | L

Der Inhalt aller Integer-Variablen (I), Float-Variablen (F), String-Variablen (S) und Line-Variablen (L) wird in den Protokoll-Arbeitsbereich 32 geschrieben. Es werden nur Variable angezeigt, die einen Inhalt haben, d.h. Integer-Variable und Line-Variable ungleich 0 und String-Variable mit einer Länge > 0.

Ohne Angabe von Parametern werden alle vier Arten von Variablen angezeigt. Es kann eine beliebige Kombination der Buchstaben I, F, S oder L direkt hintereinander geschrieben werden, z.B. `sta l` oder `sta il`

P

Der Inhalt des Schleifensymbols (aktuelle Zeilennummer, Beginn, Ende und Schrittweite) sowie der symbolischen Zeilennummern (*,%,$,?) wird in den Protokoll-Arbeitsbereich 32 geschrieben.

M

Es werden alle Namen der mit dem Kommando `VAR` (S. **258**) erstellten Memorysets von gesicherten Variablen angezeigt.

file

Dateiname als String (S. **341**) von mit dem Kommando `VAR` (S. **258**) gesicherten Variablen. Der Dateiname kann direkt als String in Hochkommas oder indirekt in Form einer Zeilennummer, Zeilennummer-Variablen oder Stringvariablen angegeben werden.

memoryset

Name eines Speicherbereichs als String (S. **341**) von mit dem Kommando `VAR` (S. **258**) gesicherten Variablen. Der Name kann direkt als String in Hochkommas oder indirekt in Form einer Zeilennummer, Zeilennummer-Variablen oder Stringvariablen angegeben werden.

Beispiel: `STA ISLP`

```
Line - Variables
#L01=0050.4500
Integer - Variables
#I01=+0000000005 #I02=+0000000006
Float - Variables
#F01=+00000000000000000123.45678900000000000000 =
+1.23456789000000000000E+002 = 0B0BEE073CDD5E40
String - Variables
#S01(0004)=var1
#S02(0004)=var2
Procedure - Variables
(1) loop: act=0005.0000 beg=0001.0000 end=0101.0000
                                      step=+00010000
(1) *=0001.0000   %=0001.0000   $=0101.0000 ?=0000.0000
(0) *=0001.0000   %=0001.0000   $=0201.0000 ?=0000.0000
```

FSEND-Header anzeigen

STATUS FSEND

Inhalt des Datei-Headers von fernen BS2000-Dateien anzeigen, die über das Programm FServer oder den Browser eingelesen wurden. Weitere Informationen siehe Kommando `WRITE` (S. **270**).

Beispiel: `STA FSEND`

```
VERSION=1.0.0.0
SENDPORT=4109
IPADDR=194.200.100.20
FILE=$TEST.TEST
HOST=TESTHOST
CODE=EDF03IRV
ID=1720786481
LASTUPD=20050628104241
RECFORM=F
ISAM=N
BUFLEN=0
```

Leerzeichen und Tabulatoren entfernen

STRIP [*rng* | *str-var* [*-str-var*]]**L**EFT | **R**IGHT | **B**OTH]

Linksbündige bzw. rechtsbündige Leerzeichen und Tabulatorzeichen werden entfernt.

rng — Zeilenbereich. Wenn kein Zeilenbereich angegeben ist, wird der gesamte Arbeitsbereich behandelt.

str-var [*-str-var*] — String-Variable oder Bereich von String-Variablen.

LEFT — Löschen der Leerzeichen und Tabulatorzeichen am Anfang der Zeile.

RIGHT — Löschen der Leerzeichen und Tabulatorzeichen am Ende der Zeile (Standard).

BOTH — Löschen der Leerzeichen und Tabulatorzeichen am Anfang und am Ende der Zeile.

Beispiele:

```
strip 1-200 b
strip #l1 l
strip #s1-#s3 b
```

Leerzeichen in Tabulatorzeichen umwandeln

STT [*rng* — Space to tab: Alle Leerzeichen, dies sich in Abhängigkeit der aktuellen Tabulatorschrittweite (siehe Menü `Optionen / Tabulator` (S. **108**)) vor der Tabulatorposition befinden, werden durch ein Tabulatorzeichen ersetzt. Die Daten werden nicht verschoben.

STT [*rng* [A] — All Space to one tab. Ein oder mehrere aufeinander folgende Leerzeichen werden in ein Tabulatorzeichen umgewandelt, unabhängig von der Position in der Zeile und der Tabulatorschrittweite. Die Daten können sich verschieben.

Mit dem Kommando TTS (S. 254) können die Tabulatorzeichen wieder in Leerzeichen umgewandelt werden.

Beispiel (Tabulatorschrittweite = 4):

```
12345678901234567890  Spalte
a   b    c      d     Vor Kommando STT
a>  b>   c> >   d     Nach Kommando STT
a>  b>   c>  d       Nach Kommando STT A
```

Zeichenfolge am Zeilenende einfügen

SUFFIX *rng* | *str-var* **W**ITH *str* Am Zeilenende wird die Zeichenfolge *str* angefügt.

Symbole definieren

SYMBOLS [,] [**A**STERISK '*spec1*'] [,] [**S**LASH [='*spec2*']

Mit diesem Kommando können die Jokersymbole Pattern für mehrere Zeichen (Standard = '*') und Pattern für ein Zeichen (Standard = '/') zur Angabe des Suchbegriffs für das Kommando ON (S. 210) auf andere Zeichen umdefiniert werden (z.B. um nach den Zeichen * und / zu suchen). Die Einstellungen in der INI-Datei bzw. die im Menü Optionen / Einstellungen / Sonderzeichen (S. 119) definierten Zeichen werden damit nicht geändert. Die Änderung ist nur für den aktuellen Programmlauf wirksam.

spec1 Musterzeichen zum Ersatz einer beliebig langen, auch leeren Zeichenfolge. Es wird durch die kürzeste mögliche Teilkette der überprüften Zeile befriedigt. Dieses Zeichen kann in dem Suchstring der ON-Kommandos angegeben werden.

spec2 Jokersymbol zum Ersatz genau eines Zeichens. Dieses Zeichen kann in dem Suchstring der ON-Kommandos angegeben werden.

Wird das Kommando ohne Parameter aufgerufen, werden die Standardwerte definiert (ASTERISK='*', SLASH='/')

MS-DOS-Kommando ausführen

SYSTEM ['*cmd*' | *str-var*] [,W|NW] [,P|NP] [,MIN|MAX|RES] [,NC]

oder

!['*cmd*' | *str-var*] [,W|NW] [,P|NP] [,MIN|MAX|RES] [,NC]

MS-DOS-Kommando bzw. WINDOWS-Programm ausführen. Nach Ausführung des Kommandos wird das Programm EDT fortgesetzt.

cmd Beliebiges MS-DOS-Kommando.

str-var String-Variable, die das MS-DOS-Kommando enthält.

Ist *cmd* bzw. *str-var* nicht angegeben, wird in das Betriebssystem verzweigt. Das Betriebssystem wird durch das Programm aufgerufen, dessen Name in der Variablen COMSPEC steht. Ist diese Variable nicht vorhanden, wird c:\command.com verwendet. Rückkehr in das Programm EDT erfolgt in der Regel mit dem MS-DOS-Kommando exit.

Die folgenden Optionen sind nur interessant, wenn das Kommando in EDT-Prozeduren verwendet wird:

W — Wait: Das Kommando wird synchron ausgeführt, d.h. EDT wartet, bis das Kommando ausgeführt ist, bzw. bis das Kommando EXIT eingegeben wird.

NW — Not Wait: Das Kommando wird asynchron ausgeführt. Nach dem Starten des Kommandos wird das Programm fortgesetzt.

P — Pause: Nach Ausführung des Kommandos *cmd* wird die Meldung „Weiter mit beliebiger Taste ...“ ausgegeben und das MS-DOS-Kommando PAUSE aufgerufen. Nach Betätigen einer beliebigen Taste wird das MS-DOS-Fenster geschlossen.

NP — No Pause: Nach Ausführung des Kommandos *cmd* wird das MS-DOS-Fenster geschlossen.

MIN — Das Fenster für das Kommando bzw. Programm wird zum Icon verkleinert.

MAX — Das Fenster für das Kommando bzw. Programm wird in der maximalen Größe angezeigt.

RES — Das Fenster für das Kommando bzw. Programm wird in der zuletzt festgelegten Größe angezeigt.

NC — No Command. Das Kommando wird ohne das Kommando, das mit dem Menü `OPTIONEN / SHELL` definiert ist, gestartet. Diese Option wirkt nur bei WINDOWS-NT/2000 und hat zur Folge, dass keine DOS-Box erzeugt wird. In diesem Fall können als Kommando nur ausführbare Dateien angegeben werden, die in der Variablen PATHEXT enthalten sind, wie z.B. *.cmd, *.bat *.exe usw. Der Name der ausführbaren Datei muss in diesem Fall immer einschließlich der Extension angegeben werden.

Ist *cmd* nicht angegeben, muss nach dem Kommandonamen SYS bzw. ! ein Komma angegeben werden (z.B. SYS,W).

Ohne Angabe der Optionen werden folgende Einstellungen verwendet:

	Dialog	Batch
SYS	P,NW,RES	NP,W,RES
SYS '*cmd*' | *str-var*	P,NW,RES	NP,W,RES

Tabulatoren definieren

TABS [: : *tab* :] *cl1* [, *cl2,...*]

Tabulator mit bis zu acht Positionen (Spalten) definieren. Die Spalten müssen aufsteigend angegeben werden. Die Positionierung erfolgt sofort mit Eingabe des Tabulatorzeichens. Falls eine bereits bestehende Zeile geändert wird und das Tabulatorzeichen eingegeben wird, bleibt der bisherige Inhalt erhalten. Die Tabulator-Einstellungen gelten nur für den aktuellen Arbeitsbereich.

TABS *cl1*[,*cl2,...*] — Wird ::*tab*: nicht angegeben, beziehen sich die Werte auf den Hardware-Tabulator (Taste `<Tab_right>` und `<Tab_left>`). Zur Hervorhebung dieses besonderen Editiermodus werden die Tabulator-Spalten (Felder) durch vertikale Striche getrennt. Mit dem Kommando `TABS OFF` oder der Tastenkombination `<Strg> <T>` kann der Tabulator ausgeschaltet werden, ohne dass die Positionen verloren gehen. Mit dem Kommando `TABS ON` oder der Tastenkombination `<Strg> <T>` kann der Tabulator wieder eingeschaltet werden.

Beim Einfügen und Löschen werden die Daten nur innerhalb einer Tabulator-Spalte (eines Feldes) verschoben. Dies gilt aber nur beim Erstellen und Ändern von Daten über die Tastatur bzw. Maus im Datenbereich. Werden Daten durch Kommandos geändert, z.B. Kommando `PREFIX ON&CHANGE` usw., so wird immer der ganze Satz verschoben.

tab — Beliebiges Zeichen, das als Tabulatorzeichen interpretiert wird.

cl1,cl2.. — Spalten, auf die mit dem Tabulatorzeichen positioniert werden soll.

TABS RANGE[=*range*] — Die Software-Tabulatorzeichen werden im angegebenen Zeilenbereich entsprechend der aktuellen Definition ausgewertet. Fehlt *range*, so werden alle Zeilen der Datei bearbeitet.

TABS ON — Falls in der gleichen oder einer vorherigen Anweisung schon Positionen für Tabulatoren definiert sind, wird die Funktion eingeschaltet, d.h. durch die Taste `<Tab_right>` positioniert sich die Schreibmarke auf die nächste definierte Spalte. Der Standardwert ist ON. Das Einschalten ist auch mit der Tastenkombination `<Strg> <T>` möglich.

TABS OFF — Die Funktion des Tabulators wird ausgeschaltet. Die definierten Positionen bleiben erhalten und können durch `TABS ON` wieder aktiviert werden. Das Ausschalten ist auch mit der Tastenkombination `<Strg> <T>` möglich.

TABS oder **T**ABS**::** — Die zuletzt gültige Tabulator-Definition wird gelöscht.

Menüzeile: `Optionen / Tabulator` (S. **108**)

Beispiel:

`tabs ::#:10,16,40,72`	Tabulatorzeichen = # Spalten 10, 16, 40 und 72
`tabs 10,16,40,72`	Tabulatorz. = <`Tab_right`> Spalten 10, 16, 40 und 72

Trace-Funktionen

TRACE ON | OFF — Trace-Funktionen ein- ausschalten. Der Umfang der Trace-Protokollierung und der Name der Trace-Datei kann mit dem Menü `Optionen / Einstellungen / Protokoll Trace-Funktionen` (S. **124**) eingestellt werden.

Tabulatorzeichen in Leerzeichen umwandeln

TTS [*rng*] — Tab to space: Alle Tabulatorzeichen werden durch entsprechend viele Leerzeichen ersetzt. Mit dem Kommando `STT` (S. **251**) können die Leerzeichen wieder in Tabulatorzeichen umgewandelt werden, z.B. zum Ändern der Tabulatorschrittweite, ohne dass sich die Daten verschieben.

Arbeitsschritte zurücknehmen

UNDO [ON | OFF | ?] — Mit dem Kommando `UNDO` werden vorausgegangene Kommandos oder Eingaben in der Markierungsspalte bzw. im Datenbereich wieder rückgängig gemacht. Die wiederholte Eingabe des UNDO-Kommandos macht die letzte, vorletzte, vorvorletzte usw. Änderung rückgängig. Die UNDO-Funktion wirkt nicht nach den Kommandos SORT, DROP, FSTAT, COMP, REFORMAT, UNFORMAT.

Die Anzahl der maximal möglichen Undo's kann definiert werden (siehe Menüzeile `Optionen` / `verschiedene Optionen` (S. **127**)). Wird in diesem Parameter als Wert "0" angegeben, so ist die UNDO-Funktion ausgeschaltet.

Menüzeile: `Bearbeiten / Rückgängig` (S. **66**)

Symbol 

ON

Nach einem vorangegangen Kommando UNDO OFF bzw. nach dem automatischen Deaktivieren in einer Prozedur wird die UNDO-Funktion wieder aktiviert.

OFF

Die aktuellen UNDO-Informationen werden gelöscht. Nachfolgende Aktionen können nicht mehr rückgängig gemacht werden. Im Prozedurmodus (Schalte -i beim Laden) ist die UNDO-Funktion automatisch deaktiviert.

Es ist zu beachten, dass für die UNDO-Informationen teilweise viel Speicher benötigt wird. Wenn z.B. alle Sätze mit dem Kommando ON&CHANGE geändert werden, verdoppelt sich der Speicherbedarf, weil alle Sätze zweimal vorhanden sind. Die UNDO-Informationen werden erst automatisch beim Schließen des Arbeitsbereich-Fensters bzw. nach dem Kommando READ in einem leeren Arbeitsbereich gelöscht. Auch nach einem Löschen aller Zeilen eines Arbeitsbereichs sind also noch die UNDO-Informationen verfügbar.

?

Es wird ein Menü mit den letzten Operationen, die rückgängig gemacht werden können, angezeigt.

```
EDT - UNDO - Funktion
* Marks: C5.01 C5.02 A5.03
  del 1-2
  Marks: X5.00
  Marks: X4.00
  Marks: I3.00
```

Die zeitlich letzte Aktion wird in der ersten Zeile dargestellt. Die Eintragungen im obigen Beispiel haben folgende Bedeutung:

Zeile 1: Markierung C Zeilen 5.01 und 5.02, Markierung A Zeile 5.03
Zeile 2: Kommando DEL 1-2
Zeile 3: Markierung X Zeile 5.00
Zeile 4: Markierung X Zeile 4.00
Zeile 5: Markierung I Zeile 3.00

ASCII-Format in fremde Satzstruktur oder UTF-8 umwandeln

UNFORMAT

Umwandeln einer mit dem Kommando `REFORMAT` formatierten Datei in das ursprüngliche Format. Der gesamte satzstrukturierte Arbeitsbereich wird in die fremde Dateistruktur umgewandelt und im Binärmodus angezeigt (zu erkennen an der Anzeige eines Sterns in der Zeilennummer). Siehe hierzu auch die Beschreibung des Kommandos REFORMAT auf S. **234**. Ausnahme: UTF muss auch angegeben werden, wenn zuvor das Kommando REFORMAT UTF verwendet wurde.

UNFORMAT { RS *n* | LF | LF2 | LF2+[*hhhh*] | LF4 | DOS | UNIX | BS2 | ST *str* } [(*n*)] } [H] }

UNFORMAT UTF [, *esc*]

Umformatieren einer ASCII/ANSI-Datei in ein fremdes Format. Der gesamte satzstrukturierte Arbeitsbereich wird in die fremde Dateistruktur umgewandelt und im Binärmodus angezeigt (zu erkennen an der Anzeige eines Sterns in der Zeilennummer). Siehe hierzu auch die Beschreibung des Kommandos REFORMAT auf S. **234**.

Datei mit Sätze fester Länge

RS *n*

Record-Size. Nach dem Schlüsselwort RS ist die Satzlänge anzugeben. Die Daten des gesamten Arbeitsbereichs werden in eine Binärdatei ohne Satz-Trennzeichen umgewandelt.

Die bisherigen Sätze der ASCII-Datei müssen die Länge *n* haben, sonst wird das Kommando abgebrochen. Nötigenfalls sind die Daten bis zu der angegebenen Länge mit Leerzeichen aufzufüllen.

Datei mit variabler Satzlänge und Satzlängenfeld

LF2

Length-Field - Länge 2. Die Datei soll so organisiert werden, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Ein Satz mit einem Längenfeld von X'0005' enthält z.B. noch drei Nutzbytes. Sätze dieses Formats werden von verschiedenen Textsystemen verwendet (DCA-Format). Jedem Satz wird ein zwei Byte langes Satzlängenfeld vorangestellt, das Satz-Trennzeichen wird gelöscht.

Zur Kennzeichnung, dass es sich um ein solches Format handelt, wird am Anfang der Datei ein Header-Satz mit dem Inhalt "EDTWFILELENGTHFIELD" erzeugt. Nach diesem String enthält der Satz den Buchstaben "E" für EBCDIC und "A" für ANSI zur Kennzeichnung der Codierung. Sobald eine solche Datei eingelesen wird, wird automatisch das Kommando REFORMAT LF und ggf. CODE EBC ausgeführt. Der Header-Satz wird nicht in den Arbeitsbereich übernommen.

LF2H

Length-Field - Länge 2 mit Header-Satz. Als 1. Satz wird den Daten ein Header-Satz vorangestellt. Beim Einlesen einer solchen Datei mit dem Kommando READ (S. **220**) wird automatisch das Kommando REFORMAT (S. **234**) und ggf. CODE (S. **181**) ausgeführt.

LF

Length-Field - Länge 2. "LF" hat die gleiche Bedeutung wie LF2 und wird aus Gründen der Kompatibilität weiterhin unterstützt.

LF2+[*hhhh*]

Length-Field - Länge 2 + 2 Byte X'*hhhh*'. Die Datei soll so organisiert werden, dass in den ersten zwei Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Danach folgen 2 Bytes mit dem Wert X'*hhhh*'. Fehlt die Angabe *hhhh*, wird X'4040' benutzt.

LF4

Length-Field - Länge 4. Die Datei soll so organisiert werden, dass in den ersten vier Bytes die Länge des Satzes (einschließlich des Satzlängenfeldes) steht. Dieses Format wird z.B. in BS2000-ZIP-Archiven benutzt, die nicht im kompatiblen Format erzeugt wurden.

Datei mit anderem Satz-Trennzeichen

ST *str*

Das Satz-Trennzeichen der Datei ist nach dem Schlüsselwort ST anzugeben. Die Zeichenfolge kann als Character-String, als Hexadezimal-String oder als String-Variable angegeben werden. Das ASCII-Satz-Trennzeichen wird durch das in der Zeichenfolge *str* angegebene neue Satz-Trennzeichen ersetzt.

UNIX

Der aktuelle Arbeitsbereich wird in das UNIX-Dateiformat mit den Satz-Trennzeichen X'0A' umgewandelt. Diese Option hat die gleiche Wirkung wie `ST X'0A'`.

DOS

Der aktuelle Arbeitsbereich wird in das MS-DOS-Dateiformat mit den Satz-Trennzeichen x'0D0A' umgewandelt. Diese Option hat die gleiche Wirkung wie `ST X'0D0A'`.

UTF-8 Daten

UTF [, *esc*]

Konvertierung eines Arbeitsbereichs mit Daten in ASCII-Codierung (ISO8859) zu Daten im UTF-8-Format. Es werden max. 2 Byte lange UNICODE-Zeichen (UCS-2, 16 Bit) unterstützt. Alle Zeichen mit einem UNICODE-Wert von 256 - 65.535 müssen als 6-Byte-Folge in der Form %Uxxxx (xxxx = 0100 bis FFFF) dargestellt sein. Das Zeichen "%" kann durch Angabe eines anderen Zeichens *esc* geändert werden, z.B. `REFORMAT UTF,*` (UNICODE-Zeichen sind in der Form *Uxxxx geschrieben).

Beispiel:

```
unformat
unformat UNIX
unformat lf
unformat st '**]'
unformat st x'0509'
unformat utf
unformat utf,*
```

Datei löschen

UNSAVE '*file*' | *str-var*

oder

ERASE '*file*' | *str-var*

UNSAVE '*ft* R=*remote-file* [LP=lpass] [FP=fpass] ' oder

ERASE '*ft* R=*remote-file* [LP=lpass] [FP=fpass] '

Lokale Datei:

file

Dateiname. Der Name muss in Hochkommas eingeschlossen werden.

str-var String-Variable, die den Dateinamen enthält.

Entfernte Datei:

ft Name eines Filetransfer-Profils, das mit dem Menübefehl `Extras/Filetransfer...` (S. 136) definiert werden kann. Es ist nur ein Filetransfer-Profil mit Typ FTP_UNIX, FTP_POSIX, FTP_BS2, FTP_MVS und FTP_OMVS zulässig. Aus den Parametern des Kommandos und den Angaben aus dem Profil wird über die interne FTP-Schnittstelle die Löschung der entfernten Datei gestartet.

remote-file Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID bzw. Pfadnamen.

lpass|? Logon-Passwort für den Zugang zum entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *lpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

fpass|? File-Passwort im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

Schreiben aller Arbeitsbereiche

UPDATE Es werden alle geänderten Arbeitsbereiche zurückgeschrieben. Vor jedem Schreiben wird eine Bestätigung angefordert. Das Kommando hat die gleiche Wirkung wie die Schaltfläche  .

Kleinbuchstaben in Großbuchstaben umwandeln

UPPER *rngcol*|*str-var*[*-str-var*]

Alle Kleinbuchstaben im Bereich *rngcol* bzw. in der oder den String-Variablen *str-var* werden in Großbuchstaben umgewandelt. Es werden auch alle Umlaute und sprachabhängige Sonderzeichen, wie z. B. â, á à usw. umgesetzt. Die Umsetzung erfolgt in Abhängigkeit von den Definitionen in der Datei `CODEPAGE.TXT` (siehe Kapitel 17 Code-Tabellen (S. 359)). Mit dem Kommando LOW kann man Großbuchstaben in Kleinbuchstaben umwandeln.

Variable sichern und laden

VAR SAVE { F=*file* | M=*memoryset* } [,U[*var-save*]

Die Variablen *var-save* bzw. alle Variablen (String-, Integer-, Float- und Line-Variable) werden in einen Speicherbereich oder in eine Datei gesichert.

Stringvariablen werden ab EDTW-Version 4.0 im UNICODE-Format gespeichert. Variablen, die in eine Datei gespeichert werden, können von einer EDTW-Version < 4.0 nicht verarbeitet werden.

VAR LOAD { F=*file* | M=*memoryset* } [,*var-load*]

Die Variablen *var-load* bzw. alle Variablen (String-, Integer-, Float- und Line-Variable) werden aus einem Speicherbereich oder aus einer Datei gelesen und in die Variablen geschrieben. Dabei ist es auch möglich, die Daten in andere Variable als beim Sichern zu laden, z.B. #S1-10 werden in #S11-#20 geladen.

Stringvariablen werden ab EDTW-Version 4.0 im UNICODE-Format gespeichert. Eine EDTW-Version ab 4.0 kann jedoch auch Dateien verarbeiten, in der die String-Variablen im alten Format gespeichert wurden.

VAR RELEASE { ALL | *memoryset* }

Alle Speicherbereich (ALL) oder Speicherbereich für ein *memoryset* freigegeben.

file

Dateiname als String (S. **341**). Der Dateiname kann direkt als String in Hochkommas oder indirekt in Form einer Zeilennummer, Zeilennummer-Variablen oder Stringvariablen angegeben werden.

memoryset

Name eines Speicherbereichs als String (S. **341**). Der Name kann direkt als String in Hochkommas oder indirekt in Form einer Zeilennummer, Zeilennummer-Variablen oder Stringvariablen angegeben werden.

U

Die in der Datei bzw. in dem Memoryset bereits gespeicherten Variablen bleiben erhalten und werden durch die zu sichernden Variablen aktualisiert bzw. ergänzt. Ohne diese Option wird die gesamte Datei bzw. das Memoryset überschrieben.

var-save

var-range [,*var-range....*]

Eine oder mehrere Variablenbereiche, die gesichert werden sollen. Fehlt diese Angabe, gilt die Aktion für alle Variablen.

var-load

var-range [> *varnum*] [, *var-range* [> *varnum*]....]

Eine oder mehrere Variablenbereiche, die geladen werden sollen. Fehlt diese Angabe, gilt die Aktion für alle Variablen. Wahlweise können die Daten in andere Variable geladen werden. In diesem Fall ist nach dem Zeichen ">" die erste Nummer der Variablen des Zielbereichs anzugeben, z.B. #s10-20>30 bedeutet, dass die Stringvariablen #S10 - #S20 in die Variablen #S30 - #S40 geladen werden.

var-range

I | F | L | S | *var* | *var-n*

Eine oder mehrere Variabel, z.B. `i1-5, #s1` oder `#l1-90`. Das Zeichen "#" kann wahlweise angegeben werden.

I Alle Integer-Variablen
F Alle Float-Variablen
L Alle Line-Variablen
S Alle String-Variablen

var

[#] I*n* | F*n* I L*n* I S*n*

Der Name einer Variablen, z.B. `i1, #s10, #l55`. Das Zeichen "#" kann wahlweise angegeben werden.

var-n

Gruppe von Variablen, z.B. `#s1-5` oder `s1-5`

varnum Nummer der ersten Variablen des Zielbereichs, falls beim Laden die Daten in andere Variable geladen werden sollen.

Hinweis:

Das Anzeigen der gesicherten Variablen ist mit dem Kommando `STA` (S. **250**) möglich.

Beispiele:

```
var save m='mem1'
var save f='var1',i
var save f=#s1,i1-5,s10-20,l
var load m=#l1,s1-10>20
var load f=#s10:1-20:,i10-20,i21>91
var load f=1:1-50:,i,f
```

View für einen Arbeitsbereich aktivieren

VIEW *n* Die aktuellen Einstellungen des Arbeitsbereichs werden unter der View-Nummer *n* gespeichert. Damit können bis zu 8 (1-8) verschiedene Sichten des Arbeitsbereichs definiert werden. Für jede View wird ein eigenes Arbeitsfenster erzeugt. Für alle Views gilt die gleiche Datenbasis; werden Daten in einer View geändert, so werden in den anderen Views ebenfalls die geänderten Daten angezeigt.

Folgende Eigenschaften werden für jede Sicht gespeichert:

- Maus-Markierungen
- Position im Arbeitsbereich (erste Zeile und erste Spalte);
- Tabulatorangaben aus Kommando `TABS`;
- Full-Modus on/off (Kommando `EDIT FULL`);
- Hexa-Modus on/off (Kommando `HEX`);
- Scale-Modus on/off (Kommando `SCALE`);
- Index-Modus on/off (Kommando `INDEX`);
- Long-Modus on/off (Kommando `EDIT LONG`);
- Low-Modus on/off (Kommando `LOW`);
- Treffer aus Kommando `ON ...FIND`: Es werden sowohl die Zeile als auch die Positionen des gefundenen Suchstrings in der Zeile gespeichert.

n View-Nummer von 1 bis 8.

Wird eine View angegeben, die bisher noch nicht verwendet wurde, so wird ein neues Arbeitsbereichs-Fenster erzeugt. Die aktuellen Attribute werden der neuen View zugewiesen und es wird der Beginn des Arbeitsbereichs angezeigt. Die View-Nummer 1 ist für die Standard-View reserviert. Für eine neue View kann daher nur 2 bis 8 verwendet werden. Die View-Nummer wird in der Titelzeile und in den Reitern angezeigt.

Wird die Nummer einer bereits bestehenden View angegeben, so wird das entsprechende Arbeitsfenster mit den individuellen Einstellungen, Maus-Markierungen, aktueller Zeilennummer usw. aktiviert.

Die View-Nummer wird in der Titelzeile des Arbeitsfensters hinter dem Dateinamen angezeigt, z.B: `<2> testfile:3` bedeutet Arbeitsbereich 2, Datei `"testfile"`, View 3. In den Reitern für die Arbeitsbereiche wird die View-Nummer für die Views 2-8 in der Form `<2:3>` angezeigt. Wird die View-Nummer nicht angezeigt, bedeutet dies, dass nur die Standard-View 1 vorhanden ist.

Warten

WAIT [*n*]

Warten von *n* Sekunden. Fehlt die Angabe von Sekunden, wird das Programm 10 Sekunden unterbrochen. Dieses Kommando ist vor allem für Prozeduren gedacht.

Lizenz-Informationen und Version anzeigen

WHO

In einem Fenster werden die Lizenzinformationen und in der letzten Bildschirmzeile die EDT-Versions-Nummer angezeigt.

Arbeitsbereich in Datei schreiben

Format1: Schreiben lokal

WRITE ['*file* [**:***ads*]' | *zip(* [*path/*] *elem)* | *str-var*] [,] [*rngcol*] [**O**VERWRITE | **U**PDATE] [D | X] [KEY] [*code*]

WRITE ALL

WRITE STDOUT | STDERR

Daten aus dem aktuellen Arbeitsbereich in die Datei *file* schreiben. Bei fehlendem Dateinamen wird der zuletzt benutzte Dateiname des aktuellen Arbeitsbereichs verwendet, sofern ein Dateiname existiert. Falls die Datei im UNIX-Format (Satz-Trennzeichen X'0A') eingelesen wurde, wird bei fehlendem Dateinamen der Arbeitsbereich auch wieder im UNIX-Format zurückgeschrieben.

file — Dateiname. Der Name muss in Hochkommas eingeschlossen werden. Enthält der Dateinamen keinen Pfadnamen, wird als Pfad das Arbeitsverzeichnis benutzt (siehe auch Kommando `CHDIR` (S. **179**)).

ads — Die Daten des Arbeitsbereichs werden als "Alternate Data Streams (ADS)" geschrieben. Diese Erweiterung wird nur von Windows NT/2000 mit NTFS-Dateisystem unterstützt. Die ADS sind für den Explorer "unsichtbar". Eine Liste der ADS können Sie mit dem Kommando `FSTAT` (S. **197**) erzeugen.

Beispiel: `datei1:stream1`.

zip([*path/*] *elem)* — Element eines ZIP-Archivs. Der Name muss in Hochkommas eingeschlossen werden. Enthält der Dateinamen des ZIP-Archivs keinen Pfadnamen, wird als Pfad das Arbeitsverzeichnis benutzt (siehe auch Kommando `CHDIR` (S. **179**)). Vor dem Elementnamen kann wahlweise ein Verzeichnisname innerhalb des ZIP-Archivs angegeben werden. Ein bestehendes Element kann auch überschrieben werden.

Das Element wird in eine temporäre Datei in das TEMP-Verzeichnis geschrieben, in das ZIP-Archiv aufgenommen und sofort wieder gelöscht.

str-var — String-Variable, die den Dateinamen enthält.

ALL — Alle Arbeitsbereiche, bei denen Änderungen vorgenommen wurden, schreiben. Es werden nur die geänderten Arbeitsbereiche geschrieben. Mit dem Button  können ebenfalls alle Arbeitsbereiche gespeichert werden.

STDOUT | STDERR

Alle Daten des Arbeitsbereichs werden nach STDOUT bzw. STDERR geschrieben. Die Systemdatei wird immer ergänzt, d.h. es erfolgt immer ein Erweitern der Systemdatei wie bei der Option UPDATE. Beim Laden des EDTW muss die entsprechende Systemdatei in eine Datei umgelenkt werden.

Beispiele für die Umleitung:

```
edtw.exe > dat.stdout 2> dat.stderr
edtw.exe > dat.stdout | more
```

Auswahl von Zeilen und Spalten

rngcol

Zeilen - und/oder Spaltenbereich der zu schreibenden Zeilen. Wenn statt des Dateinamens eine String-Variable angegeben wurde, ist die String-Variable von der Bereichsangabe *rngcol* durch ein Komma zu trennen.

Überschreiben und Erweitern von Dateien

OVERWRITE

Diese Option bewirkt, dass eine bereits bestehende Datei ohne Rückfrage überschrieben wird.

UPDATE

Diese Option bewirkt, dass der Inhalt des Arbeitsbereichs an eine bereits bestehende Datei angehängt wird.

Satzende-Kennzeichen und Isam-Key

D

Eine Datei, die im UNIX-Format eingelesen wurde (Satz-Trennzeichen X'0A'), wird im MS-DOS-Format (Satz-Trennzeichen X'0D0A') zurückgeschrieben.

X

Eine Datei, die im MS-DOS-Format eingelesen wurde (Satz-Trennzeichen X'0D0A'), wird im UNIX-Format (Satz-Trennzeichen X'0A') zurückgeschrieben.

KEY

Beim Schreiben 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).

Codierung (UNICODE-Dateien)

code

CHAR | UTF | UCB | UCL [HEA | NOHEA]

UNICODE-Arbeitsbereiche werden automatisch mit der Codierung geschrieben, die in dem eingelesenen Headers bzw. bei der Option *code* des Kommandos READ angegeben wurde. Ein-Byte-Arbeitsbereiche (ANSI, EBCDIC usw.) werden mit der Option CHAR geschrieben. Die Angabe der Codierung ist dann nicht notwendig. Wird ein neu erstellter UNICODE-Arbeitsbereich geschrieben, wird über eine Dialogbox die gewünschte Codierung angefordert.

Die Angabe eines Codes bewirkt keine Code-Konvertierung, z.B. von EBCDIC zu UNICODE. Die Zeichen werden unverändert geschrieben, es geht nur darum, ob aus einem Zeichen 1 Byte (CHAR), variabel viele Bytes (UTF) oder 2 Bytes (UCB oder UCL) erstellt werden. Die Codes ANSI und UNICODE sind bis auf das EURO-Zeichen gleich. Soweit kein EURO-Zeichen vorkommt, kann ein ANSI-Arbeitsbereich ohne Konvertierung als UNICODE-Datei geschrieben werden.

CHAR

Datei im 1-Byte-Format (z.B. ANSI, ASCII, EBCDIC). Diese Format wird automatisch benutzt, falls es sich **nicht** um einen UNICODE-Arbeitsbereich handelt. Ein UNICODE-Arbeitsbereich kann nur mit CHAR geschrieben werden, falls keine Zeichen > 255 (Achtung: € = `U'20AC'`) enthalten sind.

UTF

Datei im UTF-8-Format. UTF-8 (Abk. für 8-Bit Unicode Transformation Format) ist die am weitesten verbreitete Codierung für Unicode-Zeichen. Dabei wird jedem Unicode-Zeichen eine speziell codierte Bytekette von variabler Länge zugeordnet. UTF-8 unterstützt bis zu vier Byte, auf die sich wie bei allen UTF-Formaten alle 1.114.112 Unicode-Zeichen abbilden lassen. Das UTF-8-Header enthält `X'EFBBBF'`

UCB

UTF-16BE Unicode Big-Endian (höherwertiges Byte zuerst). UTF-16 (Abk. für 16-bit Unicode Transformation Format) ist eine Codierung, bei der jedes Unicode-Zeichen mind. 16 Bit belegt. Unicode-Zeichen, deren Code sich nicht mit 16 Bit darstellen läßt, belegen zwei 16-Bit-Wörter (code units). Das UTF-16-Header enthält `X'FEFF'`.

Das höherwertige Byte steht in den ersten 8 Bits, z.B. Die Zahl "0" (`X'30'`) erhält den Wert `X'0030'`.

UCL

UTF-16LE Unicode Little-Endian (niederwertiges Byte zuerst). Das UTF-16-Header enthält `X'FFFE'`.

Das niederwertige Byte steht in den ersten 8 Bits, z.B. Die Zahl "0" (`X'30'`) erhält den Wert `X'3000'`.

HEA

Beim Schreiben Header voranstellen (Standard). Folgende Header werden erstellt:

UTF `X'EFBBBF''`
UCB `X'EFFF''`
UTL `X'FFFE''`

NOHEA

Datei ohne Header schreiben.

Hinweis:

Ist die Option für die automatische Erstellung einer Sicherungskopie akitviert (Menüzeile: Optionen / Sicherung (S. **106**)), so wird vor dem Zurückschreiben der Datei eine Sicherungskopie erstellt.

Beispiele:

```
write'datei1'
write'datei1'k
write'datei1'ucb
par varsubst=y
write 'temp!pid'
write #s1,1-500,800-900:10-20:
write 'datei1'1-500,800-900:10-20:.
write test.zip(verz1/test1.src)
```

Format 2: Schreiben entfernt nach openFT oder FTP

WRITE '*ft* | *batch* [L=*local-File*] R=*remote-file* [M=*mode*] [*authorisation*] [FP=*fpass*]
[CCS=*ccsname*] [*mvs-fileattr*] [*par1 par2 ....*] '
[**U**PDATE | **O**VERWRITE | NEW]

oder

WRITE [**U**PDATE | **O**VERWRITE | **N**EW] [M=*mode*]

oder

WRITE *str-var* [**U**PDATE | **O**VERWRITE | **N**EW]

str-var String-Variable, die alle Parameter innerhalb der Hochkommas enthält.

Mit diesem Format des WRITE-Kommandos wird zuerst eine lokale Datei geschrieben und diese dann entweder mit Hilfe einer Batch-Prozedur weiterverarbeitet oder mit dem internen Filetransfer des EDT zu einem entfernten Rechner übertragen. Für den Filetransfer ist keine zusätzliche Software auf dem PC notwendig.

Wird nach einem entfernten `READ` das Kommando `WRITE` ohne Parameter bzw. `WRITE O` eingegeben, werden alle FT-Angaben und der Dateiname aus dem `READ`-Kommando benutzt.

Filetransfer-Profil

ft Name eines Filetransfer-Profils, das mit dem Menübefehl `Extras/Filetransfer...` (S. 182) definiert werden kann. Aus den Parametern des Kommandos und den Angaben aus dem Profil wird die Schnittstelle des internen EDT-Filetransfers erzeugt und der Filetransfer gestartet.

Sind Profile vorhanden, die in mehreren INI-Dateien mit dem gleichen Namen enthalten sind, kann durch Voranstellen der Dateinummer (1, 2 oder 3) die entsprechende INI-Datei angegeben werden.

Beispiel:

`1std` = Profil `STD` aus der privaten INI-Datei
`2std` = Profil `STD` aus der INI-Datei vom Windowsverzeichnis
`3std` = Profil `STD` aus der INI-Datei vom Ladeverzeichnis.

batch Dateiname der Batchdatei. Die Extension ".BAT" muss hierbei nicht angegeben werden. Die Batch-Prozedur wird mit folgenden Parametern gestartet:

batch-file par1 par2 .....

Zusätzlich werden folgende Umgebungs-Variablen vor dem Aufruf der Batch-Prozedur gesetzt:

`call:`	`"W"`
`locfile:`	Name der lokalen Datei
`remfile:`	Name der entfernten Datei
`lpass:`	Logon-Passwort
`fpass:`	Datei-Passwort

Die Angabe von "W" in der Variablen `call` bedeutet, dass die Batch-file von dem Kommando WRITE aufgerufen wurde. Das Entfernen der lokalen Datei nach erfolgreicher Übertragung in das entfernte System ist in der Batchfile vorzunehmen (Beispiel siehe unten).

Lokale Datei

local-file — Dateiname der zu erzeugenden Datei auf dem lokalen Rechner. Ist dieser Parameter nicht angegeben, wird eine temporäre Datei erzeugt, die nach dem Filetransfer gelöscht wird.

Entfernte Datei

remote-file — *bs2-file* | *lib*([*typ*/]*elem*[/*version*]) | */posix-file* | *mvs-file*

bs2-file — Dateiname auf dem fernen Rechner, ggf. einschl. CAT-ID und User-ID. Es kann auch eine Generation aus einer Dateigenerationsgruppe in der BS2000-Syntax angegeben werden (z.B. `fgg(*0003))`.

lib — Name der PLAM-Bibliothek auf dem fernen Rechner.

typ — Typ des PLAM-Elements. Es sind nur die Angaben S, M, J, P und D zulässig. Wird kein Typ angegeben, so wird der Typ "S" als Standard benutzt.

elem — Name des PLAM-Elements. Delta-geführte Elemente können vom openFT-BS2000 nur gelesen werden.

version — Version des PLAM-Elements. Fehlt die Version, wird die höchste Version des Elements benutzt.

/posix-file — Beginnt der Dateinamen mit dem Zeichen "/", so wird die Datei automatisch von dem entsprechenden POSIX-Dateisystem gelesen. Es ist immer der Dateiname ab ROOT anzugeben. Beim Filetransfer mit FTP muss im Filetransfer-Profil der Typ FTP-POSIX ausgewählt werden.

mvs-file — *data-set* | *pds(member)* | `'hlq.data-set'` | `'hlq.pds(member)'`

Bei der Variante `read'profile r=mvsfile'` müssen die Hochkommas verdoppelt werden, weil der ganze Parameter bereits ein String in Hochkommas ist. Soweit alle Angaben in einer Stringvariablen stehen, dürfen die Hochkommas nicht verdoppelt werden. Soll z.B. eine Datei aus der FSTAT-Liste eingelesen werden, so muss lediglich der Inhalt aus der Spalte 1-256 in eine Stringvariable übertragen werden. Dann kann die Datei mit dem Befehl `read` *strvar* eingelesen werden.

Um die Angabe der doppelten Hochkommas zu umgehen, können auch folgende alternative Formate angegeben werden:

mvs:hlq.data-set | *mvs:hlq.pds(member)* oder
$hlq.data-set | *$hlq.pds(member)*

data-set — Dateiname auf dem fernen Rechner ohne `HLQ` (high level qualifier). Vom Host-System wird der Dataset-Name automatisch um den HLQ ergänzt.

pds(member) — Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended) ohne HLQ. Vom Host-System wird der Member-Name automatisch um den HLQ ergänzt.

`'hlq.data-set'` | *mvs:hlq.data-set* | *$hlq.data-set*

Vollqualifizierter Dateiname auf dem fernen Rechner. Aus diesem String wird für den Filetransfer der Dateiname `'hlq.data-set'` erzeugt.

`'hlq.pds(member)'` | *mvs:hlq.pds(member)* | *$hlq.pds(member)*

Vollqualifiziertes Member eines PDS (Partioned Data Set) oder PDSE (Partioned Data Set Extended). Aus diesem String wird für den Filetransfer der Dateiname `'hlq.pds(member)'erzeugt.`

Zugangsdaten

Die Zugangsdaten können im Filetransfer-Profil (Extras/Filetransfer.... (S. 182)) definiert werden oder bei den Kommandos `FILE` (S. 252), `READ` (S. 302) und `WRITE` (S. 353) angegeben werden. Es ist auch möglich, einen Teile der Zugangsdaten im Profil und einen Teil bei den Kommandos anzugeben.

authorisation [LH=host] [LU=user] [LA=account] [LP=lpass|?] oder TA=ta

host | ? Host-Name für den Zugang zum entfernten Rechner. Der Hostname kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

user | ? User-ID für den Zugang zum entfernten Rechner. Die User-ID kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

acount | ? Abrechnungs-Nummer für den Zugang zum entfernten Rechner. Die Abrechnungs-Nummer kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

lpass | ? Logon-Passwort für den Zugang zum entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *lpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

ta | ? Transfer-Admission des FTAC-Profils für den Zugang zum entfernten Rechner. FTAC-Profile können mit der Software FTAC erstellt werden. Die Transfer-Admission muss mindestens 8 Stellen und darf höchstens 32 Stellen lang sein. Enthält der Name Blanks, muss er in Hochkommas eingegeben werden. Die Transfer-Admission kann auch im Filetransfer-Profil angegeben werden. Wird ein Fragezeichen angegeben, so werden noch fehlende Zugangsdaten nach dem Absenden des Kommandos angefordert.

fpass|*?* File-Passwort im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort nach dem Absenden des Kommandos in einem dunkel gesteuerten Feld angefordert.

Übertragungsmodus

mode Transfer-Modus. Diese Angabe besteht aus den 2 Information:

a) Übertragungsmodus Text/Binär:

Text-Modus bedeutet, dass die Daten mit der Standard-Translatetabelle bzw. der Translatetabelle, die im FT-Profil angegeben ist, übersetzt werden (ANSI <--> EBCDIC/ASCII).

Im **Binär-Modus** werden die Daten transparent ohne Übersetzung übertragen. Es sind drei verschiedene Modi zu unterscheiden:

a) Binär satzstrukturiert für openFT-BS2000 (M=B)
b) Binär undefiniert für openFT-BS2000 (M=U)
c) Binär für FTP (M=B)

Nach dem Einlesen der Datei in den Arbeitsbereich wird die Codierung automatisch auf den entsprechenden Code umgeschaltet (wie Kommando `EBCDIC` (S. 235)).

Gilt nur für openFT: In der lokalen Datei (Name wird entweder vom EDT vergeben oder Name aus "L=file") werden die Daten mit 4 Byte Satzlängenfeld ohne Satzende-Kennzeichen gespeichert. Dadurch ist es möglich, dass in einem Satz auch das BS2000-Satzende-Kennzeichen (X'15') oder das Unix-Satzende-Kennzeichen (X'0A') vorkommen kann.

Im Filetransfer-Profil kann der Übertragungsmodus ebenfalls eingestellt werden. Wird beim Kommando `READ` nichts angegeben, so gilt die Einstellung des FT-Profils. Wird beim Kommando `WRITE` nichts angegeben, so gilt entweder der Übertragungsmodus des vorhergehenden `READ` oder die Einstellung des FT-Profils, falls eine lokale oder neue Datei ohne vorhergehenden `READ` auf den fernen Rechner übertragen wird.

b) Feste/Variable Satzlänge. Soll eine Datei mit fester Satzlänge erzeugt werden, müssen alle Zeilen im Arbeitsbereich exakt die richtige Länge haben.

Folgende Werte sind zulässig bei **openFT-BS2-Profilen:**

T Text-Modus (Standardeinstellung). Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Beim Schreiben ohne weitere Formatangabe wird die ferne Datei allerdings mit dem Format "Variable Satzlänge" erzeugt.

TFnnn
Text-Modus feste Satzlänge. Die Datei wird mit RECFORM=F und RECSIZE=nnn gelesen bzw. erstellt. Wurde diese Option beim Lesen angegeben, so wird beim Schreiben ohne Parameter automatisch das gleiche Dateiformat erzeugt.

TV Text-Modus, variable Satzlänge (gilt nur für openFT-BS2000). Diese Angabe ist nur beim Schreiben notwendig, falls nach dem Lesen einer Datei mit fester Satzlänge eine Datei mit variabler Satzlänge erzeugt werden soll.

B Binär-Modus. Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Die Daten werden im Arbeitsbereich im EBCDIC-Code satzstrukturiert dargestellt. Ist die Datei im Binär-Modus gelesen worden und wird sie mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, wird sie automatisch wieder im Binär-Modus transferiert, allerdings immer mit variabler Satzlänge.

BFnnn
Binär-Modus feste Satzlänge. Die Datei wird wie bei M=B binär mit RECFORM=F und RECSIZE=nnn gelesen bzw. erstellt. Die Daten werden im Arbeitsbereich im EBCDIC-Code satzstrukturiert dargestellt. Wurde diese Option beim Lesen angegeben, so wird beim Schreiben ohne Parameter automatisch das gleiche Dateiformat erzeugt.

BV Binär-Modus, variable Satzlänge. Diese Angabe ist nur beim Schreiben notwendig, falls nach dem Lesen einer Datei mit fester Satzlänge eine Datei mit variabler Satzlänge erzeugt werden soll.

U Binär-Modus undefined (ohne Satzstruktur). Mit dieser Option können Dateien ohne Satzstruktur gelesen und geschrieben werden. Dieser Modus ist z.B. notwendig, wenn eine POSIX-Datei aus einem ASCII-Filesystem gelesen werden soll. Nach dem Einlesen kann der Arbeitsbereich mit dem Kommando `REFORMAT UNIX` und `CODE ASCII` satzstrukturiert dargestellt werden. Vor dem Schreiben muss der Arbeitsbereich mit dem Kommando `UNFORMAT` wieder in das Binärformat umgewandelt werden.

Folgende Werte sind zulässig bei **FTP-Profilen:**

T Text-Modus (Standardeinstellung). Mit dieser Option können Dateien mit fester oder variabler Satzlänge gelesen und geschrieben werden. Dateien, die im BS2000 mit fester Satzlänge gespeichert sind, werden beim Schreiben in die gleiche Datei automatisch im richtigen Format erzeugt. Neue Dateien können nur mit variabler Satzlänge erzeugt werden.

B Binär-Modus. Die Daten werden transparent ohne Übersetzung übertragen und im Arbeitsbereich ohne Satzstruktur dargestellt. Soweit es sich um BS2000-Dateien mit variabler Satzlänge (RECFORM=V) oder Unix/Windows-Textdateien handelt, enthalten die Daten folgende Satzende-Kennzeichen: Windows=X'0D0A', Unix = X'0A' und BS2000 = X'15'. Soweit es sich um BS2000-Dateien mit fester Satzlänge (RECFORM=F) handelt, enthalten die Daten kein Satzende-Kennzeichen. Die Codierung kann im Profil schon auf den Wert ANSI, ASCIIUNIX oder EBCDIC7 eingestellt werden.

Nach dem Einlesen kann die Satzstruktur des fernen Rechners mit dem Kommando `REFORMAT` (S. 312) wieder hergestellt werden.

Beispiel:

```
REFORMAT BS2      (BS2000, variable Satzlänge)
REFORMAT RS80     (BS2000, feste Satzlänge 80)
REFORMAT UNIX     (Unix-Datei)
REFORMAT DOS      (Windows-Datei)
```

Vor dem binären Schreiben eines solchen Arbeitsbereichs muss mit dem Kommando `UNFORMAT` wieder das Binärformat erzeugt werden.

Wird der Arbeitsbereich mit dem Kommando `WRITE` ohne weitere Parameter zurückgeschrieben, werden die Daten automatisch wieder im Binär-Modus transferiert und die Datei auf dem fernen Rechner mit den bestehenden Dateiattributen erstellt.

Datei-Passwort

fpass|*?* Passwort für den Zugriffsschutz der Datei auf dem entfernten Rechner im Format `pass` oder `X''pass''` (Hochkommas müssen verdoppelt werden, weil das Passwort bereits Teil eines Strings in Hochkommas ist). Das Passwort kann auch im Filetransfer-Profil angegeben werden. Wird für *fpass* ein Fragezeichen angegeben, so wird das Passwort in einer Dialogbox unsichtbar angefordert.

Datei-Attribute für MVS

mvs-fileattr RECFM=recfm LRECL=lrecl [BLKSIZE=blksize]

recfm Record format: F | FB | FBA | V | VB | VBA | U

lrecl Record length: Satzlänge, max. 32767

blksize — Block size: Blocklänge, max. 32767. Fehlt die Angabe der Blocklänge, wird vom MVS-System ein optimaler Wert ermittelt. In der Regel kann deshalb dieser Parameter entfallen.

Parameter für die Folgeverarbeitung

par1 par2 — Beliebige Parameter, die als Parameter beim Aufruf an die Batch-Prozedur bzw. die Prozedur für die Folgeverarbeitung übergeben werden.

Dialogbox für die Passwort-Eingabe

Das Datei- bzw. Logon-Passwort ist in der Form cccc oder x'xxxxxxxx' anzugeben. Die Hochkommas müssen in der Dialogbox nicht verdoppelt werden.

CCS=*ccsname* — Coded Character Set für BS2000-Dateien. Die Festlegung kann auch im FT-Profil (S. 144) erfolgen. Ist sowohl im FT-Profil als auch beim Kommando WRITE ein CCS angegeben, gilt das CCS des WRITE-Kommandos.

OVERWRITE — Diese Option bewirkt, dass eine bereits bestehende Datei ohne Rückfrage überschrieben wird (Standard).

UPDATE — Diese Option bewirkt, dass der Inhalt des Arbeitsbereichs an eine bereits bestehende entfernte Datei angehängt wird.

NEW — Neue Datei schreiben. Falls die Datei bereits vorhanden ist, wird der Filetransfer mit Fehlermeldung abgebrochen.

Wird keine der Optionen "`OVERWRITE`", "`UPDATE`" oder " NEW " angegeben, wird die Datei immer ohne Rückfrage überschrieben (wie Option `OVERWRITE`).

Beispiele:

`WRITE` mit **internem Filetransfer**:

```
write'std r=src1.ass'
```
Die Datei `src1.ass` wird auf den fernen Rechner geschrieben. Es werden die FT-Angaben aus dem FT-Profil `std` benutzt.

```
write'std r=src1.ass' lu=test2
```
Die Datei `src1.ass` wird auf den fernen Rechner geschrieben. Als User-ID wird TEST2 benutzt. Alle anderen FT-Angaben aus dem FT-Profil `std` benutzt.

```
write'std r=test.utf8' ccs=utf8
```
Die Datei `test.utf8` wird auf den fernen Rechner mit dem CCS UTF8 geschrieben. Es werden die FT-Angaben aus dem FT-Profil `std` benutzt.

```
write new m=bf80
```
Die Datei wird unter dem gleichen Dateinamen und mit dem gleichen FT-Profil wie beim vorausgegangene READ-Kommando geschrieben. Als Dateiformat wird RECFORM=F und RECSIZE=80 verwendet. Die Datei wird nur geschrieben, wenn sie noch nicht auf dem fernen Rechner vorhanden ist.

```
w'std r=test1'u
```
Die Daten des Arbeitsbereichs werden an die Datei `test1` angehängt. Es werden die FT-Angaben aus dem FT-Profil `std` benutzt.

Beispiel **mit Batchdatei** (aktueller Arbeitsbereich = 0):

```
write 'host1 r=source.test1 user1 acc1'
```

Starten der Batch-Datei host1.bat:

```
set locfile=C:\WINDOWS\TEMP\EDT@0.TMP
set remfile=source.test1
set call=W
host1.bat user1 acc1
```

Inhalt der Datei host1.bat:

```
if %call%==W goto write
if %call%==R goto read
if %call%==F goto fstat
.....
:read
ncopy -t -o host1 !%remfile% %locfile% %1 %2
exit
:write
ncopy -t -o %locfile% host1 !%remfile% %1 %2
if not errorlevel 0 goto ende
del %locfile%
:ende
```

Folgendes Kommando wird in der Batch-Datei zum Übertragen der Daten ausgeführt:

```
ncopy -t -o C:\WINDOWS\TEMP\EDT@0.TMP host1
!source.test1 user1 acc1
```

Hinweis:

Mit Hilfe der Batch-Datei kann ein beliebiger Arbeitsgang für eine Nachverarbeitung der geschriebenen MS-DOS Datei ausgeführt werden.
Wie oben dargestellt, kann z.B. die Datei mit einem Filetransfer-Programm zu einem BS2000-Host übertragen werden.
Es ist aber auch möglich, die im EDT bearbeitete Datei in ein ARJ-/ZIP-Archiv zu übertragen.

Beispiele für Batch-Prozeduren zum Lesen und Schreiben von BS2000-Dateien sind in den ausgelieferten Dateien MPS.BAT (Filetransfer der Firma MPS) und LOG.BAT (Filetransfer der Firma Logics) enthalten.

Format 3: Schreiben entfernt nach FILESEND

WRITE In folgenden Fällen kann eine ferne BS2000-Datei nach dem Einlesen über das Programm FServer bzw. den Browser zurück übertragen werden:

Aufruf über Programm FServer:

Mit dem BS2000-Kommando EDTW, als Teil des Softwareprodukts OPGCOM (Download: ftp://ftp1.opg.de/opgcom/Manopgcom.EXE), können Dateien vom BS2000 zu einem Windows-PC übertragen und in den EDT für Windows eingelesen werden. Das Windows-Programm FServer muss geladen sein.

Die Datenübertragung kann auch über den Actioncode EDTW des Programms CFS gestartet werden.

Aufruf über den Browser für vom URLServer gesendete Dateien:

Mit dem BS2000-Programm URLServer, als Teil des Softwareprodukts OPGCOM, können für den Browser BS2000-Dateien zur Verfügung gestellt werden, für die ein Update mit dem EDTW zulässig ist. Nach dem Download durch den Browser werden die Daten aufgrund der Verknüpfung der Dateitypen in den EDT für Windows eingelesen.

Aus Sicherheitsgründen muss im BS2000 der schreibende Zugriff über die Anweisung `FILEACCESS` mit der Option `"UPDATE=YES"` für das Programm URLServer erlaubt werden.

Hinweise:

Download OPGCOM:

Testversion OPGCOM: http://www.opg.de/download/opgcom/opgcom.zip
Manual OPGCOM http://www.opg.de/produkte/frames/opgcom.pdf
TestversionFServer http://www.opg.de/download/opgcom/setupfserver.exe

Datenformat:

Den Daten wird ein Header vorangestellt, in dem alle notwendigen Informationen für die Verwaltung enthalten sind. Diese Informationen sind nicht sichtbar, können jedoch mit dem Kommando `STA FSEND` angezeigt werden. Die Daten werden ohne Code-Konvertierung in EBCDIC übertragen und auch als EBCDIC-Daten angezeigt. Der Anzeige-Code wird automatisch eingestellt. Damit alle Zeichen dargestellt werden können, z.B. auch `X'0D'` oder `X'0A'`, werden die Daten in einem besonderen Format übertragen: Sätze variabler Länge mit Satzlängenfeld.

Dateiformat:

Im Gegensatz zum openFT und FTP können auch **ISAM-Dateien**, **PAM-Dateien, deltagespeicherte LMS-Elemente, POSIX-Dateien und BS2000-ZIP-Elemente** editiert werden.

ISAM-Dateien mit den Standard-Isamkey (KEYPOS=5, KEYLENGTH=8) werden im EDTW mit der Option "KEY" eingelesen, so dass die ISAM-Schlüssel als Zeilennummer angezeigt werden. Aber selbst ISAM-Dateien mit Nicht-Standard-Schlüsseln können im EDT bearbeitet werden. Beim Zurückschreiben wird die Datei im BS2000 auch richtig geschrieben, wenn die ISAM-Schlüssel nicht aufsteigend sind.

Die Daten werden immer mit der Option "OVERWRITE" geschrieben.

Prüfung auf Änderung außerhalb des EDTW:

Wie bei lokalen Dateien wird zyklisch und beim Wechsel in einen anderen Arbeitsbereich bzw. vor dem Schreiben geprüft, ob die BS2000-Datei inzwischen außerhalb des EDTW geändert worden ist und ggf. eine Warnung ausgegeben.

10 EDT-Kommandos für Prozeduren

Mehrfach benötigte Anweisungsfolgen für die Bearbeitung von Dateien können in EDT-Prozeduren zusammengefaßt werden. Die Prozeduren können entweder im Dialog mit dem Kommando `INPUT` (S. **206**) oder über den Schalter `-i` beim Laden des `EDT` (S. **35**) gestartet werden.

Neben den EDT-Kommandos, die nur in Prozeduren angewendet werden, sind alle EDT-Kommandos erlaubt, die auch im Dialog direkt eingegeben werden können.

Für die gesamte EDT-Prozedur können Parameter definiert werden, die beim Einlesen der Prozedur sofort mit den aktuellen Werten ersetzt werden.

Zusätzlich können für die Prozeduren innerhalb der EDT-Prozedur, die mit dem Kommando `DO` gestartet werden, Parameter definiert werden. Diese Parameter werden erst ersetzt, wenn die "innere EDT-Prozedur" abläuft.

Jedes Kommando kann, getrennt durch ein Trennzeichen, mit einem Kommentar versehen werden. Das Trennzeichen kann im Menü `Optionen / Sonderzeichen` eingestellt werden. Das Trennzeichen muss zweimal angegeben werden (z.B. `@rea'&file'  ;;Lesen Eingabedatei`).

Die Prozedurdatei kann sowohl Daten als auch Kommandos enthalten. Zur Unterscheidung der Kommandos von den Daten muss deshalb jedem Kommando in der Prozedurdatei das Anweisungssymbol "@" vorangestellt werden. Als Alternative kann auch das Zeichen "§" benutzt werden. Beide Zeichen können auch umdefiniert werden (siehe Menüzeile `Optionen / Sonderzeichen` (S. **119**)

Für DO-Prozeduren innerhalb einer Prozedurdatei gilt folgende Besonderheit. Weil in DO-Prozeduren ebenfalls Daten und Kommandos vorkommen können, müssen auch die Kommandos in DO-Prozeduren mit dem Anweisungssymbol "@" bzw. "§" beginnen. Um dies zu erreichen, müssen Kommandos, die in DO-Prozeduren gespeichert werden sollen, mit zwei Anweisungssymbolen versehen werden. Beim Einlesen wird ein Anweisungssymbol gelöscht und das Kommando mit einem Anweisungssymbol im Arbeitsbereich gespeichert. Wird in einer DO-Prozedur wieder eine DO-Prozedur erzeugt, so muss ein zusätzliches Anweisungssymbol vorangestellt werden.

Bestimmen neue Zeilennummer und Schrittweite

@*ln* [(*inc*)] [**:** *text*]

Das Kommando bestimmt eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite.

ln — Neue aktuelle Zeilennummer, z.B. 5. Der Minimalwert beträgt 0.0001, der Maximalwert 9999.9999. Fehlt *inc*, wird mit *ln* implizit auch die neue aktuelle Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.

inc — Neue aktuelle Schrittweite. 0.0001 <= *inc* <= 9999.9999.

@+ [**:** *text*] — 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.

@- [**:** *text*] — Mit der Anweisung @- wird die aktuelle Zeilennummer um die aktuelle Schrittweite vermindert oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die vorangehende Zeile umgeschaltet.

text

Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   – text steht in der aktuellen Zeile;
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   – vorhandene Tabulatorzeichen werden berücksichtigt.
2. ein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   – Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Sprungmarke definieren / Kommentare

:*label*

Nach dem Doppelpunkt kann ein beliebig langer Name als Sprungziel definiert werden. Das erste Byte des Namens muss ein Buchstabe sein. Auf diese Sprungmarke kann in GOTO- und IF–Kommandos verzweigt werden, indem der Name (ohne Doppelpunkt) als Sprungziel angegeben wird.

Im Gegensatz zum BS2000-EDT ist es nicht mehr notwendig, in GOTO- und IF–Kommandos auf Zeilennummern zu verzweigen. Diese Zeilennummern müssen sehr umständlich mit den Anweisungen @SET bzw. @*ln*(*inc*) und CONTINUE erzeugt werden.

Alternativ dazu kann auch das im BS2000-EDT verwendete Format benutzt werden.

CONTINUE [*kommentar*]

Dieses Kommando verursacht bei seiner Ausführung keine Aktion. Es kann benutzt werden, um eine Zeile als Sprungmarke zu definieren. Hauptanwendung ist die Definition einer letzten Zeile innerhalb einer Prozedur. Auf diese Zeilennummer kann in GOTO- und IF–Kommandos verzweigt werden, indem die Zeilennummer als Sprungziel angegeben wird. Sinnvollerweise sollte vor dem CONTINUE–Kommando das Kommando @SET bzw. @*ln*(*inc*) verwendet werden.

Kommentare

Kommentare können auch zu jedem Kommando, getrennt durch ein Trennzeichen, angegeben werden. Das Trennzeichen kann im Menü Optionen / Sonderzeichen (S. 119)) eingestellt werden (Standard = ";"). Das Trennzeichen muss zweimal angegeben werden.

Es ist zu beachten, dass Kommentare nur in Prozeduren enthalten sein dürfen, die mit dem Kommando INPUT bzw. über den Startparameter –i*procfile* eingelesen werden. Sie dürfen nicht direkt in einen Prozedur-Arbeitsbereich geschrieben werden, der dann mit dem Kommando DO aufgerufen wird. Die Kommentare werden beim Einlesen der Prozedurdatei entfernt.

Beispiel:
```
@rea'&file'  ;;Lesen Eingabedatei
```

Zeichenfolge vom Bildschirm einlesen

CREATE *ln*|*str-var* **R**EAD *str* [, *str*.....]

Zeichenfolge von der Tastatur einlesen.

ln Zeilennummer der Zeile, in die die eingegebene Zeichenfolge gespeichert wird.

str-var String-Variable für die Speicherung der Zeichenfolge.

str.... Zeichenfolge(n), die am Bildschirm als Eingabeaufforderung ausgegeben werden soll.

Für die Dateneingabe wird eine Dialogbox ausgegeben, in der eine Zeichenfolge bis zur maximalen Satzlänge eingegeben werden kann.

Umschalten in den Dialog-Modus

DIALOG [*text*] Dieses Kommando kann benutzt werden, um in einer Prozedurdatei auf den Dialog-Modus umzuschalten. Nach Ausführung dieses Kommandos werden die weiteren Kommandos wieder im Dialog angefordert. Das Kommando darf nur in einer Input-Datei, die beim Aufruf mit dem Parameter **-i** angegeben wurde, vorkommen. In allen anderen Fällen wird das Kommando ignoriert.

Nach der Unterbrechung der Prozedur wird wieder der Arbeitsbereich angezeigt, von dem aus die Prozedur gestartet wurde bzw. der Arbeitsbereich, der evtl. mit dem Kommando SETF (S. **246**) in der Prozedur aktiviert wurde.

text Der Text wird in einer Messagebox ausgegeben.

Mit dem Kommando `RETURN` (S. **288**) `END` oder `HALT` kann die unterbrochene Prozedur fortgesetzt werden. Dabei wird der Arbeitsbereich wieder aktiviert, der bei Ausführung des Kommandos DIALOG aktiv war.

Dialogbox "OPEN" oder "SAVE AS" erzeugen

DIALOGBOX OPEN, *title*,*path*,*file*,*strvar*

DIALOGBOX SAVE, *title*,*path*,*file*,*strvar*

Mit dem Kommando kann die Dialogbox OPEN oder SAVE AS ausgegeben werden, um vom Benutzer einen Dateinamen auswählen zu lassen. Als Wert für die Parameter *title*, *path* und *file* ist jeweils ein String anzugeben, d.h. es können alle Varianten angegeben werden, die als String (S. **341**) zulässig sind, wie z.B. Stringvariable, Zeilennummer, Line-Variable, zusammengesetzte String usw. Der ausgewählte Dateiname wird in der angegebenen Stringvariablen zurückgegeben. Die Datei kann dann mit dem Kommando READ *strvar* gelesen bzw. mit dem Kommando WRITE *strvar* geschrieben werden.

OPEN Mehrfachauswahl, Variante 1 (Stringvariable)

DIALOGBOX OPENMULTI, *title*,*path*,*file*,*strvar-path, strvar-files, ivar*

Mit dem Kommando kann die Dialogbox OPEN ausgegeben, wobei auch mehrere Dateien ausgewählt werden können. Die ausgewählten Dateinamen werden in Stringvariable geschrieben. Der Pfadname wird in die 1. Stringvariable (*strvar-path*) geschrieben. Die Dateinamen werden, jeweils getrennt durch das Trennzeichen X'00', in die 2. Stringvariable (*strvar-files*) geschrieben.

OPEN Mehrfachauswahl, Variante 2 (Datei)

DIALOGBOX OPENMULTI, *title*,*path*,*file*,*strvar-tmpfile, ivar*

Mit dem Kommando kann die Dialogbox OPEN ausgegeben, wobei auch mehrere Dateien ausgewählt werden können. Die ausgewählten Dateinamen werden in die Datei geschrieben, die in der Stringvariablen *strvar-tmpfile* angegeben ist. Für jeden vollständigen Dateinamen einschl. Verzeichnisnamen wird ein Satz in die Datei geschrieben.

title Titel für die Dialogbox als String (S. **341**).

path Verzeichnis für die Initialisierung der Dialogbox als String (S. **341**), z.B. `'c:\test'`.

file Auswahlbedingungen für das Feld "Dateiname" in der Dialogbox als String (S. **341**), z.B. `'*.TXT'`. Damit kann die Anzahl der angezeigten Dateien eingeschränkt werden.

Dateiname für die Ergebnisdatei OPENMULTI Variante 2

strvar-tmpfile Stringvariable (S. **337**), die einen Dateinamen enthält. In diese Datei werden die ausgewählten Dateinamen einschl. Verzeichnisnamen (pro Name ein Satz) geschrieben.

Ergebnis-Variable für OPEN und SAVE

strvar Stringvariable (S. **337**), in die der ausgewählte Dateinamen übertragen wird. Falls kein Dateiname ausgewählt wurde, enthält die Stringvariable einen Leerstring.

Ausgaben OPENMULTI, Variante 1 (Stringvariable)

strvar-path Stringvariable (S. **337**), in die der Verzeichnisname der ausgewählten Dateien übertragen wird.

strvar-files Stringvariable (S. **337**), in die alle ausgewählten Dateinamen ohne Verzeichnisname übertragen werden. Die Dateinamen sind durch das Trennzeichen X'00' getrennt. Nach dem letzten Dateinamen stehen 2 x'00'. Zum Trennen der Dateinamen kann das Kommando `CUT` (S. **186**) verwendet werden. Siehe hierzu auch das Beispiel.

ivar Diese Integervariable (S. **337**) enthält nach Ausführung des Kommandos die Anzahl der ausgewählten Dateien, bei Fehler oder Abbruch 0.

Ausgaben OPENMULTI, Variante 1 (Datei)

Die ausgewählten Dateinamen einschl., Pfadnamen (pro Name ein Satz) werden in die durch die Stringvariable *strvar-tmpfile* definierte Datei geschrieben.

ivar Diese Integervariable (S. 337) enthält nach Ausführung des Kommandos die Anzahl der ausgewählten Dateien, bei Fehler oder Abbruch 0.

Beispiele:

```
@dialogbox open,'Protokolldatei','c:\prot','*.log',#s1
@rea#s1
@dialogbox open,#s1,#s2,#s3,#s4
@rea#s4,b
@set #s2='projekt1'
@dialogbox open,'Protokolldatei','c:\'+#s2','*.log',#s1
@rea#s1,0.0001-0.0200:1-50:
@dialogbox save,'Protokolldatei','c:\','*.log',#s1
@w#s1
@dialogbox openmulti,'test','d:\temp','*',#s1,#s2,#i1
@if #i1 = 0 goto error
@set #i2 = 1
@:read
  @if #i2 > #i1 goto ende
  @cut #s2,1,#s3,#i2,x'00',#i5,#i6
  @proc #i2
  @rea #s1+'\'+#s3
  @end
  @set #i2 = #i2 +1
  @goto read
  @:ende
@set #s1='files.txt'
@dialogbox openmulti,'test','c:\test','*',#s1,#i1
@if #i1 = 0 goto error
@read'(files.txt)'
```

Starten von EDT-Prozeduren

DO *n* | *intvar* [(*param*[,....])] [*s*=*ln1* , *ln2* , [-] *ln3*] [**P**RINT] [F]

EDT-Prozedur, die in einem Arbeitsbereich gespeichert ist, starten. Im Gegensatz dazu wird eine EDT-Prozedur, die in einer Datei gespeichert ist, mit dem Kommando INPUT gestartet.

Nach dem Ende einer Prozedur wird wieder der Arbeitsbereich angezeigt, von dem aus die Prozedur gestartet wurde bzw. der Arbeitsbereich, der evtl. mit dem Kommando SETF (S. 246) in der Prozedur aktiviert wurde.

n Nummer des Arbeitsbereichs (1 bis 32).

intvar Integer-Variable mit der Nummer eines Arbeitsbereichs (1 bis 32).

param Parameter, die an die auszuführende Prozedur übergeben werden. Alle Parameter müssen in der Prozedur mit dem Kommando PARAMS definiert sein. Die Stellungsparameter müssen vor den Schlüsselwort-Parametern stehen. Es können max. 32 Parameter angegeben werden. Die maximale Länge aller Parameter darf 1.280 Zeichen nicht überschreiten.

s Schleifensymbol. Bei Angabe des Schleifensymbols wird die Prozedur mit jeder Zeile von *ln1* bis *ln2* durchlaufen. In der Prozedur enthält das Schleifensymbol die aktuelle Zeilennummer. Das Schleifensymbol muss ein Sonderzeichen sein. Um Konflikte mit bereits belegten Sonderzeichen zu vermeiden, sollen folgende Zeichen nicht verwendet werden:

% $? *) : # + - . < = > ' ~ _ |

Die Zeichen ";" und "(" dürfen nicht verwendet werden.

Geeignete Schleifensymbole sind:

! " { } [ ] / ^ \

Achtung: Bei folgenden Zeichen gibt es Einschränkungen:

" Das Zeichen kann in ON-Kommandos (S. **341**) verwendet werden.
! Das Zeichen kann als Einleitungszeichen für EDT-System-Variable (S. **344**) verwendet werden.
/ Das Zeichen kann als Musterzeichen (S. **210**) in Suchbegriffen verwendet werden.

ln1 Erste zu verarbeitende Zeile (Anfangswert des Schleifensymbols). Es sind alle Eintragungen des allgemeinen Parameters *ln* (S. **338**) möglich.

ln2 Letzte zu verarbeitende Zeile (Endwert des Schleifensymbols). Es sind alle Eintragungen des allgemeinen Parameters *ln* (S. **338**) möglich.

\- Das Schleifensymbol soll nach jedem Durchlauf um den Wert in *ln3* vermindert werden. Ohne diese Angabe wird das Schleifensymbol bei jedem Durchlauf um den Wert in *ln3* erhöht.

ln3 *ln* / *n*L

ln Zeilenanzahl, um den das Schleifensymbol nach jedem Durchlauf erhöht bzw. vermindert (Minuszeichen vor *ln3*) werden soll. Es sind alle Eintragungen des allgemeinen Parameters *ln* (S. **338**) möglich.

*n*L Das Schleifensymbol erhält die Zeilennummer der Zeile, die sich um *n* Zeilen vor oder nach der aktuellen Zeile befindet. Dadurch kann auch eine äußere Schleife ohne vorherige Neunumerierung programmiert werden. Die erste und letzte zu verarbeitende Zeile muss in diesem Fall existieren. Es ist darauf zu achten, dass für die Ermittlung der nächsten Zeile immer der aktuelle Stand nach der Verarbeitung eines Schleifendurchlaufs berücksichtigt wird. Wenn z.B. bei der Verarbeitung die nächste Zeile nach der aktuellen Zeile gelöscht wird, so erhält das Schleifensymbol beim nächsten Durchlauf die übernächste Zeile.

PRINT Anzeigen jeder Zeile der Prozedur im Protokollbereich (Arbeitsbereich 32) vor ihrer Verarbeitung.

F Die Daten des Protokollbereichs werden zusätzlich in eine Datei geschrieben. Der Name der Datei kann im Menü `Optionen / Protokoll` (S. **123**) eingestellt werden.

DO P Der Protokollmodus wird eingeschaltet. Dieser bewirkt die Anzeige jedes Prozedurkommandos im Protokollbereich (Arbeitsbereich 32) vor ihrer Verarbeitung. Der Modus gilt nur im aktuellen Prozedur-Arbeitsbereich.

DO PF Die Daten des Protokollbereichs werden zusätzlich in eine Datei geschrieben. Der Name der Datei kann im Menü `Optionen / Protokoll` (S. **123**) eingestellt werden.

DO N Der Protokollmodus wird ausgeschaltet.

Beispiele:

```
do 1!=%,$,1
do 1!=100,2000,1L
do 1(par1,par2)!=%,$,-1L
```

Bearbeitung des aktuellen Arbeitsbereich beenden

END Die Bearbeitung des aktuellen Arbeitsbereichs wird beendet. Es wird wieder in den Arbeitsbereich gewechselt, von dem aus die Bearbeitung mit dem Kommando PROC eingeleitet wurde.

Unbedingter Sprung

GOTO *ln*| *label* | *strvar* Unbedingter Sprung auf eine Zeile bzw. eine Sprungmarke.

ln Zeilennummer (Beschreibung siehe Parameter *ln*).

label Sprungmarke, die am Sprungziel in einer eigenen Zeile definiert ist (erstes Zeichen = ":", gefolgt von einem beliebig langen Namen, der Namen muss mit einem Buchstaben beginnen). Im GOTO-Kommando wird die Sprungmarke ohne das Zeichen ":" geschrieben (siehe auch Beschreibung der Sprungmarke).

strvar String-Variable, die den Namen einer Sprungmarke enthält.

Bedingter Sprung bei Fehlern

IF ERRORS | **N**O ERRORS | **D**MS ERRORS | **N**O **D**MS ERRORS GOTO { *label* | *ln* | *strvar* } | RETURN

IF ERRORS | **N**O ERRORS | **D**MS ERRORS | **N**O **D**MS ERRORS : *text*

Prüfen, ob bei der Verarbeitung der vorhergehenden Kommandos Fehler aufgetreten sind. Ist die Bedingung erfüllt, so wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando RETURN abgebrochen. Die Fehlerschalter können mit dem Kommando RESET wieder zurückgesetzt werden.

ERRORS Es ist einer der folgenden EDT-Fehler aufgetreten:
- Fehler bei der Anforderung von Speicher;
- Fehler bei den Kommandos SET, Format 5a und 5b (Funktion Time oder Date);
- Fehlermeldung "Invalid Operand";
- Fehlermeldung "Unbekanntes Kommando";
- Fehlermeldung "Length of Variable wrong" (z.B. wenn die Sende-Variable nicht existiert);
- Fehlermeldung "Line-Number not found" (z.B. wenn die referenzierte Zeile nicht existiert).
- Rückkehrcode "ungleich" nach dem Kommando COMP (S. 184), falls die Arbeitsbereiche ungleich sind oder der Vergleich abgebrochen wurde.

NO ERRORS Es ist ein kein EDT-Fehler aufgetreten.

DMS ERRORS Es ist einer der folgenden Betriebssystem-Fehler aufgetreten:
- OPEN,- Lese- oder CLOSE-Fehler bei dem Kommando READ oder INPUT;
- OPEN,- Schreib- oder CLOSE-Fehler bei dem Sichern der Datei bzw. Erstellen der Backup-Datei (Kommando WRITE).

NO DMS ERRORS Es ist ein kein Betriebssystem-Fehler aufgetreten.

GOTO Ist die Bedingung erfüllt, so wird das Kommando GOTO ausgeführt.

RETURN Ist die Bedingung erfüllt, so wird das Kommando RETURN ausgeführt.

text Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   – text steht in der aktuellen Zeile;
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   – vorhandene Tabulatorzeichen werden berücksichtigt.
2. ein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   – Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Vergleich von zwei Operanden

IF [S|V] *str1 rel str2* { GOTO { *label* | *ln* | *strvar* } | RETURN | [: | _] *text* }

IF [L] *ln1 rel ln2* { GOTO { *label* | *ln* | *strvar* } | RETURN | : *text* }

IF [I] *int1.. rel int2* { GOTO { *label* | *ln* | *strvar* } | RETURN | : *text* }

IF [I] *float1 rel float2* { GOTO { *label* | *ln* | *strvar* } | RETURN | : *text* }

IF *ln-var* = | NE SELECTED { GOTO { *label* | *ln* | *strvar* } | RETURN } | :*text* }

IF *ln-var* = | NE EXIST { GOTO { *label* | *ln* | *strvar* } | RETURN } | :*text* }

IF WINGEN *windows* = | NE EXIST { GOTO { *label* | *ln* | *strvar* } | RETURN } | :*text* }

In Abhängigkeit des Vergleichsergebnisses zweier Operanden wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando `RETURN` abgebrochen.

S Falls als Zeichenfolge eine Zeilennummer angegeben wird, kann nicht eindeutig unterschieden werden, ob die Zeilennummer oder der Inhalt der Zeile verglichen werden soll. In diesen Fällen ist bewirkt die Angabe von "S", dass der Inhalt verglichen wird. Auch bei einer eindeutigen Syntax beschleunigt die Angabe von "S" die Verarbeitung.

V wie "S", jedoch wird der Vergleich ohne Beachtung der Klein-/Großschreibung durchgeführt.

str1/*str2* Zeichenfolge, Beschreibung siehe Parameter *str* (S. **341**).

L Diese Angabe bedeutet, dass die Zeilennummer und nicht der Inhalt der Zeile verglichen wird.

ln1/*ln2* Zeilennummern, Beschreibung siehe Parameter *ln* (S. **338**). Es werden nicht die Zeileninhalte, sondern nur die Zeilennummern verglichen. Soll der Zeileninhalt verglichen werden, ist als zweiter Operand eine Zeichenfolge (siehe Parameter *str* (S. **341**)) oder der Parameter "S" anzugeben.

I Muss nur angegeben werden, wenn mit *int1* ein Integer-Wert (1,2, ... 9999) angegeben wird. Dadurch ist es möglich, einen Integer-Wert von einer Zeilennummer zu unterscheiden.

int1 / *int2* Miteinander zu vergleichende Integer-Werte. Es kann jeweils eine ganze positive oder negative Zahl oder eine Integer-Variable (#I0 bis #I99) angegeben werden.

float1 / float2

Miteinander zu vergleichende Gleitpunkt-Werte. Es kann jeweils eine Gleitpunktzahl oder eine Float-Variable (#F0 bis #F99) angegeben werden.

ln-var

Line-Variable, die auf eine Zeile im aktuellen Arbeitsbereich verweist.

Prüfen, ob die Zeile markiert ist

SELECTED

Die Bedingung ist erfüllt, wenn die Zeile markiert (=SELECTED) bzw. nicht markiert (NE SELECTED) ist.

Prüfen, ob die Zeile existiert

EXIST

Die Bedingung ist erfüllt, wenn die Zeile existiert (=EXIST) bzw. nicht existiert (NE EXIST).

Prüfen, ob ein mit WINGEN (S. **307**) **erzeugtes Fenster noch existiert.**

WINGEN *windows*

Die Bedingung ist erfüllt, wenn das asynchron erzeugte Fenster noch existiert (=EXIST) bzw. nicht existiert (NE EXIST). Für *windows* ist die mit dem Kommando WINGEN OUT zugewiesene Nummer anzugeben.

Beispiel: `@if wingen 2 = existing goto win2`

rel

Vergleichsrelation:

EQ	=	gleich
NE	<> | !=	ungleich
GT	>	größer
LT	<	kleiner
GE	>=	größer oder gleich
LE	<=	kleiner oder gleich

GOTO

Kommando `GOTO` ausführen, wenn die Bedingung erfüllt ist.

RETURN

Kommando `RETURN` ausführen, wenn die Bedingung erfüllt ist.

text

Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   – text steht in der aktuellen Zeile;
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   – vorhandene Tabulatorzeichen werden berücksichtigt.
2. ein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt.
   – Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

: | _

Bei Stringvergleichen ist das Trennzeichen ":" zwischen dem IF-Kommando und der nachfolgenden True-Anweisung nicht immer eindeutig von dem Zeichen ":" für die Spaltenangabe zu unterscheiden. Deshalb ist immer dann das Zeichen "_" als Trennzeichen anzugeben, wenn der Parameter *text* nicht mit dem Anweisungssymbol (@ oder §) beginnt.

Beispiele:

```
if s !:5-6: = 'xy' _ Daten für neue Zeile
if s !:5-6: = 'xy' : @col 5 on ! insert 'x'
if s !:5-6: = 'xy' : @@on& c'test' to 'test1'
if s !:5-6: = 'xy' goto weiter
if s !:5-6: = 'xy' return
if #i1 = 1 goto weiter
if #l1 = 1.05 return
if #l1 = selected goto weiter
if #l1 ne existing return
```

Prüfen auf Treffer nach ON

IF **.T**RUE[(*arb*)]. [*rel cl1*] | **.F**ALSE[(*arb*)]. { GOTO { *label* | *ln* | *strvar* } | RETURN } | :*text* }

Prüfen, ob bei der letzten Verarbeitung eines ON-Kommandos ein Treffer festgestellt wurde. Ist die Bedingung erfüllt, so wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando RETURN abgebrochen.

.TRUE[(*arb*)]**.** Die Bedingung ist erfüllt, wenn bei der letzten Ausführung eines ON-Kommandos ein Treffer festgestellt wurde. Je nach der Einstellung im Menü Optionen / Einstellungen / Verschiedenes / "Find-True/False-Schalter global" (S. 127) bezieht sich die Bedingung auf das Ergebnis des letzten ON-Kommandos des aktuellen Arbeitsbereiches oder global auf alle Arbeitsbereiche. Ist zusätzlich ein Arbeitsbereich angegeben, bezieht sich die Bedingung auf das letzte ON-Kommando, das im angegebenen Arbeitsbereich ausgeführt wurde.

rel cl1 Bei TRUE ist die Bedingung nur erfüllt, wenn auch die durch *cl* angegebene Spalte mit der Spalte des ersten festgestellten Treffers übereinstimmt.

.FALSE[(*arb*)]**.** Die Bedingung ist erfüllt, wenn bei der letzten Ausführung eines ON-Kommandos kein Treffer festgestellt wurde. Je nach der Einstellung im Menü Optionen / Einstellungen / Verschiedenes / "Find-True/False-Schalter global" (S. 127) bezieht sich die Bedingung auf das Ergebnis des letzten ON-Kommandos des aktuellen Arbeitsbereiches oder global auf alle Arbeitsbereiche. Ist zusätzlich ein Arbeitsbereich angegeben, bezieht sich die Bedingung auf das letzte ON-Kommando, das im angegebenen Arbeitsbereich ausgeführt wurde.

GOTO Kommando GOTO ausführen, wenn die Bedingung erfüllt ist.

RETURN Kommando RETURN ausführen, wenn die Bedingung erfüllt ist.

text Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   – text steht in der aktuellen Zeile;
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   – vorhandene Tabulatorzeichen werden berücksichtigt.

2. ein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt.

   – Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Prüfen auf leeren Arbeitsbereich

IF .EMPTY[(*arb*)]. { GOTO { *label* | *ln* | *strvar* } | RETURN | :*text* }

Prüfen, ob der aktuelle Arbeitsbereich leer ist. Ist die Bedingung erfüllt, so wird entweder auf die Sprungmarke verzweigt oder die Prozedur mit dem Kommando `RETURN` abgebrochen. Ist zusätzlich ein Arbeitsbereich angegeben, bezieht sich die Bedingung auf den entsprechenden Arbeitsbereich.

GOTO — Kommando `GOTO` ausführen, wenn die Bedingung erfüllt ist.

RETURN — Kommando `RETURN` ausführen, wenn die Bedingung erfüllt ist.

text — Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   – text steht in der aktuellen Zeile;
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   – vorhandene Tabulatorzeichen werden berücksichtigt.
2. ein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt.

   – Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Meldung ausgeben

MSGBOX *str*|*str-var* [, [*art*] [*icon*] [*def*]] [, *strvar*]

In einem kleinen Fenster wird eine Meldung ausgegeben. Als Text kann ein String oder eine Stringvariable angegeben werden. Über Optionen kann das Aussehen und der aktive Button beeinflußt werden. Das Ergebnis der Eingabe (Anfangsbuchstabe der Schaltfläche) wird in die Stringvariable *strvar* bzw. #S0 übertragen.

In dem String bzw. in der Stringvariablen können die Zeichen X'0D0A' für eine neue Zeile und das Zeichen X'09' für einen Tabulator enthalten sein.

art

Bestimmung der Schaltfläche(n), die in der Messagebox enthalten sein sollen:

A (Abort) 

C (Cancel) 

I (Ignore) 

N (No) 

O (OK) 

R (Retry) 

Y (Yes) 

Folgende Kombinationen sind zulässig:

`O` (OK, Standard, wenn dieser Parameter fehlt)
`OC` (OK, Cancel):
`YN` (Yes und No)
`YNC` (Yes, No und Cancel):
`ARI` (Abort, Retry und Ignore):
`RC` (Retry und Cancel)

icon

Icon, das in der Messagebox angezeigt werden soll:

X (Ausrufezeichen) 

F (Information) 

Q (Question, Fragezeichen) 

S (Stop) 

def

Default-Button: Bezeichnung der Schaltfläche, die akiviert sein soll, d.h. wenn der Benutzer die Taste `<Enter>` drückt, wird diese Taste als Ergebnis zurückgegeben.

1 Erste Schaltfläche, z. Yes bei "YNC"
2 Zweite Schaltfläche, z.B. No bei "YNC"
3 Dritte Schaltfläche, z.B. Cancel bei "YNC"

strvar

String-Variable, in der als Ergebnis der Anfangsbuchstabe der Taste (im englischen Format A, C, I, N, O, R oder Y) zurückgegeben. Standard = #S0.

Bemerkungszeile

NOTE *comment*

REMARK *comment*

oder

comment

Kommentar, beliebiger Text.

Kommentare können in INPUT-Prozeduren auch zu jedem Kommando, getrennt durch ein Trennzeichen, angegeben werden. Das Trennzeichen kann im Menü `Optionen / Sonderzeichen` (S. **119**) eingestellt werden (Standard = ";"). Das Trennzeichen muss zweimal angegeben werden.

Es ist zu beachten, dass Kommentare nach dem Trennzeichen ";;" nur in INPUT-Prozeduren angegeben werden dürfen. Die Kommentare werden nämlich nach dem Einlesen der INPUT-Prozedur gelöscht und sind beim Ablauf der Prozedur nicht mehr zulässig.

Beispiel: `@rea'&file'  ;;Lesen Eingabedatei`

Definieren von EDT-Parametern

PARAMS *par* [, *par*....]

Definition aller symbolischer Parameter, die innerhalb einer Prozedur benötigt werden. Mit dem `PARAMS`-Kommando können Parameter für zwei verschiedene Anwendungsfälle definiert werden:

a) Parameter für die ganze Prozedurdatei (INPUT-Prozedur):

In diesem Fall muss das `PARAMS`-Kommando als erster Satz in der Prozedurdatei stehen. Die symbolischen Parameter werden sofort beim Einlesen der Datei mit den aktuellen Werten ersetzt. Die aktuellen Werte werden mit dem Kommando `INPUT` bzw. nach dem Schalter **-i** beim Laden des EDT angegeben.

b) Parameter für DO-Prozedur in einem Arbeitsbereich:

Innerhalb einer Prozedurdatei können wieder Prozeduren definiert werden. Eine solche "innere" Prozedur wird in einem Arbeitsbereich gespeichert. Die Definition einer solchen Prozedur wird mit dem Kommando `@PROC` eingeleitet und mit dem Kommando `@END` beendet. Das `PARAMS`-Kommando muss in der ersten Zeile der Prozedur, also direkt hinter dem `PROC`-Kommando stehen. Die symbolischen Parameter werden erst beim Ablauf der Prozedur ersetzt. Die Prozedur wird mit dem Kommando `DO` gestartet, mit dem auch die Parameter übergeben werden.

par

Parameter. Ein Parameter beginnt mit dem Zeichen "&", gefolgt von einem Buchstaben und beliebig vielen Buchstaben oder Ziffern.

Stellungsparameter müssen vor Schlüsselwort-Parametern stehen.

Schlüsselwort-Parameter können mit einem Wert vorbelegt werden. In Schlüsselwort-Parametern steht hinter dem Parameternamen ein Gleichheitszeichen. Wird kein Anfangswert zugewiesen, so ist unmittelbar nach dem Gleichheitszeichen ein Komma anzugeben oder das `PARAMS`-Kommando zu beenden.

Es können maximal 32 Parameter, jeweils durch Komma getrennt, angegeben werden. Die Gesamtlänge aller Parameter darf 1.280 Stellen nicht überschreiten.

Beispiele:

```
@params &datei1,&datei2,&spalte
```
Stellungsparameter

```
@params &datei1=dat.std,&first=%,&last=$
```
Schlüsselwortparameter

```
@params &datei1,&first=%,&last=$
```
Stellungs- und Schlüsselwortparameter

Wechseln von Arbeitsbereichen in Prozeduren

PROC *n* | *intvar*

Umschalten in einen anderen Arbeitsbereich. Dieses Kommando hat die gleiche Wirkung wie im Dialog das Kommando "0" bis "32".

Es ist jedoch zu beachten, dass nur der Arbeitsbereich gewechselt wird, die Anzeige ändert sich dadurch nicht. Soll während der Prozedur ein anderer Arbeitsbereich angezeigt werden, kann dies mit dem Kommando SETF (S. 246) erfolgen.

n

Nummer eines Arbeitsbereichs (0 bis 32).

intvar

Integer-Variable mit der Nummer eines Arbeitsbereichs (0 bis 32).

Der Arbeitsbereich bleibt solange aktuell, bis mit einem weiteren PROC-Kommando erneut in einen anderen Arbeitsbereich gewechselt wird oder mit dem Kommando END zurückgekehrt wird.

Mit PROC wird in einen anderen Arbeitsbereich gewechselt, ohne den darunter liegenden zu beenden (geschachtelte Arbeitsbereiche).

Informationen über Arbeitsbereiche anzeigen

PROC [**F**REE | **US**ED | **N**USED | **DA**REA | **PA**REA | **DS**TACK | **PS**TACK]

Informationen über Arbeitsbereiche ausgeben.

FREE | USED | NUSED |

Nummern aller benutzten und nicht benutzten sowie aller freien Arbeitsbereiche anzeigen.

DAREA | PAREA

Nummern der belegten Daten-Arbeitsbereiche bzw. Prozedur-Arbeitsbereiche während des Prozedurablaufs in aufsteigender Reihenfolge anzeigen.

DSTACK | PSTACK

Nummern der belegten Arbeitsbereiche in Prozeduren, getrennt nach Prozeduren und Daten, in der Reihenfolge der Verarbeitung (DO-Kommandos bzw. PROC-Kommandos) anzeigen.

Wird kein Operand angegeben, werden die Informationen für alle Arbeitsbereiche angezeigt.

Informationen über Arbeitsbereiche in Variablen speichern

PROC **NU**SED=*var* | **F**REE=*var* | **US**ED=*var* | **DA**REA=*var* | **PA**REA=*var* | **DS**TACK=*var* | **PS**TACK=*var*

FREE

Nummern aller freien Arbeitsbereiche in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen. Freie Arbeitsbereiche sind Arbeitsbereiche, die leer sind.

USED — Nummern der benutzten Arbeitsbereiche in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

NUSED — Nummern der nicht benutzten Arbeitsbereiche (ohne Fenster) in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

DAREA — Nummern der belegten Daten-Arbeitsbereiche während des Prozedurablaufs in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

PAREA — Nummern der belegten Prozedur-Arbeitsbereiche während des Prozedurablaufs in aufsteigender Reihenfolge, getrennt durch ein Leerzeichen.

DSTACK — Nummern der belegten Daten-Arbeitsbereiche während des Prozedurablaufs, in der Reihenfolge der Verarbeitung (`PROC`-Kommandos).

PSTACK — Nummern der belegten Prozedur-Arbeitsbereiche während des Prozedurablaufs, in der Reihenfolge der Verarbeitung (`DO`-Kommandos).

var — #I*n* | #S*n*

Als Variable kann wahlweise eine String-Variable oder eine Integer-Variable angegeben werden.

#I*n* — FREE, USED, NUSED, DAREA oder PAREA:

Die Integer-Variable enthält die Nummer des ersten Arbeitsbereichs.

DSTACK oder PSTACK:

Die Integer-Variable enthält die Nummer des letzten, d.h. des aktuellen Arbeitsbereichs.

#S*n* — Die Stringvariable enthält eine Liste der jeweiligen Arbeitsbereiche. Die Arbeitsbereichs-Nummern werden durch eine Leerstelle getrennt.

Beispiele:

```
PROC USED=#S1
PROC DSTACK=#I1   #I1 = aktueller Daten-Arbeitsbereich
```

EDT- und DMS-Fehlerschalter zurücksetzen

RESET — Nach einem DMS-Error (Fehler beim Lesen und Schreiben einer Datei) und bei sonstigen Fehlern, die bei der Ausführung von Kommandos entstehen, werden Fehlerschalter gesetzt, die mit dem Kommando `IF` abgefragt werden können. Diese Schalter können mit dem Kommando `RESET` wieder zurückgesetzt werden.

RESET **M**ODIFY — Der Änderungsstatus (zu erkennen an dem Zeichen "*" nach dem Dateinamen in der Titelzeile) eines Arbeitsbereichs wird zurückgesetzt. Es bewirkt, dass beim Beenden des EDT keine Rückfrage erfolgt, ob die Daten gesichert werden sollen.

Bemerkungszeile

REMARK *comment*

comment Kommentar, beliebiger Text (gleiche Wirkung wie Kommando `NOTE`).

Kommentare können auch zu jedem Kommando, getrennt durch ein Trennzeichen, angegeben werden. Das Trennzeichen kann im Menü `Optionen / Sonderzeichen` (S. **119**) eingestellt werden (Standard = ";"). Das Trennzeichen muss zweimal angegeben werden.

Beispiel: `@rea'&file' ;;Lesen Eingabedatei`

Beenden des Bildschirmdialogs und Abbrechen von Prozeduren

RETURN Beenden einer Prozedur. Soll die Prozedur unter Steuerung des Schleifensymbols mehrmals durchlaufen werden, so werden die noch offenen Durchläufe nicht mehr ausgeführt.

Wird das Kommando im Dialog eingegeben, so sind zwei Situationen zu unterscheiden:

a) Wurde der Dialog durch das Kommando `DIALOG` von einer INPUT-Prozedur aus eingeleitet, wird die Prozedur nach dem Kommando `DIALOG` fortgesetzt. Dabei wird der Arbeitsbereich wieder aktiviert, der bei Ausführung des Kommandos `DIALOG` aktiv war.

b) Wurde das Kommando nicht aus einer Prozedur aufgerufen, wird das Programm EDT beendet. Das Kommando hat die gleiche Wirkung wie das Kommando `HALT`.

SET bietet in 6 Formaten folgende Funktionen:

Integervariablen mit Werten versorgen (Format 1a)

- weist einen ganzzahligen Ausdruck zu
- weist ein abdruckbare Zahl als Ganzzahl zu
- weist den Inhalt einer Zeilennummer-Variablen als Ganzzahl zu
- weist die Länge einer Zeile zu
- weist den ASCII-Code einer Zeichenfolge zu
- weist die Anzahl der Sekunden seit 1.1.70 bzw. die Millisekunden seit Programmstart zu
- weist den Wochentag zu
- weist die Anzahl der Sätze zu

Float-Variablen mit Werten versorgen (Format 1b)

- weist einen Gleitpunkt-Ausdruck zu
- weist ein abdruckbare Zahl als Ganzzahl zu

Stringvariablen mit Werten versorgen (Format 2)

– weist eine Zeichenfolge zu
– weist den Inhalt einer Ganzzahlvariablen, eine Zeilennummer oder den Namen einer Zeichenfolgevariablen zu
– legt den in eine abdruckbare Zahl konvertierten Inhalt einer Ganzzahlvariablen in einer Zeichenfolgevariablen ab
– legt das abdruckbare Bild einer Zeilennummer in einer Zeichenfolgevariablen ab
– legt den Namen einer Zeichenfolgevariablen in einer Zeichenfolgevariablen ab

Linevariablen mit Werten versorgen (Format 3)

– weist eine Zeilennummer zu
– weist den Inhalt einer Ganzzahlvariablen umgewandelt in eine Zeilennummer zu
– weist eine abdruckbare Zahl als Zeilennummer zu
– weist die interne Darstellung einer Zeichenfolge zu

Werte in Zeilen ablegen (Format 4)

– legt den Inhalt einer Ganzzahlvariablen in einer abdruckbaren Form in einer Zeile ab
– schreibt den Namen einer Zeichenfolgevariablen in eine Zeile
– legt den abdruckbar gemachten Inhalt einer Zeilennummer-Variablen in einer Zeile ab

Datum und Uhrzeit (Format 5)

– legt Datum oder Uhrzeit in einer Zeichenfolgevariablen ab
– legt Datum oder Uhrzeit in einer Zeile ab

Bestimmen der neuen aktuellen Zeilennummer und Schrittweite (Format 6)

bestimmt eine neue aktuelle Zeilennummer und die Schrittweite

Integer-Variable mit Wert versorgen

SET *int-var* = [+|-] *int* [+|-|/|*|% *int*] [.....]

Mit diesem Format wird einer Integer-Variablen ein ganzzahliger Ausdruck zugewiesen. Der Wertebereich der Integer-Variablen wurde vergrößert. In Integer-Variablen können Zahlen von Minus -2^{63} bis $2^{63}-1$ (9.223.372.036.854.775.807 = 8.388.607 Terrabyte) verarbeitet werden.

int-var Integer-Variable.

int Ganze vorzeichenlose Zahl, eine Integer-Variable oder eine Float-Variable. Von der Float-Variablen wird nur der ganzzahlige Wert benutzt.

+|−|/|%|* Führt eine arithmetische Verknüpfung der angegebenen Werte durch.
+ = Addition
− = Subtraktion
/ = Division, es wird der ganzzahlige Wert zurückgegeben
% = Division, es wird der Rest zurückgegeben
* = Multiplikation

..... Es können mehrere Operationen miteinander verknüpft werden. Die Rechenoperationen werden in der angegebenen Reihenfolge ausgeführt, es gilt nicht die Regel "Punkt vor Strich", z.B. liefert das Kommando `set #I1=1+2+3*10` das Ergebnis `60`.

Beispiel:
```
set #i1=613
set #i2=60
set #i3=#i1%#i2
set #i4=#i1/#i2
```
`#i3` enthält `13` (Rest aus der Division `#i1/#i2` = 613/60)
`#i4` enthält `10` (ganzzahliges Ergebnis aus der Division)

SET *int-var* = **S**UBSTR *str*

Mit diesem Format wird eine abdruckbare Zahl in Parameter *str* einer Integer-Variablen als Wert zugewiesen (z.B. abdruckbare Zahl 17 = Integer-Wert 17)

Enthält *str* ein Vorzeichen (+ oder -), wird dies bei der Konvertierung berücksichtigt. Leerzeichen und Tausender-Trennzeichen werden unterdrückt.

SET *int-var* = *ln-var*

Mit diesem Format wird der Inhalt einer Line-Variablen in eine ganze Zahl umgewandelt und der Integer-Variablen als Wert zugewiesen (z.B. Zeilennummer 55.6 = Integer-Wert 556000).

SET *int-var* = **LENGTH** *ln* | *ln-var* | *str-var*

Mit diesem Format wird die Länge einer Zeile *ln* bzw. einer Stringvariablen ermittelt und einer Integer-Variablen als Wert zugewiesen. Existiert die Zeile nicht, wird der Wert 0 zugewiesen. Als Zeilennummer kann auch eine String-Variable angegeben werden.

SET *int-var* = **ST**RING *str*

Der ASCII-Code der Zeichenfolge *str* wird einer Integer-Variablen als Wert zugewiesen (z.B. *str* = '1', interne Darstellung X'31', der zugewiesene Wert beträgt also 49).

str

Die Zeichenfolge kann direkt in Hochkommata oder indirekt über eine Zeilennummer, eine Line-Variable oder eine String-Variable (jeweils mit Spaltenbereich möglich) angegeben werden. Eine ausführliche Beschreibung des Parameters str finden Sie auf S. 341. Die vier Byte lange Zeichenfolge besteht aus acht hexadezimalen Ziffern im ASCII-Code.

Besteht die Zeichenfolge aus mehr als vier Zeichen, werden nur die ersten vier Zeichen berücksichtigt.

SET *int-var* = T[IME] [*string*] | C[LOCK]

Die Anzahl der Sekunden oder Milli- bzw. Mikrosekunden, die seit dem 1.1.1970 00:00:00 Uhr vergangen sind, wird einer Ganzzahl-Variablen als Wert zugewiesen. Dadurch ist es möglich, Zeitmessungen durchzuführen.

T[IME]

Der Wert wird in Sekunden berechnet.

string

Zeichenfolge (Direkt in Hochkommas, Line-Variable, String-Variable oder Zeilennummer) mit einem Datum oder einer Uhrzeit. Folgende Formate sind zulässig:

Datum (TT = Tag, MM = Monat, JJ/JJJJ = Jahr, LLL = laufender Tag des Jahres):

```
JJ-MM-TT
JJ-MM-TTLLL
JJJJ-MM-TTJ
JJ-MM-TTLLL
TT.MM.JJ
TT.MM.JJLLL
TT.MM.JJJJ
TT.MM.JJJJLLL
MM/TT/JJ
MM/TT/JJLLL
MM/TT/JJJJ
MM/TT/JJJJLLL
```

Uhrzeit: (HH = Stunden, MM = Minuten, SS = Sekunden)

```
HH:MM:SS
HHMMSS
```

Ohne Parameter *string* wird die aktuelle lokale Zeit verwendet.

C[LOCK]

Der Wert wird je nach System in Milli- oder Mikrosekunden berechnet.

Beispiele:

```
set #i1 = TIME
set #i2 = CLOCK
set #i3 = TIME '10.01.2007'
set #i4 = TIME #s1
set #i5 = TIME #l1:1-10:
set #i6 = TIME 500:51-60:
```

SET *int-var* = **D**AY [*intvar-date*]

Die Nummer des aktuellen Wochentages (Sonntag = 1, Montag = 2, Dienstage = 3, Mittwoch = 4, Donnerstag = 5, Freitag = 6, Samstag = 7) bzw. des Wochentages aus der Zeitangabe in der Integer-Variablen *intvar-date* wird in die Variable übertragen.

intvar-date Integer-Variable mit einer Zeitangabe (Sekunden seit dem 1.1.1970). Fehlt dieser Parameter, wird das aktuelle lokale Datum verwendet. Die Zeitangabe der Integer-Variablen kann mit dem Kommando SET *int-var* = TIME (S. 291) erzeugt werden.

SET *int-var* = RECORDS

Die Anzahl der Sätze des aktuellen Arbeitsbereichs wird in die Variable übertragen.

Float-Variable mit Wert versorgen

SET *float-var* = [+|-] *float* [+|-|/|*| *float*] [.....]

Mit diesem Format wird einer Float-Variablen ein Ausdruck zugewiesen.

float-var Float-Variable.

float Gleitpunktzahl, ganze Zahl, eine Float-Variable oder Integer-Variable. Für die Eingabe einer Gleitpunktzahl kann wahlweise Punkt oder Komma als Dezimal-Trennzeichen verwendet werden.

+|−|/|* Führt eine arithmetische Verknüpfung der angegebenen Werte durch.
+ = Addition
− = Subtraktion
/ = Division
* = Multiplikation

..... Es können mehrere Operationen miteinander verknüpft werden. Die Rechenoperationen werden in der angegebenen Reihenfolge ausgeführt, es gilt nicht die Regel "Punkt vor Strich", z.B. liefert das Kommando `set #F1=1+2+3*10` das Ergebnis `60`.

Beispiel:

```
set #f1=613,50
set #f1=613.50
set #f2=605E20
set #f3=#i1/#i2*#f1
set #f4=#f1*1,16/1,95583
```

SET *float-var* = **S**UBSTR *str*

Mit diesem Format wird eine abdruckbare Zahl in Parameter *str* einer Float-Variablen als Wert zugewiesen. Als Dezimal-Trennzeichen ist Komma oder Punkt zulässig. Die Zahl kann als Dezimal- oder als Exponentialzahl angegeben werden. Gültige Werte sind z.B.

```
"17,50", "1,75E1", "1.75E001", "1,75E+001"
"-17,50", "-1,75E1", "-1.75E001", "-1,75+E001"
"0,1750", "1,75E-1", "1.75E-001"
```

Enthält *str* ein Vorzeichen (+ oder -), wird dies bei der Konvertierung berücksichtigt. Leerzeichen werden unterdrückt.

Um den Inhalt einer Float-Variablen in eine Zeichenfolge umzuwandeln, steht das Kommando SET (S. 295) #S*nn*=E|F.....#F*nn* zur Verfügung.

String-Variable mit Wert versorgen

SET *str-var* = *str*

Mit diesem Format wird einer String-Variablen eine Zeichenfolge zugewiesen. Der bisherige Inhalt wird überschrieben.

str-var — String-Variable.

str — Zeichenfolge, direkt in Hochkommata, als String-Variable oder als Zeilennummer (jeweils mit Spaltenangabe und Wiederholungsfaktor möglich). Eine ausführliche Beschreibung siehe S. 341. Es sind aber keine mehrfachen Zeichenfolgen möglich. Das Aneinanderhängen von Zeichenfolgen ist mit dem Kommando CREATE oder CAT möglich.

SET *str-var* = INTERNAL *int-var* | *str-var2* | *ln*

Mit diesem Format wird einer String-Variablen der Inhalt einer Integer-Variablen, der Name einer String-Variablen oder eine Zeilennummer zugewiesen. Die Werte werden im internen Format in die String-Variable geschrieben. Bei Intel-Prozessoren steht das niederwertigste Byte links.

int-var — Der Inhalt einer Integer-Variablen ist intern als vier Byte lange Dualzahl abgespeichert. Diese Dualzahl wird unverändert in die ersten acht Byte der String-Variablen *str-var* geschrieben (z.B. ganze Zahl 49 = X'0000000000000031', bei Intel-Prozessoren: X'3100000000000000').

ln — Die Zeilennummer ist intern als vier Byte langes Feld mit einer Ziffer je Halbbyte gespeichert. Dieses Feld wird unverändert in die ersten vier Byte der String-Variablen *str-var* geschrieben (z.B. Zeilennummer 56.23 = x'0008947c', bei Intel-Prozessoren: X''7c940800').

str-var2 — Der Name der String-Variablen *str-var2* wird in die ersten vier Byte der String-Variablen *str-var* geschrieben.

SET *str-var* [,*cl*] = **C**HAR | V | CL [K] *int-var*

Der Inhalt einer Integer-Variablen *int-var* wird in die entsprechende abdruckbare Zahl konvertiert und ab der Spalte *cl* in die String-Variable *str-var* geschrieben.

CHAR — Die Konvertierung führt zu einer 11 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CHAR K — Die Konvertierung führt zu einer 14 Zeichen langen Zahl mit führenden Blanks, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CL — Die Konvertierung führt zu einer 21 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

CLK — Die Konvertierung führt zu einer 27 Zeichen langen Zahl mit führenden Nullen, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

V — Die Konvertierung führt zu einer variabel langen Zahl (max. 21 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

VK — Die Konvertierung führt zu einer variabel langen Zahl (max. 27 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen und Tausender-Trennzeichen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

K — Kilo Separator. Tausender-Trennzeichen nach jeweils 3 Ziffern einfügen. Diese Zusatzoption stellt eine **Erweiterung** zum BS2000-EDT dar.

Beispiele:

```
@set #i1=1234567890
@set #i2=-1234567890
@set #i3=9223372036854775809
@set #s11=c   #i1
@set #s12=cl  #i1
@set #s13=v   #i1
@set #s14=ck  #i1
@set #s15=clk #i1
@set #s16=vk  #i1
@set #s21=c   #i2
@set #s22=cl  #i2
@set #s23=v   #i2
@set #s24=ck  #i2
@set #s25=clk #i2
@set #s26=vk  #i2
@set #s32=cl  #i3
@set #s33=v   #i3
@set #s35=clk #i3
@set #s36=vk  #i3
```

Ausgabe Kommando `STA:`

```
Integer - Variables
#I01=+1234567890 #I02=-1234567890
#I03=-9.223.372.036.854.775.807 (-8191,99P)
String - Variables
#S11(0011)= 1234567890
#S12(0021)= 00000000001234567890
#S13(0010)=1234567890
#S14(0014)= 1.234.567.890
#S15(0027)=              1.234.567.890
#S16(0013)=1.234.567.890
#S21(0011)=-1234567890
#S22(0021)=-00000000001234567890
#S23(0011)=-1234567890
#S24(0014)=-1.234.567.890
#S25(0027)=             -1.234.567.890
#S26(0014)=-1.234.567.890
#S32(0021)=-09223372036854775807
#S33(0020)=-9223372036854775807
#S35(0027)= -9.223.372.036.854.775.807
#S36(0026)=-9.223.372.036.854.775.807
```

SET *str-var* [,*cl*] = E | F [*opt*] *float-var*

Der Inhalt einer Float-Variablen *float-var* wird in die entsprechende abdruckbare Zahl konvertiert und ab der Spalte *cl* in die String-Variable *str-var* geschrieben.

E — Darstellung als Gleitpunktzahl mit Mantisse und Exponent.

F — Darstellung als Dezimalzahl, wahlweise mit Punkt oder Komma als Dezimal-Trennzeichen.

opt — [Z | B | C] [*length-bevor*] .|, [Z | B | C] [*length-after*] K

Ohne Angabe von *opt* wird eine Dezimalzahl von 20 Stellen vor und nach dem Komma mit Vorzeichen (insgesamt 42 Stellen) bzw. die Mantisse mit 20 Stellen nach dem Komma und der Exponent mit Vorzeichen und 3 Stellen (insgesamt 28 Stellen) ausgegeben. Bei einer positiven Zahl wird als Vorzeichen eine Leerstelle, bei einer negativen Zahl ein Minuszeichen vorangestellt. Als Standard für das Dezimal-Trennzeichen wird das Komma benutzt. Führende und rechtsbündige Nullen werden angezeigt (Option Z).

```
 12345678901234567890,12345678901234567890
-12345678901234567890,12345678901234567890
 1,12345678901234567890E+123
-1,12345678901234567890E-123
```

Z — Zero. Führende Nullen vor dem Komma bzw. rechtsbündige Nullen nach dem Komma bei einer Dezimalzahl bleiben erhalten. Diese Option ist Standard.

B — Blank. Führende Nullen vor dem Komma bzw. rechtsbündige Nullen nach dem Komma bei einer Dezimalzahl oder Mantisse werden durch Leerstellen ersetzt.

C — Cut. Führende Nullen vor dem Komma bzw. rechtsbündige Nullen nach dem Komma bei einer Dezimalzahl oder Mantisse werden abgeschnitten.

length-bevor — Länge der Vorkommastellen bei Darstellung als Dezimalzahl (bei Gleitpunktzahl immer 1). Standard = 20.

length-after Länge der Nachkommastellen. Werden aufgrund der Längenangabe Stellen abgeschnitten, wird die letzte Stelle kaufmännisch gerundet. Standard = 20.

. Punkt als Dezimal-Trennzeichen.

, Komma als Dezimal-Trennzeichen.

Bei fehlender Angabe "," oder "." gilt folgendes:
Sprache Deutsch = Komma,
Sprache Englisch = Punkt.

K Kilo Separator. Tausender-Trennzeichen nach jeweils 3 Ziffern einfügen. Diese Zusatzoption stellt eine **Erweiterung** zum BS2000-EDT dar. Als Tausender-Trennzeichen wird als Standard das Zeichen "**.**" benutzt. Falls als Dezimal-Trennzeichen das Zeichen "**.**" ausgewählt wurde, wird das Zeichen "," als Tausender-Trennzeichen benutzt.

Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

Beispiele:

```
@set #f1=123.456789
@set #s1=e #f1
@set #s2=e . #f1
@set #s3=e ,c10 #f1
@set #s4=e ,b10 #f1
@set #s5=e ,z10 #f1
@set #s11=f #f1
@set #s12=f . #f1
@set #s13=f c10,c2 #f1
@set #s14=f b10,b2 #f1
@set #s15=f 10,2 #f1
@sta
```

Ausgabe Kommando `STA:`

```
Float - Variables
#F01=+0000000000000000123.45678900000000000000
1.23456789000000000000E+002 = 0B0BEE073CDD5E40

String - Variables
#S01(0028)= 1.23456789000000000000E+002
#S02(0028)= 1.23456789000000000000E+002
#S03(0016)= 1,23456789E+002
#S04(0018)= 1,23456789  E+002
#S05(0018)= 1,2345678900E+002
#S11(0042)= 0000000000000000123.45678900000000000000
#S12(0042)= 0000000000000000123.45678900000000000000
#S13(0007)= 123,46
#S14(0014)=        123,46
#S15(0014)= 0000000123,46
```

SET *str-var* [*,cl*] = **C**HAR *ln-var*

Die Zeilennummer der Line-Variablen *ln-var* wird im abdruckbaren Format (0000.0001 bis 9999.9999) ab der Spalte *cl* in die String-Variable *str-var* geschrieben. Die Zeichenfolge hat immer eine Länge von neun Stellen. Führende Nullen werden durch Leerzeichen ersetzt, die fünfte Stelle enthält immer einen Punkt.

SET *str-var* [,*cl*] = **C**HAR *str-var*

Der Name der String-Variable *str-var2* wird ab der Spalte *cl* in die String-Variable *str-var* übertragen. Die Zeichenfolge hat immer eine Länge von vier Stellen. (#S00 bis #S99).

SET *str-var* = **V**ARSUBST *str-var*

Eine evtl. in der Stringvariablen enthaltene EDT-System-Variable (S. **344**), wie z.B. `!file` wird durch den Variablenwert ersetzt. Dadurch ist es möglich, auch in Dateien enthaltene Variablen zu ersetzen, da die Variablen-Ersetzung nur in direkten Strings ('string') erfolgt.

SET *str-var* = CONVX | CONVC *string*

CONVX — Der Character-String *string* wird in einen Hexa-String umgewandelt.

CONVC — Der Hexa-String *string* wird in einen Character-String umgewandelt.

Line-Variable mit Wert versorgen

SET *ln-var* = *ln*

Der Line-Variablen *ln-var* wird die Zeilennummer der Zeile *ln* zugewiesen. Beschreibung des Parameters ln siehe S. **338**.

SET *ln-var* = *int-var*

Der Inhalt der Integer-Variablen *int-var* wird in eine Zeilennummer konvertiert und der Line-Variablen *ln-var* zugewiesen. Der Wertebereich von *int-var* beträgt 1 bis 99999999, dies führt zu den Zeilennummern 0000.0001 bis 9999.9999.

SET *ln-var* = **S**UBSTR *str*

Die abdruckbare Zeilennummer im Parameter *str* wird der Line-Variablen *ln-var* zugewiesen.

str — Die Zeilennummer kann direkt in Hochkommata oder indirekt über eine Zeilennummer, eine Line-Variable oder eine String-Variable (jeweils mit Spaltenbereich möglich) angegeben werden. Eine ausführliche Beschreibung des Parameters str finden Sie auf S. **341**. Die Zeichenfolge kann ein bis neun Stellen lang sein. Bei einer Länge > 4 muss sie einen Punkt enthalten (z.B. 1234.1, 560.23, oder 0.1.)

SET *ln-var* = **ST**RING *str*

Die Zeilennummer im internen Format im Parameter *str* wird der Line-Variablen *ln-var* zugewiesen (z.B. X'00BC614E' = Zeile 1234.5678).

str — Die Zeichenfolge kann direkt in Hochkommata oder indirekt über eine Zeilennummer, eine Line-Variable oder eine String-Variable (jeweils mit Spaltenbereich möglich) angegeben werden. Eine ausführliche Beschreibung des Parameters str finden Sie auf S. **341**. Die vier Byte lange Zeichenfolge besteht aus acht hexadezimalen Ziffern im ASCII-Code.

Besteht die Zeichenfolge aus weniger als vier Zeichen, wird linksbündig mit Nullen aufgefüllt. Bei mehr als vier Zeichen werden lediglich die ersten vier Zeichen berücksichtigt.

Werte in Zeilen ablegen

Die nachfolgenden SET-Kommandos verändern immer den Inhalt der Zeile, deren Nummer in der Line-Variable enthalten ist und nicht die Line-Variable selbst.

SET *ln-var* [,*cl*] = **C**HAR | CL | V [K] *int-var*

Der Inhalt der Integer-Variablen wird in ein abdruckbares Format konvertiert und in die durch die Line-Variable angegebene Zeile abgelegt.

cl Spalte, ab der der Inhalt von *int-var* geschrieben werden soll.

CHAR Die Konvertierung führt zu einer 11 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CHAR K Die Konvertierung führt zu einer 14 Zeichen langen Zahl mit führenden Blanks, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist.

CL Die Konvertierung führt zu einer 21 Zeichen langen Zahl mit führenden Nullen, wobei das erste Zeichen entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

CLK Die Konvertierung führt zu einer 27 Zeichen langen Zahl mit führenden Nullen, wobei das Zeichen vor der ersten Zahl entweder ein Leerzeichen (positive Zahl) oder ein Minuszeichen (negative Zahl) ist. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

V Die Konvertierung führt zu einer variabel langen Zahl (max. 21 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

VK Die Konvertierung führt zu einer variabel langen Zahl (max. 27 Stellen) mit einem Minuszeichen bei negativen Zahlen vor der ersten Stelle ohne führende Nullen und Tausender-Trennzeichen. Diese Variante stellt eine **Erweiterung** zum BS2000-EDT dar.

K Kilo Separator. Tausender-Trennzeichen nach jeweils 3 Ziffern einfügen. Diese Zusatzoption stellt eine **Erweiterung** zum BS2000-EDT dar.

Beispiele:

```
@set #l11=c   #i1
@set #l12=cl  #i1
@set #l13=v   #i1
@set #l14=ck  #i1
@set #l15=clk #i1
@set #l16=vk  #i2
```

SET *ln-var* [,*cl*] = **C**HAR *str-var*

Der Name einer String-Variablen wird ab der Spalte *cl* in die durch die Line-Variable angegebene Zeile abgelegt.

Es wird immer eine vier Stellen lange Zeichenfolge mit dem Inhalt #S00 bis #S99 erzeugt.

cl Spalte, ab der der Inhalt der Name von *str-var* geschrieben wird.

SET *ln-var1* [,*cl*] = **C**HAR *ln-var2*

Der Inhalt der Line-Variablen *ln-var2* wird in eine Zeichenfolge umgewandelt und ab der Spalte *cl* in die durch die Line-Variable *ln-var1* angegebene Zeile abgelegt.

Die Konvertierung des Wertes einer Line-Variablen führt immer zu neun Zeichen in der Form *nnnn.nnnn*, wobei jedes *n* eine Ziffer darstellt. Führende Nullen werden durch ein Leerzeichen ersetzt.

cl Spalte, ab der die Zeilennummer geschrieben werden soll.

Datum und Uhrzeit

Mit dieser Variante des SET-Kommandos kann das Datum oder die Uhrzeit aus der aktuellen lokalen Systemzeit oder der in einer Integer-Variablen enthaltenen Zeit in eine Zeichenfolge umgewandelt werden. Die Zeichenfolge kann in eine Stringvariable oder in eine Zeile übertragen werden.

SET *str-var* [,*cl*] = **D**ATE [ISO[4]] [G|E] | **D**ATE4 [G|E] | **T**IME [G|E] [*intvar*]

SET *ln-var* [,*cl*] = **D**ATE [ISO[4]] [G|E] | **D**ATE4 [G|E] | **T**IME [G|E] [*intvar*]

Das Datum oder die Uhrzeit wird ab der Spalte *cl* in die durch die Zeilennummer-Variable *ln-var* angegebene Zeile oder in die Stringvariable *str-var* abgelegt. Dabei kann wahlweise das aktuelle Datum oder die aktuelle Uhrzeit oder die in der Integer-Variablen *intvar* enthaltene Zeitangabe verwendet werden.

Bis zur Version 1.505 des EDT für MS-DOS wurden die Datums- und Zeitangaben in einem vom BS2000-EDT abweichenden Format aufbereitet. Wenn dieses Format weiterhin gelten soll, muss die Option `Datum im deutschen Format` im Menü `Optionen / Weitere Optionen` (S. 127) aktiv sein. In Abhängigkeit der Parameter und der Option werden die Datums- und Zeitangaben in folgender Form aufbereitet:

Option im Menü `Optionen / Weitere Optionen`

 Datum im Deutschen Format  Datum im Deutschen Format

DATE	`mm/dd/yyjjj`	`dd.mm.yy`
DATE**4**	`mm/dd/yyyyjjj`	`dd.mm.yyyy`
DATE **I**SO	`yy-mm-ddjjj`	`yy-mm-dd`
DATE **I**SO**4**	`yyyy-mm-ddjjj`	`yyyy-mm-dd`
DATE **G**	`dd.mm.yyjjj`	`dd.mm.yy`
DATE**4 G**	`dd.mm.yyyyjjj`	`dd.mm.yyyy`
DATE **E**	`mm/dd/yyjjj`	`mm/dd/yy`
DATE**4 E**	`mm/dd/yyyyjjj`	`mm/dd/yyyy`
TIME	`hhmmss`	`hh:mm:ss`
TIME **G**	`hh:mm:ss`	`hh:mm:ss`
TIME **E**	`hhmmss`	`hhmmss`

mm = Monat bzw. Minute
dd = Tag
yy = Jahr
jjj = Jahrestag
hh = Stunde
ss = Sekunde

intvar

Integer-Variable mit einer lokalen Zeitangabe in der Form "Sekunden seit 1.1.1970". Die Zeitangabe der Integer-Variablen kann mit dem Kommando SET *int-var* = TIME (S. 291) erzeugt werden. Enthält die Integer-Variable einen Wert < 86.400, wird unterstellt, dass es sich um eine Uhrzeit von 00:00:00 bis 23:59:59 handelt (ohne Berücksichtigung der Zeitzone).

Wird diese Integer-Variable nicht angegeben, so wird die aktuelle lokale Zeit verwendet.

Beispiel:

```
set #l1=1.5
set #s1 = date          aktuelles Datum
set #i1 = time #s1      Datum in Sekunden
set #i1 = #i1 + 86400   um einen Tag erhöhen
set #s1 = date #i1      neues Datum in #s1
```

Mit diesen Kommandos wird das aktuelle Datum um 1 Tag erhöht.

Bestimmen neue Zeilennummer und Schrittweite

SET *ln* [(*inc*)] [**:** *text*]

Das Kommando bestimmt eine neue aktuelle Zeilennummer und eine neue aktuelle Schrittweite.

ln — Neue aktuelle Zeilennummer, z.B. 5. Der Minimalwert beträgt 0.0001, der Maximalwert 9999.9999. Fehlt *inc*, wird mit *ln* implizit auch die neue aktuelle Schrittweite festgelegt, z.B. legt 5 die Schrittweite 1 und 5.0 die Schrittweite 0.1 fest.

inc — Neue aktuelle Schrittweite. 0.0001 <= *inc* <= 9999.9999.

SET+ [**:** *text*] — Die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die nächste aktuelle Zeile umgeschaltet.

SET- [**:** *text*] — Die aktuelle Zeilennummer wird um die aktuelle Schrittweite vermindert oder es wird im SEQUENTIAL-Modus (siehe Anweisung @EDIT) auf die vorangehende Zeile umgeschaltet.

text — Beliebige Zeichenfolge.
Ist das erste von einem Leerzeichen verschiedene Zeichen

1. kein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen als zum text gehörende Leerzeichen behandelt. Für die Behandlung gilt:
   – text steht in der aktuellen Zeile;
   – die aktuelle Zeilennummer wird um die aktuelle Schrittweite erhöht;
   – vorhandene Tabulatorzeichen werden berücksichtigt.
2. ein Anweisungssymbol,
   werden die dem : folgenden Leerzeichen ignoriert. Folgt als zweites Zeichen
   – kein Anweisungssymbol, wird text als EDT-Anweisung interpretiert und sofort ausgeführt;
   – Anweisungssymbol, wird text als Textzeile wie bei 1) behandelt

Zeile für Dialogbox definieren

WINDEF *str-var=string* [,*textline* [*/ inputline*]] | ,P [,*textline*]]
WINDEF CLEAR
WINDEF T=*title*

Definition einer Zeile einer Dialogbox mit max. 16 Eingabefeldern, die mit dem Kommando `WINOUT` (S. **302**) ausgegeben wird. Für jede Zeile der Dialogbox ist ein WINDEF-Kommando notwendig. Zwischen den Kommandos `WINDEF` können auch andere EDT-Kommandos vorkommen, z.B. `IF`-Kommandos zum variablen Aufbau der Dialogbox.

str-var — Stringvariable, in der die eingegebene Zeichenfolge gespeichert wird. Enthält die Stringvariable vor dem Aufruf der Dialogbox bereits einen Wert, so wird diese Zeichenfolge als Standardwert in dem entsprechenden Feld angezeigt.

string Zeichenfolge, die als Feldbeschreibung links vor dem Feld angezeigt werden soll. Es kann auch eine Stringvariable oder eine Zeilennummer angegeben werden.

textline Anzahl der Zeilen für die Feldbeschreibung, Standard = 1.

inputline Anzahl der Zeilen für das Eingabefeld, Standard = 1.

P Feld für die verborgene Eingabe eines Passwortes. Die Anzahl der Eingabe-Zeilen beträgt in diesem Fall immer 1.

CLEAR Die Zeilen aus früheren `WINDEF` - Kommandos werden gelöscht.

title Zeichenfolge (Konstante, Zeilennummer, Stringvariable) für die Titelzeile.

Beispiele:

```
@windef clear
@windef t='Adresse'
@windef #s1='Nachname '
@windef #s2='Adresse',1/3
@windef #s3=#s20
```

Dialogbox erzeugen und ausgeben

WINGEN Das Kommando `WINGEN` bietet eine völlig neue Möglichkeit zur grafischen Gestaltung von Dialogboxen in EDT-Prozeduren. Durch diese Erweiterung können Sie mit der EDT-Prozedursprache Anwendungen mit einem grafischen User Interface, genauso wie z.B. mit Visual Basic o. ä. Programmiersprachen, erstellen.

Mit dem Kommando `WINGEN` kann eine benutzerdefinierte Dialogbox erstellt werden, die Editfelder, Listboxen, Comboboxen, Togglebuttons, Radiobuttons, Rahmen, Spinbuttons, Schieberegler und Tabellen enthalten kann.

Die ausführliche Beschreibung finden Sie im Kapitel 11 (S. **307**) GUI für EDT-Prozeduren - Kommando WINGEN

Dialogbox anzeigen

WINOUT [*textwidth / inputwidth*]

Anzeigen der mit den Kommandos `WINDEF` (S. **301**) definierten Dialogbox. Wurde die Schaltfläche "Abbrechen" gedrückt, so wird die Integer-Variable `#I99` auf "1" gesetzt.

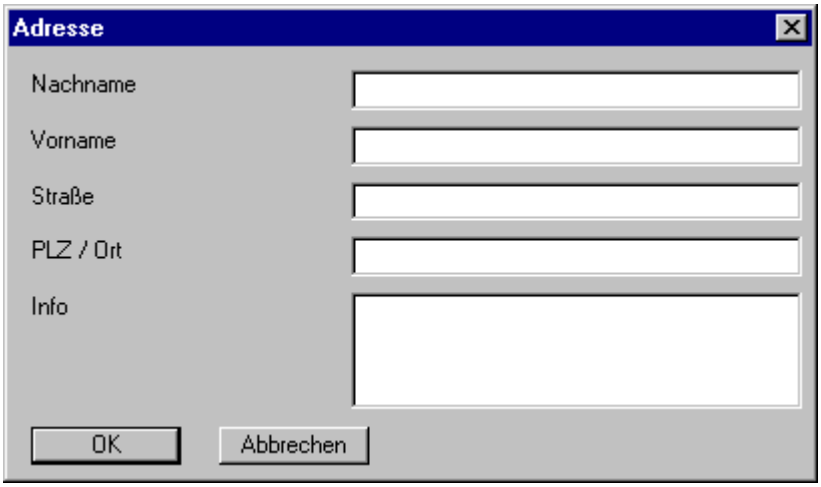
textwidth Breite des Bereiches für den Hinweistext der Eingabefelder in der Einheit "durchschnittliche Buchstabenbreite", Standard = 30

inputwidth Breite des Bereiches für die Eingabefelder in der Einheit "durchschnittliche Buchstabenbreite", Standard = 30.

Beispiel:

```
@del #s1-#s4
@windef clear
@windef t='Adresse'
@windef #s1='Nachname '
@windef #s2='Vorname'
@windef #s3='Straße'
@windef #s4='PLZ / Ort'
@set#s15='Info'
@windef #s5=#s15,3/3
@winout 24/36
@if #i99=1 goto ende
@rem Verarbeitung der Adresse
@:ende
```

Erzeugte Dialogbox:



Fentergröße ändern

WINSIZE MIN|MAX|NORM [,*arbnr*[V*viewnr*]]

Fenstergröße des Hauptfensters bzw. der Arbeitsbereichs-Fenster verändern:

MIN — Fenster minimieren, nur als Icon in Taskleiste bzw. im EDT-Hauptfenster anzeigen.

NORM — Normalgröße anzeigen in Abhängigkeit der letzten Einstellung.

MAX — Fenster maximieren, Vollbild anzeigen.

arbnr — Nummer des Arbeitsbereichs: die Änderung der Fenstergröße gilt für den angegebenen Arbeitsbereich.

V*viewnr* — Nummer der View (S. **260**) eines Arbeitsbereichs: die Änderung der Fenstergröße gilt für die angegebene View des Arbeitsbereichs.

Beispiel: `winsize min,1V2`
View 2 des Arbeitsbereichs 1 wird minimiert.

Beispiele für EDT-Prozeduren

1. Beispiel für bedingte und unbedingte Sprünge in einer INPUT-Prozedur

Die Prozedur durchsucht jede Zeile eines Arbeitsbereichs nach einem, zwei oder drei Suchbegriffen. Die Zeilen, die alle Suchbegriffe enthalten, werden am Bildschirm angezeigt und in die Datei find.dat geschrieben.

Inhalt der INPUT-Prozedur proc1:

```
@params &search1,&search2,&search3 ;; Parameter definieren
@proc 1                           ;; Wechseln in Arbeitsbereich 1
@@on ! find '&search1'            ;; in aktueller Zeile nach 1-tem Such-
@@rem                                begriff suchen
@@if .f. goto notfind             ;; Goto notfind, wenn nicht gefunden
@@if '&search2' = '' goto end     ;; Goto end, wenn 2. Suchbegriff fehlt
@@on ! find '&search2'            ;; in aktueller Zeile nach 2. Suchbe-
@@rem                                griff  suchen
@@if .f. goto notfind             ;; Goto notfind, wenn nicht gefunden
@@if '&search3' = '' goto end     ;; Goto end, wenn 3. Suchbegriff fehlt
@@on ! find '&search3'            ;; in aktueller Zeile nach 3-tem Such-
@@rem                             ;; begriff suchen
@@if .f. goto notfind             ;; Goto notfind, wenn nicht gefunden
@@goto end                        ;; Springen zu end
@@:notfind                        ;; Definieren Sprungmarke notfind
@@delete !                        ;; Aktuelle Zeile löschen
@@:end                            ;; Definieren Sprungmarke end
@end                              ;; Wechseln in Arbeitsbereich 0
@do1 !=%,$,1                      ;; Prozedur 1 aufrufen, erste Zeile %,
@rem                              ;; letzte Zeile $, Schrittweite 1,
@rem                              ;; Schleifensymbol !
@print                            ;; alle Zeilen anzeigen
@w'find.dat'o                     ;; Arbeitsbereich in Datei find.dat
@rem                              ;; schreiben
```

Aufruf der INPUT-Prozedur `proc1`:

```
edt testdat -iproc1 Meier Hans 10.12.65
```

Von der Datei `testdat` werden alle Sätze angezeigt und in die Datei `find.dat` geschrieben, die in einer beliebigen Spalte jeweils den Nachnamen "Meier", den Vornamen "Hans" und das Geburtsdatum "10.12.65" enthalten. Beim Laden des EDT wird als erster Parameter der Name der zu bearbeitenden Datei und als zweiter Parameter mit dem Schalter "`-i`" der Name der INPUT-Prozedur angegeben. Nach der INPUT-Prozedur folgen die Parameter für die INPUT-Prozedur.

2. Beispiel für eine äußere Schleife

Mit äußeren Schleifen können Prozeduren mehrmals vollständig durchlaufen werden. Sollen jedoch nur Teile einer Prozedur mehrmals durchlaufen werden, müssen diese Teile in Form von inneren Schleifen formuliert werden.

```
@proc 1                          ;; Wechseln in Arbeitsbereich 1
@@column 10 on ! insert !:27-36: ;; in aktuelle Zeile Spalten einfügen
@end                              ;; in Arbeitsbereich 0 wechseln
@do 1 !=1,5,1                     ;; Do-Prozedur aufrufen f. die Zeilen 1 - 5
```

Aufruf der INPUT-Prozedur `proc1`:

```
edt testdat2 -iproc2
```

In diesem Beispiel werden in den Zeilen 1 - 100 die Spalten 27-36 nach der Spalte 10 eingefügt. Die Prozedur wird für jede Zeile einmal durchlaufen, beginnend mit der Zeile 1. Bei jedem Durchlauf wird das Schleifensymbol "!" um die Schrittweite 1 erhöht. Die Prozedur wird so oft durchlaufen, bis das Schleifensymbol die Nummer der als letzte zu bearbeitende Zeile erreicht hat, in diesem Fall die Zeile 100. Mit dem Schleifensymbol kann die gerade aktuell zu bearbeitende Zeile angesprochen werden.

3. Beispiel für eine innere Schleife

Mit inneren Schleifen können auch Zeilen mit variabler Schrittweite bearbeitet werden. Außerdem muss nicht die gesamte Prozedur durchlaufen werden

```
@proc 5                       ;; Wechseln in Arbeitsbereich 5
@@params &file,&begin,&end    ;; Parameter definieren
@@rea'&file'                  ;; Datei einlesen
@@on & find '&begin'          ;; Suchen Beginn des zu verarbeitenden
@@rem                            Bereichs
@@if .false. goto end         ;; bei Fehler Abbruch
@@set #l1=?                   ;; Speichern Zeilennummer in Line-Variable #L1
@@on & find '&end'            ;; Suchen Ende des zu verarbeitenden Bereichs
@@if .false. goto end         ;; bei Fehler Abbruch
@@set #l2=?                   ;; Speichern Zeilennummer in Line-Variable #L2
@@set #l0=#l1+1l              ;; Setzen #L0 auf 2-te Zeile des Bereichs
@@:begin                      ;; Beginn der Schleife
@@if L #l0 >= #l2 goto copy   ;; Abfrage auf Ende des Bereichs
@@set #l3=#l0-1l              ;; Setzen #L0 auf vorhergehende Zeile
@@if #L0:1-5:<>#l3:1-5: goto add  ;;Vergleich des Schlüssels aktuelle Zeile
@@rem                            mit letzter Zeile
@@delete #l3                  ;; vorhergehende Zeile mit gleichem Schlüssel
@@rem                            löschen
@@:add                        ;; Sprungziel add
@@set #l0=#l0+1l              ;; Erhöhen #L0 um eine Zeile
@@goto begin                  ;; Sprung zu Schleifenbeginn
@@:copy                       ;; Sprungziel copy
@@proc2                       ;; Wechseln in Arbeitsbereich 2
@@copy #l1.-#l2(0) last       ;; Kopieren des verarbeiteten Bereichs in
@@rem                            Arbeitsbereich 2
@@end                         ;; Wechseln in Arbeitsbereich 5
@@delete                      ;; Löschen des Arbeitsbereichs 0
@@:end                        ;; Sprungziel end
@end                          ;; Wechseln in Arbeitsbereich 0, Ende Prozedur
@do 5(file1,start,ende)       ;; Do-Prozedur aufrufen für die Datei file1
@do 5(file2,anfang,schluss)   ;; Do-Prozedur aufrufen für die Datei file2
@proc2                        ;; Wechseln in Arbeitsbereich 2
@sort                         ;; Sortieren der Daten in Arbeitsbereich 2
@w'sort.dat'o                 ;; Schreiben sortierte Daten in Datei sort.dat mit
@rem                          ;;  Option Overwrite
```

Aufruf der INPUT-Prozedur `proc3`:

```
edt -iproc3
```

In diesem Beispiel wird der Beginn und Ende eines zu verarbeitenden Bereichs gesucht und in den Line-Variablen #L1 und #L2 gespeichert. Die Schlüssel (Spalte 1 bis 5) jeder Zeile werden mit dem Schlüssel der vorhergehenden Zeile verglichen. Zeilen mit gleichem Schlüssel werden gelöscht. In der Line-Variablen #L0 steht die Nummer der aktuell zu verarbeitenden Zeile. Sie wird um eine relative Zeilennummer erhöht, d.h. die Line-Variable #L0 wird nicht um eine feste Schrittweite erhöht, sondern ihr wird mit dem Kommando `SET #L0=#L0+1L` die nächste belegte Zeilennummer zugewiesen (z.B. nach Zeile 20 kommt Zeile 50 oder nach Zeile 20 kommt Zeile 20.0001).

Die Prozedur `proc5` wird zweimal aufgerufen, jeweils mit verschiedenen Dateien. Die bearbeiteten Zeilen werden in den Arbeitsbereich 2 kopiert. Die Option `LAST` bewirkt, dass die Daten an das Ende des Arbeitsbereichs 2 angehängt werden. Nach der Verarbeitung der zwei Dateien wird der Arbeitsbereich 2 sortiert. Die sortierten Daten werden in die Datei `sort.dat` geschrieben.

11 GUI für EDT-Prozeduren - Kommando WINGEN

Allgemeines

Das Kommando WINGEN bietet eine völlig neue Möglichkeit zur grafischen Gestaltung von Dialogboxen in EDT-Prozeduren. Durch diese Erweiterung können Sie mit der EDT-Prozedursprache Anwendungen mit einem grafischen User Interface, genauso wie z.B. mit Visual Basic o. ä. Programmiersprachen, erstellen.

Für einfache Fenster (bis zu 16 Eingabefelder, links Text, rechts Eingabefelder) steht auch das Kommando WINDEF (S. 301) zur Verfügung.

Mit dem Kommando WINGEN kann eine benutzerdefinierte Dialogbox erstellt werden, die Editfelder, Listboxen, Comboboxen, Togglebuttons, Radiobuttons, Rahmen, Spinbuttons, Schieberegler und Tabellen enthalten kann.

Bei nahezu allen Elementen kann die Darstellungseigenschaft, ob sichtbar oder unsichtbar, ob enabled (schwarz) oder disabled (hellgrau) zu Beginn festgelegt werden sowie unter dem Prozedurablauf umgeschaltet werden. Auch eine Textänderung ist bei Editfeldern, Festtexten, Boxrahmen sowie Buttons aller Art möglich.

Die Rückgabewerte für die Editfelder sind Zeichenfolgen und werden in einer Stringvariablen bereitgestellt, die Rückgabewerte aller anderen Feldtypen sind numerischer Art und werden in einer Integer-Variable abgestellt, Listboxen und Comboboxen können den Wert in einer Integer-Variable, Line-Variable oder Stringvariable ablegen. Die Selektion von Zeilen in einer Table bewirkt ein Setzen des Selektionsschalters in dem dazugehörigen Satz des Arbeitsbereiches.

Es können maximal 32 Fenster (0-31) definiert werden. Während des Ablaufs einer Prozedur kann eine Fensternummer freigegeben und mit einem anderen Fenster belegt werden. Eine Dialogbox kann max. **256** Fensterobjekte enthalten.

Mit der Taste TAB bzw. Shift/TAB kann von Feld zu Feld gesprungen werden, hierbei ist die Reihenfolge der Anweisungen zur Definition der Elemente eines Fensters ausschlaggebend.

Obwohl das Kommando WINGEN auch in der Kommandozeile per Hand eingegeben werden kann, ist dessen Verwendung vor allem in einer Prozedur zu sehen.

Referenzvariable

Referenzvariable sind Stringvariable bzw. Integer-Variable oder Line-Variable, die einem Objekt in einem WINGEN-Fenster zugewiesen sind. Referenzvariable haben zwei unterschiedliche Aufgaben:

a) Bei der Ausgabe des Fensters wird der Inhalt der Referenzvariablen zur Initialisierung des Objektes verwendet. Dies ist z.B. bei einem Editfeld der Inhalt des Editfeldes, bei einer Listbox oder Combobox die Nummer der selektierten Zeile in der Box.

b) Wenn ein Button betätigt wird, der eine Übernahme der Variablen auslöst (Option UPDATE), so wird der aktuelle Inhalt des Objektes oder die aktuelle Selektion in die dazugehörige Referenzvariable übernommen.

Die String-Variablen, Line-Variablen oder Integer-Variablen werden erst zum Zeitpunkt der echten Fensterausgabe (Kommando WINGEN OUT) festgestellt und ausgewertet. Der Stand der Variable zum Zeitpunkt der Definition des Fensterobjektes ist nicht relevant.

Rückgabeinformationen

Wenn das Fenster durch einen Button mit der Option "CLOSE, UPDATE" geschlossen wird, so werden die Referenzvariablen der Felder mit den aktuellen Werten versorgt. Die Integer-Variable I99 erhält den Wert, der als Rückkehrcode definiert ist.

Wenn eine Button-Prozedur durch einen Button des Typs "PROC*n*, UPDATE" gestartet wird, so werden vor Aufruf der Prozedur die Referenzvariablen mit den aktuellen Werten und die Integer-Variable I99 mit dem Rückkehrcode versorgt.

Die Versorgung der Referenzvariablen geschieht wie folgt:

Editfelder: Die referenzierte String-Variable wird mit dem Inhalt des Editfeldes geladen.

Togglebuttons: In der referenzierten Integer-Variable wird der Wert 1 für eingeschaltet bzw. der Wert 0 für ausgeschaltet bereitgestellt.

Radiobuttons: In der referenzierten Integer-Variable wird die Nummer des ausgewählten Radiobuttons der RADIOGROUP in der Reihenfolge derer Definition bereitgestellt. Der erste Button hat die Nummer 0.

Listboxen und Comboboxen: Als Referenz-Variable kann eine Integer -, Line - oder eine String-Variable angegeben werden:

Integer-Variable In der referenzierten Integer-Variable wird die Nummer des ausgewählten Eintrags bereitgestellt. Der erste Eintrag hat die Nummer 0.

Line-Variable: In der referenzierten Line-Variable wird die Zeilennummer des ausgewählten Eintrags bereitgestellt.

String-Variable In der referenzierten String-Variable wird der Inhalt des ausgewählten Eintrags bereitgestellt.

Spinbuttons und Schieberegler: Der hieraus resultierende Wert wird in der referenzierten Integer-Variable abgespeichert.

Tabelle: In einer Tabelle können mehrere Zeilen selektiert werden (durch Klicken mit der Maus und Festhalten der Shift-Taste bzw. Ctrl-Taste). Die Selektion einer Zeile in der Tabelle bewirkt gleichzeitig die Selektion des entsprechenden Satzes im Arbeitsbereich. Um in einer Prozedur auf diese Eigenschaft zugreifen zu können, wurde das Kommando IF erweitert: Mit dem Kommando IF *line-var* .. selected ..... kann die Eigenschaft abgefragt werden.

Vorbesetzung der Fensterobjekte

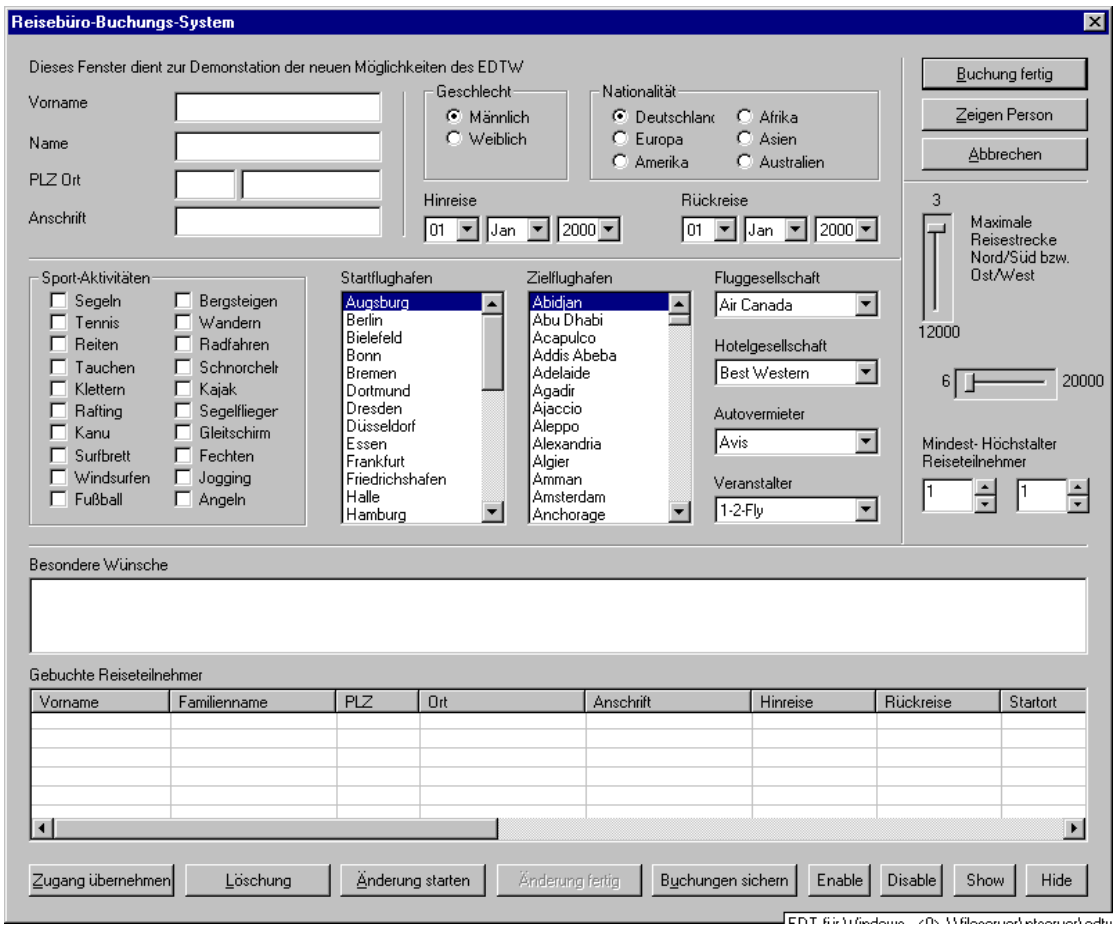
Die Rückgabewerte in den Stringvariablen für die Editfelder und in den Integervariablen und den Linevariablen der Datenarbeitsbereiche für die Togglebuttons, Radiobuttons, Comboboxen und Listboxen sowie die Integervariablen für die Spinbuttons und Schieberegler dienen ebenfalls für eine Vorbesetzung der Selektion bei Ausgabe des Fensters. Das bedeutet, eine wiederholtes Kommando WINGEN OUT liefert dieselbe Selektion der Felder, wie diese bei Verlassen des Fensters mit einem Button mit dem Attribut UPDATE bereitgestellt wurde.

Wenn eine Vorbesetzung stattfinden soll, so sind in den String-, Line- oder Integer-Variablen die entsprechenden Angaben bereitzustellen.

Beispiel

Ein umfangreiches Beispiel zu diesem Kommando ist der Prozedur WINGEN.CMD (auf dem Installationsverzeichnis) zu entnehmen, das mit dem Kommando INPUT aus dem Arbeitsbereich 0 gestartet werden kann. Darin werden mit einem WINGEN-Fenster Angaben für eine Buchung in einem fiktiven Reisebüro angefordert. Dieses Beispiel nutzt alle Varianten und Möglichkeiten des Kommandos WINGEN. Zu diesem Beispiel gehört auch die Buchungsdatei WINGEN.BUCHUNG. Eine ausführliche Beschreibung der Prozedur finden Sie in der Datei WINGEN.DOC. (weitere Beispiele wingen2.cmd und wingen4.cmd).

Beispiel eines Fensters aus der Prozedur `WINGEN.CMD`:



Beschreibung der Syntaxelemente für alle Varianten

Objektname

:*name*: Symbolischer Name eines Objektes in einer Dialogbox

Unter der Bezeichnung *name* kann das Element angesprochen und mit den Anweisungen `ENABLE`, `DISABLE`, `SHOW`, `HIDE` und `TEXT` in seiner Darstellung geändert werden. Ebenso kann *name* in den Maß- und Positionsangaben zur Bezugnahme auf bereits bestehende Fensterobjekte verwendet werden.

Der Wert *name* muss über alle Fenster hinweg eindeutig sein, darf also nicht in anderen Fenstern ein weiteres Mal verwendet werden. Diese Einschränkung hat jedoch den Vorteil, dass in einem anderen Fenster ein Fensterobjekt geändert werden kann bzw. eine Bezugnahme auf ein Objekt eines anderen Fensters erfolgen kann. Der Wert *name* darf maximal 8 Stellen lang sein. Das erste Zeichen darf keine Ziffer sein.

Windows-Attribute für die Objekte

(*winattr*) (*x , y, width, height* [, *font, fcolor, bcolor*])

Die Attribute beschreiben die Position des linken oberen Eckes des Objektes (x,y), die Breite (*width)* und die Höhe (*height)* des Objekts sowie die Schriftart, die Schriftfarbe und die Hintergrund-Farbe.

Die Maß- und Positionsangaben *x, y, width* und *height* sind Einheiten von Dialogfenstern, eine in MS-Windows allgemein verwendete Größe. Etwa 12 Einheiten sind die Höhe einer Zeile, etwa 4 Einheiten sind die Breite eines Buchstabens.

Durch Verweis auf bereits vorher definierte Objekte ist es möglich, nur wenige Maßangaben absolut anzugeben und die weiteren Objekte relativ zu diesen Objekten zu definieren, so dass bei Verschiebung von mehreren Objekten nur die absoluten Angaben geändert werden müssen.

Alle Integer-Angaben Können als absolute Zahl oder als Integer-Variablen angegeben werden. Für die Angaben *x*, *y*, *width* und *height* können komplexe Ausdrücke mit beliebig vielen Rechenoperation mit Verweisen, absoluten Zahlen und Integer-Variablen angegeben werden.

x	*item* [+ | - | / | * *item*] [....]	Position des Objektes vom linken Fensterrand
y	*item* [+ | - | / | * *item*] [....]	Position des Objektes vom oberen Fensterrand
width	*item* [+ | - | / | * *item*] [....]	Breite des Objekts
height	*item* [+ | - | / | * *item*] [....]	Höhe des Objekts

item = int | *int-var* | *name*/L | *name*/R | *name*/T | *name*/B | *name*/W | *name*/H

int	absolute Zahl.
int-var	Integer-Variable.
name/L	*x*-Position des linken Randes des Objekts *name.*
name/R	*x*-Position des rechten Randes des Objekts *name.*
name/T	*y*-Position des oberen Randes des Objekts *name.*
name/B	*y*-Position des unteren Randes des Objekts *name.*
name/W	Breite des Objekts *name.*
name/H	Höhe des Objekts *name.*

font Die Nummer der Schriftart, die mit dem Kommando `WINGEN FONT` definiert wurde. Wird "0" angegeben, so wird ein evtl. gültiger Wert aus dem Kommando `WINGEN STD` durch den Systemwert überschrieben.

fcolor Nummer der Schriftfarbe, die mit dem Kommando `WINGEN COLOR` definiert wurde. Wird "0" angegeben, so wird ein evtl. gültiger Wert aus dem Kommando `WINGEN STD` durch den Systemwert überschrieben.

bcolor Nummer der Hintergrundfarbe, die mit dem Kommando `WINGEN COLOR` definiert wurde. Wird "0" angegeben, so wird ein evtl. gültiger Wert aus dem Kommando `WINGEN STD` durch den Systemwert überschrieben.

Beispiele:

```
(10,10,100,15)
(10,10,100,15,1,1)
(feld1/L,feld1/L+15,feld/H)
```

Datei wingen2.cmd:

```
@proc10
@del
  @@ msgbox'ok Button gedrückt'
@end
@wingen         use=1
@wingen         clear
@wingen         font=1,'Courier New',20
@wingen         font=2,'Times',12
@wingen         color=1,255,0,0
@wingen         color=2,0,255,0
@wingen         title='wingen2.cmd - Positionsangaben'
@rem            1. Zeile = Text + Eingabefeld
@wingen :text1: static=(10,10,90,12,1,1,2),'Feld1'
@wingen :feld1: edit=(100,text1,50,text1,2,2,1),#s10
@rem winattr=         100,10,   50,15
@rem            2. Zeile = Text + Eingabefeld
@wingen :text2: static=(text1,text1+2,text1,text1),'Feld2'
@wingen :feld2: edit=(feld1,text2,feld1,feld1),#s11
@rem            winattr=100,   30,   50,   15
@rem            3. Zeile = Text + Eingabefeld
@wingen :text3: static=(text1,text2+2,text1,text1),'Feld3'
@wingen :feld3: edit=(feld1,text3,feld1,feld1),#s12
@rem            winattr=100,   50,   50,   15
@rem            4. Zeile = nur Text
@crea         #s13:'Langer Text zu Feld4 über die ganze Breite'
@wingen :text4: static=(text1,text3+12,feld1/r-text1/l,15),#s13
@rem            winattr=   10,      70,      150-10   ,15
@rem            5. Zeile = nur Eingabefeld
@wingen :feld4: edit=(text4,text4+2,text4,feld1),#s14
@rem            winattr=100,     90,  140,   15
@wingen button=(text1,feld4+10,feld1,feld1),'&OK',update,proc10,10
@wingen button=(feld1,feld4+10,feld1,feld1), '&Cancel',close,cancel,99
@wingen out=1
```

Datei wingen4.cmd:

```
@wingen use=1
@wingen clear
@wingen title='Test alle Objekte'
@wingen line=(8,10,1,300)
@wingen line=(61,10,1,300)
@wingen :f1:static=(10,10,50,12),'Test STATIC'
@wingen :f2:edit=(f1,f1+2,f1,f1),#s1
@wingen :f3:box=(f1,f2+2,f1,f1+12),'Test Box'
@wingen :f4:line=(f1,f3+7,f1,1)
@wingen :f4b:radiogroup=#i10
@wingen :f4c:radio=(f1,f4+7,f1,f1),'Radio1'
@wingen :f5:radio=(f1,f4c+2,f1,f1),'Radio2'
@wingen :f6:spin=(f1,f5+2,f1,f1),1,1000,#i11
@wingen :f7:toggle=(f1,f6+2,f1,f1),'Toggle1',#i12
@wingen :f7b:toggle=(f1,f7+2,f1,f1),'Toggle2',#i13
@wingen :f8:button=(f1,f7b+2,f1,f1),'Button',close,update,10
@wingen :f9:slider=(f1/l+10,f8+2,f1-20,f1),10,10,1,99,#i14
@wingen :f10:listbox=(f1,f9+2,f1,36),#l10,#l11,10,#i15
@wingen :f11:combobox=(f1,f10+2,f1,48),#l10,#l11,10,#i16
@wingen :f12:table=(f1,f11+2,100,48),#l10,#l11,11,'PLZ~40~Ort~120'
@#i10=0
@#i11=523
@#i12=0
@#i13=1
@#i14=3566
@#l10=1
@#l11=5
@proc10
@del
Zeile1
Zeile2
Zeile3
Zeile4
Zeile5
@end
@proc11
@del
80331~München
23669~Timmendorfer Strand
13088~Berlin
21073~Hamburg
56729~Langscheid
@end
@wingen out=1
```

Kommando WINGEN

Mit dem Kommando WINGEN können Sie sowohl die einzelnen Objekte der Dialogbox definieren und verändern als auch die Dialogbox ausgeben. Vorerst funktioniert dieses Kommando nur ab Windows-Version WINDOWS-NT 4.0 bzw. Windows Me. Für jede Objektart bzw. für jede Funktion gibt es eine Variante des Kommandos:

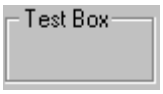
Definition der Dialogbox:

SIZE	Definiert die Position und Größe des Fensters
TITLE	Definiert die Überschrift des Fensters
FONT	Definiert eine Schriftart
COLOR	Definiert eine Farbe für die Schrift oder den Hintergrund
STD	Definiert eine Standard-Schriftart und die Hintergrundfarbe

Funktionen:

USE	Weist eine Fensternummer für alle folgenden WINGEN-Kommandos zu
CLEAR	Löschen des Fensterpuffers
DISABLE	Schaltet ein Objekt auf nicht erreichbar (disabled) und zeigt es hellgrau an.
ENABLE	Schaltet ein Objekt auf erreichbar (enabled) und zeigt es schwarz an.
HIDE	Schaltet ein Objekt eines Fensters auf nicht sichtbar.
SHOW	Schaltet ein Objekt eines Fensters sichtbar.
TEXT	Ändert den Text eines Objekts und Eigenschaften für Schaltflächen.
ATTR	Ändert Attribute (Font und Farbe) eines Objektes.
ATTRP	Ändert die Werte-Angaben zum PROGRESS-BAR.
OUT	Gibt ein Fenster aus und wartet auf Eingabe.
IF	IF WINGEN n = EXISTING: Abfrage, ob ein Fenster noch existiert, siehe hierzu Kommando IF (S. 280)

Übersicht der Objekte einer Dialogbox:

BOX		Rahmen, der mit einem erklärenden Text versehen ist.
BUTTON		Schaltfläche
COMBOBOX		Combobox mit Daten aus einem Arbeitsbereich.
EDIT		Eingabefeld, das einzeilig oder mehrzeilig sein kann.
LINE		Senkrechte oder waagrechte Linie
LISTBOX		Listbox mit Daten aus einem Arbeitsbereich.

PROGRESS Progress-Bar (Fortschrittsanzeige).

RADIOGROUP Gruppe von mindestens zwei Radiobuttons.

RADIO Radiobutton.

SPIN Spinbutton zur Festlegung eines numerischen Wertes.

SLIDER Schieberegler zur Festlegung eines numerischen Wertes.

STATIC Festtext, z.B. als Erklärung

TOGGLE Togglebutton.

TABLE Tabelle mit Kopfzeile und Spalten

Kommando WINGEN - Fensterattribute

Definition von Schriftarten

WINGEN FONT = *n, 'font', size* [,BOLD] [,UNDERLINE] [,ITALIC] [,BLINK] [,STRIKEOUT] [,WIDTH=*width*] [,WEIGHT=*weight*] [,ANGLE=*angle*]

Hier können bis zu 64 Schriftarten definiert werden. Die Nummer der Schriftart kann bei den Attributen zu den einzelnen Elementen einer Dialogbox angegeben werden. Siehe Beschreibung Attribute (S. **310**).

n Nummer der Schriftart

font Name der Schrift in Hochkommas eingeschlossen.

size Schriftgröße als Integer oder Integer-Variable.

BOLD Fettschrift, die gleiche Wirkung wie WEIGHT=700.

UNDERLINE Unterstrichen.

ITALIC Kursivschrift.

BLINK Die Schriftfarbe wechselt in einer Frequenz von 500ms zwischen Originalfarbe und Komplementärfarbe.

STRIKEOUT Durchgestrichen.

width Schriftbreite bzw. bei Proportionalschrift die durchschnittliche Schriftbreite in Pixel (logical units).

weight Stärke der Schrift. Zulässig sind die Werte 100,200,300,400,500,600,700,800 und 900. Bei den meisten Schriften wirken sich nur die Werte 400 (normal) und 700 (fett) aus. WEIGTH=700 hat die gleiche Wirkung wie die Angabe BOLD.

angle Gedrehte Schrift in zehntel Winkelgraden, z.B. 900 ist senkrecht von unten nach oben, 2700 ist senkrecht von oben nach unten, 1800 = waagrecht gedreht (Schrift steht auf dem Kopf). Diese Option wirkt nur bei TrueType-Schriften.

Beispiele:

```
@wingen font=1,'Courier New',20
@wingen font=2,'Times',#i98,bold,
@wingen font=1, 'Courier New'14,width=8
@wingen font=2, 'Arial',18,weight=400
@wingen font=3, 'Arial',18,weight=700
@wingen font=4, 'Fixedsys',8,angle=1800
@wingen font=5, 'MS Serif',22
@wingen font=6, 'Modern',14
@wingen font=7, 'Marigold',28
@wingen font=8, 'Lucida Console',14
@wingen font=9, 'Impact',14
@wingen font=10,'Palatino',14
@wingen font=11,'Letter Gothic',14
@wingen font=12,'Tahoma',14,bold
@wingen font=13,'Times',18,blink
@wingen font=14,'Helvetica',14,italic
@wingen font=15,'Georgia',14,underline
```

Definition von Farben

WINGEN COLOR = *n,rot,grün,blau*

Hier können bis zu 64 Farben definiert werden. Die Nummer der Farbe kann bei den Attributen zu den einzelnen Elementen einer Dialogbox angegeben werden. Siehe Beschreibung Attribute (S. 310).

n Nummer der Farbe.

rot Der Wert von 0 - 255 für Rot

grün Der Wert von 0 - 255 für Grün

blau Der Wert von 0 - 255 für Blau

Beispiel:

```
@wingen color=1,255,0,0
@wingen color=2,192,192,192
```

Definition von Standardwerten für die Schrift und die Hintergrundfarbe

WINGEN STD = [*font*] [,*fcolor*] [,*bcolor*]

Hier können Standardwerte für die Schriftart, die Schriftfarbe und die Hintergrundfarbe definiert werden. Siehe Beschreibung Attribute (S. 310).

font Die Nummer der Schriftart, die mit dem Kommando `WINGEN FONT` definiert wurde.

fcolor Nummer der Schriftfarbe, die mit dem Kommando `WINGEN COLOR` definiert wurde.

bcolor Nummer der Hintergrundfarbe, die mit dem Kommando `WINGEN COLOR` definiert wurde.

Beispiel: `@wingen std=1,5,6`

Definition der Fenstergröße

WINGEN SIZE = *x*, *y* [,*width, height*]

Die Attribute beschreiben die Position des linken oberen Eckes (x,y) sowie die Breite (*width)* und die Höhe (*height)* der Dialogbox.

Die Maß- und Positionsangaben *x, y, width* und *height* sind Einheiten von Dialogfenstern, eine in MS-Windows allgemein verwendete Größe. Etwa 12 Einheiten sind die Höhe einer Zeile, etwa 4 Einheiten sind die Breite eines Buchstabens.

x Position der Dialogbox vom linken Fensterrand.

y Position der Dialogbox vom oberen Fensterrand.

width Breite der Dialogbox.

height Höhe der Dialogbox.

Wenn das Kommando ganz entfällt oder als Lage des Fensters der Wert `0,0` angegeben wird, so wird das Fenster in der Mitte des Bildschirms plaziert.

Werden die Größenangaben nicht angegeben, so erfolgt eine interne Berechnung der Größe. Es wird der rechte und untere Rand des Fensters nach den im Fenster enthaltenen Objekten berechnet. Dabei wird rechts bzw. unten ein Freiplatz gelassen, der so breit ist wie der kleinste Freiplatz links bzw. oben. Wenn eine Combobox den unteren Rand der Objekte darstellt, so kann es zu Darstellungsproblemen kommen, da als Größe der Combobox der Wert im aufgeklappten Zustand berechnet wird und somit ein großer Freiplatz unten entstehen kann.

Beispiel:

```
@wingen size=10,10,530,400
@wingen size=0,0,530,400
@wingen size=10,10
```

Text der Kopfzeile

WINGEN TITLE=*string*

Die angegebene Zeichenfolge wird als Kopfzeile des Fensters angezeigt. Es kann ein beliebiger Text in Form einer Zeichenfolge oder als String-Variable angegeben werden.

Beispiel:

```
@wingen title='Reisebüro-Buchungs-System'
@wingen title=#s1
```

System-Schaltflächen in der Titelzeile

Folgende System-Schaltflächen werden automatisch generiert:

 Das Fenster bzw. das Hauptfenster wird zum Icon verkleinert. Das Icon wird in der Regel links über der Taskleiste angezeigt. Falls die Option "Taskleiste automatisch ausblenden" eingestellt ist, kann sich das Icon auch hinter der Taskleiste befinden.

 Das Fenster wird von der maximalen Größe zur zuletzt eingestellten Fenstergröße verkleinert.

 Das Fenster wird bis zur maximalen Größe vergrößert.

 Das Fenster wird geschlossen.

Kommando WINGEN - Funktionen

Ändern von Attributen (Schriftart und Farbe) eines Objekts

WINGEN ATTR=*name* [,*font*] [,*fcolor*] [, *bcolor*]

name Name des Objekts.

Die Attribute Schriftart, Schriftfarbe und Hintergrund werden geändert.

font Die Nummer der Schriftart, die mit dem Kommando `WINGEN FONT` definiert wurde. Wird "0" angegeben, so wird ein evtl. gültiger Wert aus dem Kommando `WINGEN STD` durch den Systemwert überschrieben.

fcolor Nummer der Schriftfarbe, die mit dem Kommando `WINGEN COLOR` definiert wurde. Wird "0" angegeben, so wird ein evtl. gültiger Wert aus dem Kommando `WINGEN STD` durch den Systemwert überschrieben.

bcolor Nummer der Hintergrundfarbe, die mit dem Kommando `WINGEN COLOR` definiert wurde. Wird "0" angegeben, so wird ein evtl. gültiger Wert aus dem Kommando `WINGEN STD` durch den Systemwert überschrieben.

Ändern von Attributen für den PROGRESS-BAR

WINGEN ATTRP=*name* [,*begin*] [,*end*] [, *act*]

name Name des Objekts.

Die Attribute für den PROGRESS-BAR (S. **330**) werden geändert.

begin Der Anfangs-Wert für den PROGRESS-BAR in Form einer Zahl oder einer Integer-Variablen.

end Der Ende-Wert für den PROGRESS-BAR in Form einer Zahl oder einer Integer-Variablen.

act Der aktuelle Wert für den PROGRESS-BAR in Form einer Zahl oder einer Integer-Variablen.

Beispiel:

```
@wingen attrp=progress1,0,15,#i30
@wingen attrp=progress1,#i20,#i21,#i30
```

Löschen des Fensterpuffers

WINGEN CLEAR

Alle Angaben für die mit einen vorhergegangenen `WINGEN USE` bestimmte Fensternummer und der damit verbundenen Objekte werden gelöscht und zurückgesetzt. Bei der Definition eines Fensters sollte dieses Kommando immer das erste Kommando sein.

Objekte nicht erreichbar

WINGEN DISABLE=*name*

Das Objekt eines Fensters wird auf nicht erreichbar geschaltet. Es wird dann von Windows i.d.R. hellgrau dargestellt. Dieses Attribut kann nicht für den Typ `LINE` angewandt werden, da es immer das Attribut `DISABLED` aufweist.

Dieses Kommando kann in einer Benutzerprozedur für eine Schaltfläche vorhanden sein, aber auch während der Erstellungsphase für ein `WINGEN`-Fenster, wenn ein soeben definiertes Objekt als Ausgangsstatus dieses Format aufweisen soll.

Beispiel: `@wingen disable=obj1`

Objekte erreichbar

WINGEN ENABLE=*name*

Das Objekt eines Fensters wird auf erreichbar geschaltet. Es wird dann von Windows i.d.R. schwarz dargestellt. Dies ist die Standardeinstellung. Dieses Attribut kann nicht für den Typ `LINE` angewandt werden, da dieser immer das Attribut `DISABLED` aufweist.

Beispiel: `@wingen enable=obj1`

Objekte unsichtbar

WINGEN HIDE=*name*

Das Objekt eines Fensters wird auf unsichtbar geschaltet. Dieses Attribut kann auf alle Elementtypen angewandt werden.

Beispiel: `@wingen hide=obj1`

Dialogbox ausgeben

WINGEN OUT= *winnum* [**,** ASYNC | CLOSE | WAIT=*sec*]

Die Dialogbox wird ausgegeben. Wenn die Ausgabe eines Fensters öfters bzw. an unterschiedlichen Stellen durchgeführt werden soll, so genügt nur die Verwendung dieses Kommandos. Das Fenster muss nicht vor jeder Ausgabe definiert werden. Dies kann z.B. einmalig zu Beginn der Prozedur erfolgen.

winnum Nummer des Fensters, wie sie mit dem Kommando `WINGEN USE` erklärt wurde.

ASYNC Das Fenster wird asynchron ausgegeben, d.h. Die Prozedur wird nach Ausgabe des Fensters fortgesetzt. Die Daten für das Fenster können mit den Kommandos `WINGEN TEXT` (S. **322**), `WINGEN ATTR` (S. **319**) und `WINGEN ATTRP` (S. **319**) während des Prozedur-Ablaufs geändert werden, z.B. zum Anzeigen von Zwischen-Ergebnissen. Das Fenster wird mit dem Kommando WINGEN OUT=*winnum*,CLOSE geschlossen. Zur Prüfung, ob das Fenster noch existiert, z.B. weil es vom Benutzer explizit geschlossen wurde, steht das Kommando `IF WINGEN` *winnum* = `EXIST` (S. **280**) zur Verfügung.

In einem asynchronen Fenster sollten nur STATIC-Objekte und PROGRESS-Objekte vorkommen. Bei anderen Fenster-Objekten, z.B. EDIT oder Buttons mit einer Aktion ist durch den asynchronen Verlauf nicht gewährleistet, dass die Verarbeitung in der Prozedur und die Verarbeitung durch einen Mausklick im asynchronen Fenster in der gewollten Reihenfolge ausgeführt wird. Außerdem kann es zu Kollisionen kommen, wenn die Hauptverarbeitung und die asynchrone Verarbeitung die gleichen Ressourcen (Arbeitsbereich, Variablen, Daten) verwenden.

Beim Schließen des Fensters werden folgende Verarbeitungen durchgeführt:

a) Das Fenster wird durch das Kommando WINGEN OUT=n,CLOSE geschlossen: Das Fenster wird ohne Verarbeitung geschlossen, auch wenn bei CANCEL- oder Default-Button (S. 324) eine Aktion definiert ist.

b) Das Fenster wird vom Benutzer durch einen Mausklick auf die System-Schaltfläche ☒ geschlossen: Es wird die Verarbeitung des Buttons, der als Cancel-Button definiert wurde, durchgeführt. Falls kein Cancel-Button definiert wurde, wird das Fenster ohne Aktion geschlossen.

c) Das Fenster wird vom Benutzer durch einen Mausklick auf einen Button geschlossen: Es wird die für den Button definierte Verarbeitung durchgeführt.

d) Falls das Fenster beim Beenden des EDTW noch existiert, wird es ohne Aktion geschlossen.

WAIT=*sec* Das Fenster wird synchron ausgegeben und automatisch nach *sec* Sekunden bzw. nach einem vorherigen Mausklick auf einen Button geschlossen. Beim Schließen des Fensters werden folgende Verarbeitungen durchgeführt:

a) Das Fenster wird nach Ablauf der angegebenen Sekunden geschlossen: Es wird die Verarbeitung des Buttons, der als Default-Button (S. 324) definiert ist, durchgeführt. Falls kein Default-Button definiert ist, wird das Fenster ohne Aktion geschlossen.

b) Das Fenster wird vom Benutzer durch einen Mausklick auf die System-Schaltfläche ☒ geschlossen: Es wird die Verarbeitung des Buttons, der als Cancel-Button definiert ist, durchgeführt. Falls kein Cancel-Button definiert ist, wird das Fenster ohne Aktion geschlossen.

c) Das Fenster wird vom Benutzer durch einen Mausklick auf einen Button geschlossen: Es wird die für den Button definierte Verarbeitung durchgeführt.

CLOSE Ein asynchrones Fenster wird geschlossen. Das Fenster wird ohne Verarbeitung geschlossen, auch wenn bei CANCEL- oder Default-Button (S. 324) eine Aktion definiert ist.

Beispiele:

```
@wingen out=1
@wingen out=1,ASYNC
@wingen out=1,CLOSE
@wingen out=1,WAIT=15
```

Objekte sichtbar

WINGEN SHOW=*name*

Das Objekt eines Fensters wird auf sichtbar geschaltet. Dies ist die Standardeinstellung. Dieses Attribut kann auf alle Elementtypen angewandt werden.

Beispiel: `@wingen show=obj1`

Text eines Objekts ändern

WINGEN TEXT=*name* , *string* [, *param*]

Der Text eines Objekts wird geändert. Die Darstellungsattribute ändern sich hierdurch nicht. Dieses Kommando ist nur bei den Elementtypen `STATIC`, `BOX`, `EDIT`, `RADIOGROUP`, `RADIO`, `BUTTON` und `TOGGLE` anwendbar.

param Wird der Text für einen Button geändert, so müssen nach dem Text die gleichen Parameter angegeben werden wie es auch bei der Definition der Schaltfläche erforderlich ist. Damit kann die Schaltfläche einem anderen Arbeitsbereich zugeordnet werden bzw. einen anderen Übergabecode liefern oder eine andere Funktion ausführen, nachdem der Text geändert wurde.

Beispiele:

```
@wingen text=obj1,'Neuer Wert des Box-Rahmens'
@wingen text=can,'Änderung abbrechen',cancel,proc,10,48
@wingen text=obj3,'Mann/Frau',#i70
```

Fensternummer zuweisen

WINGEN USE=*winnum*

Allen folgenden `WINGEN`-Kommandos außer dem Kommando `WINGEN OUT` wird die angegebene Fensternummer zugewiesen. Somit ist es möglich, bei der dynamischen Definition von Objekten für mehrere Fenster immer wieder zwischen den Fenstern umzuschalten.

winnum Zahl zwischen 0 und 31

Beispiel: `@wingen use=1`

Kommando WINGEN - Fensterobjekte

BOX **Definition eines Rahmens**

WINGEN [*:name:*] **BOX =** *winattr* , *string*

Dieses Kommando generiert einen Rahmen für eine Gruppe von Objekten mit einem erklärenden Text in der horizontalen oberen Linie.

string Text, der in der oberen horizontalen Linie des Rahmens angezeigt wird. Wird hier ein Leerstring (zwei Hochkommas) angegeben, wird ein Rahmen ohne Text generiert.

Beispiel: `@wingen box=(10,105,140,116),'Besondere Wünsche'`

BUTTON **Definition Schaltfläche**

WINGEN [*:name:*] **BUTTON=***winattr, string* { ,CLOSE | ,PROC*n* | ,PROC*int-var* | ,INPUT='*file'* [*(params)*]} [,DEFAULT | ,CANCEL], [,UPDATE] , *retcode*

Mit diesem Kommando wird eine Schaltfläche definiert. Bei einem Klick mit der Maus auf diese Schaltfläche wird entweder das Fenster geschlossen oder eine EDT-Prozedur ausgeführt.

string Beschriftung des Buttons.

UPDATE Wird der Parameter angegeben, so werden beim Drücken des Buttons alle Referenzvariablen der Dialogbox mit den Werten der Felder versorgt.

CLOSE Das `WINGEN`-Fenster wird geschlossen und das Kommando nach `WINGEN OUT` ausgeführt.

PROC*n* Die EDT-Prozedur in Ebene *n* wird mit dem EDT-Kommando `DO` aufgerufen.
PROC*int-var* Die EDT-Prozedur in *int-var* wird mit dem EDT-Kommando `DO` aufgerufen.
INPUT=*'file'* Die EDT-Prozedur *file* wird mit dem EDT-Kommando `INPUT` aufgerufen.

Die Prozedur kann durch Versorgung eines Rückgabewertes in der Integer-Variablen `#I99` den weiteren Ablauf wie folgt beeinflussen:

#I99=0 Das Fenster wird nicht neu aufgebaut.
#I99=-1 Das Fenster wird mit den Daten aus den Referenzvariablen neu aufgebaut.
#I99> 0 Das Fenster wird geschlossen und nach dem Kommando `WINGEN OUT` fortgefahren. Die Integer-Variable `#I99` behält diesen Wert.

DEFAULT Diese Schaltfläche wird als Default-Button erklärt. Dieser Button wird auch bei Drücken der Eingabetaste (Enter) und bei einem Fenster mit der Option WAIT (S. **320**) nach Zeitablauf ausgelöst. Es darf nur ein Button pro Fenster mit dieser Eigenschaft versehen werden.

CANCEL Diese Schaltfläche wird als Cancel-Button erklärt. Das Schließen des Fensters mit dem Close-Icon ☒ rechts oben in der Titelzeile oder Drücken der Taste ESCAPE löst die hinter diesem Button stehende Aktion aus. Es darf nur ein Button pro Fenster mit der dieser Eigenschaft versehen werden.

Existiert kein Button mit dieser Eigenschaft, so wird bei Eintreffen von ESCAPE oder dem Close-Icon das Fenster geschlossen, es erfolgt keine Übernahme der Referenzvariablen. In die Variable `#I99` wird der Wert 2 geladen.

retcode

Vor dem Aufruf der Prozedur bzw. nach dem Schließen des Fensters wird in die Integer-Variable `#I99` der Wert *retcode* geladen. Damit kann für mehrere Schaltflächen dieselbe Prozedur aufgerufen werden. Anhand des Übergabecodes ist feststellbar, welche Schaltfläche gedrückt wurde.

Beispiele:

```
wingen :can: button=(100,10,30,12),'Abbruch',close,cancel,2
```

Es wird ein Button definiert, der mit dem Wert "Abbruch" beschriftet ist. Bei Drücken dieses Buttons wird das Fenster geschlossen und nach dem Kommando `WINGEN OUT` fortgefahren. Die Daten in den Feldern des Fensters werden nicht in die Referenzvariablen übernommen. Die Integer-Variable `#I99` wird mit dem Wert 2 versorgt. Das Schließen des Fensters mit dem Close-Icon rechts oben in der Titelzeile oder Drücken der Taste ESCAPE löst ebenfalls die hinter diesem Button stehende Aktion aus.

.

```
@wingen :ok:button=(100,10,30,12),'OK',close,update,11
```

Es wird ein Button definiert, der mit dem Wert `'OK'` beschriftet ist. Bei Drücken dieses Buttons wird das Fenster geschlossen und nach dem Kommando `WINGEN OUT` fortgefahren. Die Daten in den Feldern des Fensters werden in die Referenzvariablen übernommen. Die Integer-Variable `#I99` wird mit dem Wert 11 versorgt.

```
@ wingen :ok:button=(100,10,30,12),'OK',proc1,update,default,11
```

Es wird ein Button definiert, der mit dem Wert `'OK'` beschriftet ist. Bei Drücken dieses Buttons wird das Kommando `DO 1` aufgerufen. Vor Aufruf der Prozedur werden die Daten aus den Feldern in die Referenzvariablen übernommen. Die Integer-Variable `#I99` wird mit dem Wert 11 versorgt. Der Button wird auch durch Drücken der Eingabetaste ausgelöst.

WINGEN [:*name*:] **COMBOBOX=***winattr, line-var1, line-var2, arb,* { *int-var* | *line-var* | *string-var* }

Beschreibung siehe Kommando `WINGEN LISTBOX` (S. **328**).

EDIT **Definition Eingabefeld EDIT**

WINGEN [:*name*:] **EDIT=***winattr*, *str-var* [,PASS]

Es wird ein Eingabefeld erzeugt, in dem Zeichen eingegeben werden können. Wenn die Höhe des Feldes den Wert 12 übersteigt, wird ein mehrzeiliges Eingabefeld und vertikaler Scrollmöglichkeit erstellt, andernfalls ein einzeiliges Eingabefeld mit horizontaler Scrollmöglichkeit.

str-var Der Inhalt der String-Variablen wird als Anfangswert des Eingabefeldes angezeigt. Nach dem Beenden des Dialogs bzw. nach Anklicken eines Buttons wird der Inhalt des Eingabefeldes in die String-Variable übertragen.

PASS Feld für Passwort-Eingabe: Die in diesem Feld eingegebenen Daten werden als "*" angezeigt.

Beispiele:

```
@wingen edit=(80,25,100,14),#s2
@wingen :plz:edit=(80,60,30,14),#s4,PASS
```

LINE **Definition einer Linie**

WINGEN [:*name*:] **LINE=***winattr*

Wird als Breite der Wert "1" angegeben, so wird von der angegebenen Position eine senkrechte Linie nach unten in der angegebenen Höhe gezogen.

Wird als Höhe der Wert "1" angegeben, so wird von der angegebenen Position eine waagrechte Linie nach rechts in der angegebenen Breite gezogen.

Beispiele:

```
@wingen line=(10,230,510,1) ;; waagrechte Linie
@wingen line=(430,65,1,90)  ;; senkrechte Linie
```

LISTBOX **Definition einer Listbox oder Combobox**

WINGEN [:*name*:] **LISTBOX=***winattr*, *line-var1, line-var2, arb,* { *int-var* | *line-var* | *string-var* } [{,PROC*n* | ,PROC*int-var* | ,INPUT='*file*' [*(params)*]} , *retcode*]

WINGEN [:*name*:] **COMBOBOX=***winattr*, *line-var1, line-var2, arb,* { *int-var* | *line-var* | *string-var* }

Es wird eine Listbox oder Combobox erzeugt, die mit den Werten aus Zeilen des Arbeitsbereiches *arb* versorgt wird, beginnend mit der Zeilennummer in *line-var1* und endend mit der Zeilennummer in *line-var2*. Wenn eine sortierte Darstellung gewünscht ist, so müssen die Zeilen im Arbeitsbereich bereits sortiert sein, evtl. durch Verwendung des Kommandos `SORT` (z.B. `SORT #L2.-#L3`). Wenn mehr Zeilen enthalten sind, als Platz definiert wurde, wird ein vertikaler Scrollbar erzeugt.

Die Listbox wird permanent in der definierten Höhe dargestellt. Die Combobox wird vorerst einzeilig dargestellt, nach dem Anklicken "aufgeklappt" und nach der Auswahl wieder einzeilig angezeigt.

Falls die längste Zeile die Breite der Listbox bzw. Combobox überschreitet, wird ein horizontaler Scrollbar eingeblendet

line-var1 Zeilennummer der ersten Zeile, die in der Liste angezeigt werden soll.

line-var2 Zeilennummer der letzten Zeile, die in der Liste angezeigt werden soll.

arb Arbeitsbereich, in dem die Daten für die Listbox/Combobox stehen.

Referenzvariable:

Als Referenzvariable kann eine Integer-Variable, eine Line-Variable oder eine String-Variable angegeben werden.

int-var Die Referenzvariable enthält die Position des selektierten Eintrags. Der erste Eintrag befindet sich auf Position 0.

line-var Die Referenzvariable enthält die Zeilennummer (im Wert zwischen *line-var1* und *line-var2*) des selektierten Eintrags.

string-var Die Referenzvariable enthält den Inhalt des selektierten Eintrags.

Wenn vor Ausgabe der Dialogbox ein unzulässiger Wert in der Referenzvariablen angegeben wird (zu hoher Index in *int-var*, Zeilennummer nicht im Bereich von *line-var1* bis *line-var2*, Zeichenfolge in *string-var* ist nicht in der Box enthalten), so wird der erste Eintrag in der Box selektiert.

Aktion für Doppelklick (nur Listbox):

PROC*n* Bei einem "Doppelklick" wird die EDT-Prozedur *n* mit dem EDT-Kommando `DO` aufgerufen.

PROC*int-var* Bei einem "Doppelklick" wird die EDT-Prozedur in *int-var* mit dem EDT-Kommando `DO` aufgerufen.

INPUT=*'file'* Bei einem "Doppelklick" wird die INPUT-Prozedur *file* mit dem EDT-Kommando `INPUT` aufgerufen.

Die Prozedur kann durch Versorgung eines Rückgabewertes in der Integer-Variablen `#I99` den weiteren Ablauf wie folgt beeinflussen:

#I99=0 Das Fenster wird nicht neu aufgebaut.
#I99=-1 Das Fenster wird mit den Daten aus den Referenzvariablen neu aufgebaut.
#I99> 0 Das Fenster wird geschlossen und nach dem Kommando `WINGEN OUT` fortgefahren. Die Integer-Variable `#I99` behält diesen Wert.

retcode Vor dem Aufruf der Prozedur wird in die Integer-Variable `#I99` der Wert *retcode* geladen. Damit kann für mehrere Schaltflächen dieselbe Prozedur aufgerufen werden. Anhand des Übergabecodes ist feststellbar, welche Listbox ausgewählt wurde.

Beispiele:

```
@wingen listbox=(160,115,80,120),#l2,#l3,6,#i53
@wingen listbox=(250,115,80,120),#l4,#l5,6,#l6
@wingen listbox=(250,115,80,120),#l4,#l5,6,#s6,proc10,1
@wingen combobox=(250,115,80,120),#l4,#l5,6,#s6
```

PROGRESS **Definition eines Progress-Bar**

WINGEN [:*name*:]

PROGRESS=*winattr*[, *beg* , *end*]

Der aktuelle Wert für den Progress-Bar wird mit dem Kommando `WINGEN ATTRP` (S. 319) gesetzt.

beg Anfangswert für den Progress-Bar.

end Ende-Wert für den Progress-Bar.

Der Anfangs- und Ende-Wert kann auch beim Kommando `WINGEN ATTRP` angegeben werden.

Beispiel:

```
@wingen :prog1: progress = (10,80,90,12),0,100
```

RADIO **Definition einer Gruppe von Radiobuttons**

WINGEN [:*name*:] **RADIOGROUP=***int-var*

Radiobuttons werden im Allgemeinen dann verwendet, wenn von zwei oder mehreren Schaltzuständen nur einer gestattet ist. Die Definition der Gruppe dient nur der Beschreibung des Rückgabewertes, es wird kein sichtbares Objekt erzeugt. Jedes Anklicken (Selektieren) eines Buttons deselektiert alle anderen Buttons einer Gruppe.

int-var Die Referenzvariable enthält die Nummer, welcher Button der Gruppe selektiert ist. 0 ist der erste Button der Gruppe.

WINGEN [:*name*:] **RADIO=***winattr*, *string*

Radiobuttons für eine Gruppe müssen unmittelbar hintereinander definiert werden, es dürfen keine anderen Objekte für das Fenster dazwischen definiert werden.

string Text zu dem Radiobutton.

Beispiel:

```
@wingen :Vers:radiogroup=#i10
@wingen radio=(100,42,70,10),'Lebensversicherungen'
@wingen radio=(100,32,70,10),'Sachversicherungen'
@wingen radio=(100,52,70,10),'Haftpflichtversicherungen'
```

SLIDER **Definition eines Schiebereglers**

WINGEN [:*name*:] **SLIDER=***winattr*, *width-value, height-value, min, max, int-var*

Wenn die Breite größer als die Höhe ist, so wird ein waagrechter Schieberegler, andernfalls ein senkrechter Schieberegler definiert.

Der Schieberegler, ähnlich einem Regler an einem HiFi-Mischpult, hat zwei Endpunkte mit einem Minimalwert und einem Maximalwert. Die beiden Staticfelder, die diese Werte anzeigen werden automatisch erzeugt.

Bei einem senkrechten Schieberegler wird der Minimalwert bzw. Maximalwert über bzw. unter dem Schieberegler zentriert in einem Staticfeld dargestellt. Diese beiden Felder werden automatisch erzeugt. Die Höhe des gesamten Objekts errechnet sich somit aus der Höhe des Minimalwert-Feldes, des Slider-Buttons und des Maximalwert-Feldes.

Bei einem waagrechten Schieberegler wird der Minimalwert bzw. der Maximalwert links bzw. rechts neben dem Schieberegler in einem Staticfeld dargestellt. Diese beiden Felder werden automatisch erzeugt. Die Breite des gesamten Objekts errechnet sich somit aus der Breite des Minimalwertes, des Slider-Buttons und des Maximalwertes.

Die Position sowie die Breite und Höhe, die mit dem Parameter *winattr* definiert wird, bezieht sich nur auf den Schieberegler ohne die beiden Static-Felder für die Darstellung des Minimal- und Maximalwertes.

width-value Breite der Static-Felder zur Anzeige des Minimalwertes bzw. Maximalwertes.

height-value Höhe der Static-Felder zur Anzeige des Minimalwertes bzw. Maximalwertes.

min Absolute Zahl oder Integer-Variable, die den niedrigsten Wert definiert.

max Absolute Zahl oder Integer-Variable, die den höchsten Wert definiert.

int-var Die Referenzvariable enthält den Wert, der mit dem Schieberegler gesetzt wird.

Beispiele:

```
@wingen slider = (440,80,18,50),22,10,1,12000,#i10
@wingen slider = (455,150,50,18),22,10,1,20000,#i11
```

SPIN **Definition eines Spinbuttons**

WINGEN [:*name*:] **SPIN=***winattr, min, max, int-var*

Ein Spinbutton stellt eine einfache Möglichkeit dar, einen numerischen Wert einzugeben, der sich in einem bestimmten Bereich bewegt. Es handelt sich um zwei übereinander liegende Buttons mit Pfeilen, einer nach oben und einer nach unten. Links daneben befindet sich ein Editfeld, in dem der mit den Pfeilen ausgelöste Wert auch sichtbar dargestellt wird. Es ist auch möglich, durch Eingabe in das Editfeld den Wert zu ändern.

Die Breite des Objekts umfasst die Breite des eigentlichen Spinbuttons und die des Editfeldes. Die Breite des Spinbuttons ist der halbe Wert der Höhe des Feldes. Der Rest der Gesamtbreite wird für das Editfeld verwendet.

min Absolute Zahl oder Integer-Variable, die den niedrigsten Wert definiert.

max Absolute Zahl oder Integer-Variable, die den höchsten Wert definiert.

int-var Die Referenzvariable enthält den Wert, der mit dem Spinbutton gesetzt wird.

Beispiele:

```
@wingen spin=(100,200,50,12),1,120,#i4
@wingen spin=(120,200,50,12),#i85,#i86,#i5
```

STATIC **Definition Festtext**

WINGEN [:*name*:] **STATIC=***winattr*, *string*

string Zeichenfolge oder String-Variable mit dem Text. Der Festtext hat die Eigenschaft "automatischer Zeilenumbruch", so dass ein längerer Text, der nicht nebeneinander Platz findet, unterhalb ausgegeben werden kann, sofern für die Höhe genügend Platz definiert worden ist.

Beispiele:

```
@wingen static=(160,105,80,10),'Startflughafen'
@wingen :Zeile2:static=(250,105,80,10),#s2
```

TABLE **Definition einer Tabelle**

WINGEN [:*name*:] **TABLE=***winattr*, *line-var1, line-var2, arb, header* [, { *int-var* | *line-var* | *string-var* } [{,PROC*n* | ,PROC*int-var* | ,INPUT='*file'* [*(params)*]} , *retcode*]

Eine Tabelle ist eine Darstellung von mehreren Zeilen und Spalten. Sie wird z.B. vom Windows-Explorer im rechten Fensterteil verwendet, um Dateiname, Größe, Typ, Änderungsdatum und Attribute darzustellen. In der Regel hat eine Tabelle eine Kopfzeile mit den Spaltenüberschriften.

In der Tabelle werden Sätze des Arbeitsbereiches *arb* dargestellt, beginnend mit der Zeile in *line-var1* und endend mit der Zeile in *line-var2*. Die Einteilung der Spalten sowie die Festlegung der Spaltenüberschriften erfolgt durch die Daten in *header*.

line-var1 Zeilennummer der ersten Zeile, die in der Liste angezeigt werden soll.

line-var2 Zeilennummer der letzten Zeile, die in der Liste angezeigt werden soll.

arb Arbeitsbereich, in dem die Daten für die Listbox/Combobox stehen.

header Für eine Spalte sind zwei Angaben erforderlich: die Spaltenüberschrift und die Spaltenbreite, getrennt durch das Zeichen "~" (Tilde). Als Trennzeichen zwischen den Spalten ist ebenfalls das Zeichen Tilde anzugeben. Am Ende darf keine Tilde angegeben werden.

Die Sätze, mit denen die Tabelle aus dem Arbeitsbereich gefüllt werden soll, müssen einen festgelegten Aufbau haben: Die Felder sind durch das Zeichen Tilde zu trennen. Nach dem letzten Feld darf keine Tilde vorhanden sein.

Referenzvariable bei Doppelklick:

Als Referenzvariable kann eine Integer-Variable, eine Line-Variable oder eine String-Variable angegeben werden.

int-var Die Referenzvariable enthält die Position des selektierten Eintrags. Der erste Eintrag befindet sich auf Position 0.

line-var Die Referenzvariable enthält die Zeilennummer (im Wert zwischen *line-var1* und *line-var2*) des selektierten Eintrags.

string-var Die Referenzvariable enthält den Inhalt des selektierten Eintrags.

Wenn vor Ausgabe der Dialogbox ein unzulässiger Wert in der Referenzvariablen angegeben wird (zu hoher Index in *int-var*, Zeilennummer nicht im Bereich von *line-var1* bis *line-var2*, Zeichenfolge in *string-var* ist nicht in der Box enthalten), so wird der erste Eintrag in der Box selektiert.

Aktion bei Doppelklick:

PROC*n* Bei einem "Doppelklick" wird die EDT-Prozedur *n* mit dem EDT-Kommando `DO` aufgerufen.

PROC*int-var* Bei einem "Doppelklick" wird die EDT-Prozedur in *int-var* mit dem EDT-Kommando `DO` aufgerufen.

INPUT=*'file'* Bei einem "Doppelklick" wird die INPUT-Prozedur *file* mit dem EDT-Kommando `INPUT` aufgerufen.

Die Prozedur kann durch Versorgung eines Rückgabewertes in der Integer-Variablen `#I99` den weiteren Ablauf wie folgt beeinflussen:

#I99=0 Das Fenster wird nicht neu aufgebaut.
#I99=-1 Das Fenster wird mit den Daten aus den Referenzvariablen neu aufgebaut.
#I99> 0 Das Fenster wird geschlossen und nach dem Kommando `WINGEN OUT` fortgefahren. Die Integer-Variable `#I99` behält diesen Wert.

retcode Vor dem Aufruf der Prozedur wird in die Integer-Variable `#I99` der Wert *retcode* geladen. Damit kann für mehrere Schaltflächen dieselbe Prozedur aufgerufen werden. Anhand des Übergabecodes ist feststellbar, welche TABLE ausgewählt wurde.

Beispiel:

```
@set #l10=1
@set #l11=5
@wingen :f12:table=(10,10,100,48),#l10,#l11,11,'PLZ~40~Ort~120',#s
5,proc11,38
```

Datensätze in Arbeitsbereich 11:

```
80331~München
23669~Timmendorfer Strand
13088~Berlin
21073~Hamburg
56729~Langscheid
```

TOGGLE **Definition eines Togglebuttons**

WINGEN [*:name:*] **TOGGLE=***winattr, string, int-var*

Ein Togglebutton kann ein- oder ausgeschaltet werden. Die Stellung eines Togglebuttons ist unabhängig von anderen Togglebuttons.

string Text zu dem Togglebutton.

int-var Referenzvariable: 1 = selektiert, 0 = nicht selektiert

Beispiel:

```
@wingen toggle=(20,115,50,10),'Wartung',#i60
```

12 Beschreibung der EDT-Syntaxvariablen

Integer-Variable

int-var Integer-Variable #I0 ..... #I99. Eine Integer-Variable kann einen Wert zwischen -4,61[18] und +4,61[18] enthalten. Sie ist mit dem Wert 0 vorbesetzt. Die Integer-Variablen #I0 und #I1 werden von den Kommandos ON, CAT und CUT verändert.

Die Integer-Variablen werden unter dem Namen #I0, #I1, ... #I99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann auch umdefiniert werden (Menü Optionen / Sonderzeichen (S. 119)). Die Integer-Variablen können mit dem Kommando SET angelegt oder geändert werden.

Float-Variable

float-var Float-Variable #F0 ..... #F99. Das Setzen einer Float-Variablen kann mit einem Dezimalwert (z.B. 123,50) oder mit Mantisse und Exponent (z.B. 1,2E10) erfolgen. Die Float-Variablen können mit dem Kommando SET (S. 292) angelegt oder geändert werden. Mit dem Kommando SET (S. 295) *str-var* F .....*float-var* kann die Float-Variable abdruckbar in eine Zeichenfolge umgewandelt werden. Mit dem Kommando STA (S. 250) können die Inhalte der Float-Variablen angezeigt werden.

Die Float-Variablen werden unter dem Namen #F0, #F1, ... #F99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann auch umdefiniert werden (Menü Optionen / Sonderzeichen (S. 119)).

Line-Variable

ln-var Line-Variable #L0 ..... #L99. Eine Line-Variable kann eine Zeilennummer zwischen 0000.0001 und 9999.9999 enthalten. Die Line-Variablen sind mit Wert 0 vorbesetzt. Dies ist ein ungültiger Wert. Demnach müssen den Line-Variablen vor ihrer Benutzung zugelassene Werte zugewiesen werden.

Die Line-Variable #L0 wird von dem Kommando ON verändert.

Die Line-Variablen werden unter dem Namen #L0, #L1, ... #L99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann auch umdefiniert werden (Menü Optionen / Sonderzeichen (S. 119)). Die Line-Variablen können mit dem Kommando SET angelegt oder geändert werden.

String-Variable

str-var String-Variable #S0 ..... #S99. In einer String-Variablen können bis zu 32.767 Zeichen gespeichert werden. String-Variablen können mit dem Kommando SET oder CREATE angelegt oder geändert werden. Sie werden unter dem Namen #S0, #S1, ... bis #S99 angesprochen. Die erste Stelle des Namens (Zeichen "#") kann auch umdefiniert werden (Menü Optionen / Sonderzeichen (S. 119)). Die Daten in Stringvariablen werden immer in UNICODE codiert und bei Verwendung in einem Arbeitsbereich in den Zielcode umgewandelt.

Spaltenangabe

col :*cl1* | *cl1-cl2* [,*cl1* | *cl1-cl2* .....] :

Eine oder mehrere durch Komma getrennte Spalten oder Spaltenbereiche:

cl1 Spaltenbereich von der Spalte *cl1* bis zum Zeilenende. Soll nur eine Spalte ausgewählt werden, so muss die Schreibweise :*cl1-cl2*: verwendet werden. Dabei haben *cl1* und *cl2* den gleichen Wert. Statt einer absoluten Zahl kann auch eine Integer-Variable (#I0 - #I99) angegeben werden, in der die Spaltennummer gespeichert ist.

cl1-cl2 Spaltenbereich von Spalte *cl1* bis Spalte *cl2* (Grenzen eingeschlossen). Die Spalte *cl2* muss größer oder gleich Spalte *cl1* sein.

Beispiele:

```
:4:
```
Spalte 4 bis Zeilenende

```
:1-5:
```
Spalte 1 bis 5

```
:5-5:
```
nur Spalte 5

```
#I1=1
:#I1-5:
```
Spalte 1 bis 5

```
#I1=900
#I2=1000
:#I1.-#I2:  oder  :#I1.-.#I2:
```

Spalte 900 bis 1000. Vor dem Bindestrich muss ein Punkt angegeben werden, damit der Bindestrich von dem Operator "Minuszeichen" (Subtraktion) unterschieden werden kann.

Zeilennummer

ln *zlnr* | *symb-ln* | *ln-var*

zlnr Direkte Angabe der Zeilennummer von 0.0001 bis 9999.9999.

symb-ln Symbolische Zeilennummer:

- % Niedrigste Zeilennummer des Arbeitsbereichs.
- * Aktuelle Zeilennummer des Arbeitsbereichs.
- $ Höchste Zeilennummer des Arbeitsbereichs. Ist der Arbeitsbereich leer oder enthält er nur eine Zeile, so ist % = $.
- ? Zeilennummer der ersten Trefferzeile eines früheren ON-Kommandos.

Die symbolischen Zeilennummern beziehen sich immer auf den aktuellen Arbeitsbereich. Im Gegensatz dazu gelten die Line-Variablen #L0 - #L99 global für alle Arbeitsbereiche.

ln-var Eine der Line-Variablen #L0 - #L99.

Zusammengesetzte Ausdrücke für Zeilennummern

Die Zeilennummer kann auch durch die Angabe einer Addition oder Subtraktion einer der oben beschriebenen Zeilenangaben (*zlnr*, *symb-ln*, *ln-var*) dargestellt werden. Als Operand dieser Rechenoperation ist auch die Angabe einer Integer-Variablen (#I0-#I99) oder einer relativen Zeilennummer (1L ... *n*L) zulässig.

1L ... *n*L — oder #I0 ... #I99: Relative Zeilennummer. Mit *n*L bzw. #I*n* werden *n* Zeilen übersprungen, unabhängig davon, welche Schrittweite zwischen den einzelnen übersprungenen Zeilen liegt.

Beispiel:

```
Arbeitsbereich:  1.00 String1
                 3.00 String2
                 9.00 String3

set #L1=1
copy #L1+2L to 999

Arbeitsbereich:  1.00 String1
                 3.00 String2
                 9.00 String3
               999.00 String3
```

Folgende Kombinationen einer Rechenoperation sind zulässig:

sym-ln | *ln-var*		+|-	#I0 ... #I99
sym-ln | *ln-var*		+|-	1L.... *n*L
sym-ln | *ln-var*		+|-	*sym-ln* | *ln-var*
sym-ln | *ln-var*	+|- *zlnr*	+|-	#I0 ... #I99
sym-ln | *ln-var*	+|- *zlnr*	+|-	1L.... *n*L
sym-ln | *ln-var*	+|- *zlnr*	+|-	*sym-ln* | *ln-var*

Die Zeilen (Datensätze) werden im EDT nach dem Einlesen mit 1.0000 - 9999.0000 numeriert. Durch Einfügen von Zeilen und Neunumerierung können sich auch Nummern ungleich 00 nach dem Punkt ergeben.

Beispiele:

`1`	Zeilennummer 1
`500.6789`	Zeilennummer 500.6789
`%+#I10`	n-te Zeile (Wert aus #I10) nach der ersten Zeile
`#L1+5L`	5-te Zeile nach der Zeilennummer aus #L1
`#L1-#L2`	Zeilennummer, errechnet durch Subtraktion #L1 - #L2
`#L1+#I1`	n-te Zeile (Wert aus #I1) nach der Zeilennummer aus #L1

Zeilenbereich ohne Spaltenangabe

rng — *ln1* [*-ln2*] [, ..] | & |
strvar [+|- *intvar*] [. −*strvar* [+|- *intvar*]]

Ein Bereich kann aus einer oder mehreren Zeilen bestehen. Ein Bereich von mehreren hintereinanderliegenden Zeilen wird mit einem Bindestrich zwischen der ersten und der letzten Zeile angegeben. Wird als erste Zeile eine Line-Variable angegeben, muss vor dem Bindestrich das Zeichen "." angegeben werden, um den Bindestrich von dem Operator "-" (Subtraktion) unterscheiden zu können. Mehrere Bereiche werden durch ein Komma getrennt. Alle Zeilen einer Arbeitsdatei werden durch das Zeichen "&" dargestellt. Das Zeichen kann auch umdefiniert werden (Menü Optionen / Sonderzeichen (S. **119**)).

ln1 erste oder einzige Zeile des Bereichs (ausführliche Beschreibung siehe Parameter *ln*).

ln2 letzte Zeile des Bereichs (ausführliche Beschreibung siehe Parameter *ln*).

& Alle Zeilen eines Arbeitsbereichs

strvar [+|- *intvar*] [. −*strvar* [+|- *intvar*]]

Statt Zeilennummern kann auch eine String-Variable oder ein Bereich von mehreren String-Variablen angegeben werden. Die gemischte Angabe von String-Variablen und Zeilennummern ist nicht zulässig. Der Inhalt der Integervariablen erhöht bzw. vermindert die Nr. der Stringvariablen. Die Modifizierung der String-Variablen-Nummer mit dem Inhalt der Integer-Variablen kann in EDT-Prozeduren zur variablen Verarbeitung von String-Variablen verwendet werden.

Beispiele:

```
4
```
Zeile 4

```
5.01-7.5
```
Zeilen 5.01 bis 7.50

```
6,12.02-40,9,66.5,102.01-333
```
Zeile 6, Zeilen 12.02-40.00, Zeile 9, Zeile 66.50 und Zeilen 102.01 bis 333

```
set #L1=100.00
set #L2=300.00
set #L3=700.00
#L2-#L1.-#L3  oder  #L2-#L1.-.#L3
```

Zeilennummer 200-700, die erste Zeile wird durch die Subtraktion 300-100 ermittelt. Vor dem Bindestrich muss hier ein Punkt angegeben werden.

```
set#i1=5
#s1-#s3    oder    #s1.-#s3
#s10-#i1.-#s10              bedeutet #s5 bis #s10
#s1-#s1+#i1                 bedeutet #s1 bis #s6
```

Zeilenbereich mit Spaltenangabe

rngcol *rng* [: *col* [, *col*....] :] [, *rngcol*.....]
strvar [+|- *intvar*] [. −*strvar* [+|- *intvar*]] [: *col* [, *col*....] :]

Ein oder mehrere Zeilenbereiche *rng*, aus denen wahlweise ein Spaltenbereich *col* ausgewählt werden kann. Ist nach *rng* kein Spaltenbereich angegeben, so wird die ganze Zeile ausgewählt. Für jeden Bereich *rng* können eine oder mehrere Spalten bzw. eine oder mehrere Spaltenbereiche angegeben werden.

strvar [+|- *intvar*] [. −*strvar* [+|- *intvar*]]

Statt Zeilennummern kann auch eine String-Variable oder ein Bereich von mehreren String-Variablen angegeben werden. Die gemischte Angabe von String-Variablen und Zeilennummern ist nicht zulässig. Der Inhalt der Integervariablen erhöht bzw. vermindert die Nr. der Stringvariablen. Die Modifizierung der String-Variablen-Nummer mit dem Inhalt der Integer-Variablen kann in EDT-Prozeduren zur variablen Verarbeitung von String-Variablen verwendet werden.

Das Kommando ON (S. **210**) überträgt die Nummer der String-Variablen, in der ein Suchbegriff gefunden wurde, in die Integer-Variable #I99. So kann z.B. mit der Angabe `#S0+#I99` auf diese String-Variable zugegriffen werden.

col Spalte bzw. Spaltenbereich (ausführliche Beschreibung siehe Parameter *col* (S. **338**)).

Beispiele:

```
4
```
Zeile 4, alle Spalten

```
4:1-5:
```
Zeile 4, Spalte 1 bis 5

```
5.01-7.5:3-3,10-20:
```
Zeilen 5.01 bis 7.50, Spalte 3 und Spalten 10 bis 20

```
6,12-40,45:5-10,30-40:,66.55:3:
```
Zeile 6, 12 bis 40 und 45 (Spalten 5 bis 10 und 30-40) und Zeile 66.55, Spalten 3 bis Zeilenende

```
set #L1=100.00
set #L2=200.00
set #I1=10
set #I2=20
#L1.-#L2:#I1.-#I2,30-50:
```

Zeile 100 bis 200, Spalten 10 bis 20 und Spalten 30 bis 50. Vor dem Bindestrich muss ein Punkt angegeben werden, damit der Bindestrich von dem Operator "Minuszeichen" (Subtraktion) unterschieden werden kann.

```
&:10-20:
```
alle Zeilen, Spalten 10 bis 20

```
set#i1=5
#s1-#s3:1-1:    oder    #s1.-#s3:1-1:
#s10-#i1.-#s10:5-10:        bedeutet #s5 bis #s10, Spalte 5-10
#s1-#s1+#i1:10-20:          bedeutet #s1 bis #s6, Spalte 10-20
```

Zeichenfolge

Eine Zeichenfolge *string* kann eine Character-Zeichenfolge, eine Hexa-Zeichenfolge, eine Stringvariable bzw. Teile einer Stringvariablen oder der Inhalt einer Zeile bzw. Teile einer Zeile sein. Es können beliebig viele Einzelstrings miteinander verknüpft werden.

+ | ++ Das Zeichen "+" dient als Verknüpfungszeichen zwischen mehreren Einzelstrings. Falls das Zeichen "+" in Ausnahmefällen nicht eindeutig ist, weil es z.B. für das Konstrukt #L1+#L2 (Inhalt der Zeile, die aus der Zeilennummer #L1 + #L2 gebildet wird), muss als Verknüpfung "++" verwendet werden.

Beispiel:

```
@crea1:'123456'
@crea2:'xyz'
@crea3:'xxZ3'
@#l1=1
@#l2=2
@#s1='abcdef'
@#s2='XXX' + #s1:1-3: + #l1:1-3: + 2:2-3:*3
@#s3=#l1+#l2:3-4:+#l1++#l2
```

Inhalt von #S2: `'XXXabc123yzyzyz'`

Der zusammengesetzte String besteht aus folgenden Einzelstrings:

Character-Zeichenfolge	: `'XXX'`
Spalten 1-3 von #S1	: `'abc'`
Spalten 1-3 von #L1 (Zeile1)	: `'123'`
Spalten 2-3 von Zeile 2 (3 mal)	: `'yzyzyz'`

Inhalt von #S3: `'Z3123456xyz`

Der zusammengesetzte String besteht aus folgenden Einzelstrings:

Spalten 3-4 von #L1+#l2 = Zeile3: `'z3'`
Alle Spalten der Zeile #L1 = Zeile1: `'123567'`
Alle Spalten der Zeile #L2 = Zeile2: `'xyz'`

string { '*char*' | X'*hex*' | V'*char*' | L'*char*' | U'*char*' | *str-var*[:*col*:] | *ln*[:*col*:] } [**int*]

'*char*' Character-Zeichenfolge. Falls die Option LOW OFF (Kleinbuchstaben in Großbuchstaben umwandeln) aktiv ist, wird die Zeichenfolge in Großbuchstaben umgewandelt.

In der Zeichenfolge können auch spezielle **EDT-System-Variable** und **Umgebungsvariable** verwendet werden, wie z.B. aktueller Dateiname, Datum, Zeit usw. Weitere Informationen siehe S. **344**.

X'*hex*' Hexadezimal-Zeichenfolge

`U'`*1234*`'` Hexadezimal-Zeichenfolge im Unicode-Format. Jedes Zeichen besteht aus 4 Halbbytes. Normale ANSI-Zeichen enthalten im 1. Byte X'00' und im 2. Byte den ANSI-Wert.

1 1. Halbbyte des UNICODE-Zeichens
2 2. Halbbyte des UNICODE-Zeichens
3 3. Halbbyte des UNICODE-Zeichens
4 4. Halbbyte des UNICODE-Zeichens

Beispiel:

`U'00300031'` entspricht der Zeichenfolge `'01'`
`U'20AC'` entspricht der Zeichenfolge `'€'`

Es ist zu beachten, dass die Bedeutung des Unicode-Strings `U'1234'` ab der Version 4.0 geändert wurde: Vor der Version 4.0 wurde ein Zeichen des U-Strings jeweils in ein UNICODE-Zeichen umgewandelt, z.B. `U'0'` wurde als `X'0030'` interpretiert. Siehe hierzu auch das Kommando `SHOWNIL` (S. **247**), das in der Version `3` zur Bearbeitung von UNICODE-Dateien zur Verfügung stand.

V'*char*' Diese Variante ist nur als Suchbegriff zulässig. Die Zeichenfolge wird unabhängig von der Klein-/ Großschreibung gesucht, auch wenn unter `Optionen/ Einstellungen/Suchen` oder mittels des Kommandos `SEARCH-OPTION` (S. **244**) die Option "Klein / Großschreibung beachten" aktiviert ist.

L'*char*' Diese Variante ist nur als Suchbegriff zulässig. Die Klein-/ Großschreibung wird beim Suchen beachtet, auch wenn unter `Optione/Einstellungen/Suchen` oder mittels des Kommandos `SEARCH-OPTION` (S. **244**) die Option "Klein / Großschreibung beachten" deaktiviert ist. Wird beim Kommando ON&.... zusätzlich die Option ALL direkt vor L'...' angegeben, darf in diesem Fall "ALL" nicht abgekürzt werden, weil z.B. "ON&CAL'..' als "ON& CHANGE ALL '..' interpretiert wird, vielmehr ist "ON&C ALL L'...' anzugeben.

str-var[:*col*:]
: Zeichenfolgevariable. Der Inhalt der Zeichenfolgevariablen, ggf. eingeschränkt durch die Spaltenangabe *col*, wird als Zeichenfolge benutzt. Die Zeichen in einer Zeichenfolgevariablen werden immer in UNICODE-Codierung gespeichert. Bei Bedarf wird die Zeichenfolgevariable in den Code des entsprechenden Arbeitsbereichs umgewandelt.

ln[:*col*:]
: Als Zeichenfolge wird der Inhalt der Zeile *ln*, ggf. eingeschränkt durch die Spaltenangabe *col*, benutzt. Die Zeilennummer kann entweder direkt oder als Line-Variable angegeben werden.

**int*
: Die wahlweise Angabe **int* ist dafür vorgesehen, eine Zeichenfolge zu wiederholen. Statt einer Ganzzahl kann auch eine Integer-Variable angegeben werden.

col
: **:***cl1* | *cl1-cl2* [,*cl1* | *cl1-cl2*.....] **:**

  Eine oder mehrere durch Komma getrennte Spalten oder Spaltenbereiche:

cl1
: nur Spalte *cl1*. Es ist zu beachten, dass nicht wie bei der Spaltenangabe *col* die Spalten vom *cl1* bis zum Zeilenende ausgewählt werden.

  Dabei haben *cl1* und *cl2* den gleichen Wert. Statt einer absoluten Zahl kann auch eine Ganzzahl-Variable (#I0 - #I99) angegeben werden, in der die Spaltennummer gespeichert ist.

cl1-cl2
: Spaltenbereich von Spalte *cl1* bis Spalte *cl2* (Grenzen eingeschlossen). Die Spalte *cl2* muss größer oder gleich Spalte *cl1* sein.

Beispiele:

`'String1'`	Direkte Angabe eines Strings
`x'0506'`	Direkte Angabe eines Hexa-Strings
`v'string1'`	Suchstring ohne Beachtung der Groß-Kleinschreibung
`'Datei !file'`	es wird ein String mit dem aktuellen Dateinamen erzeugt, z.B. '`Datei testdat`'
`1:4:`	Zeile 1, Spalte 4
`#l1:1-5:`	Zeile in #L1, Spalte 1 bis 5
`#l2:5-5:`	Zeile in #L2, Spalte 5
`#I1=1 500:#I1-5:`	Zeile 500, Spalte 1 bis 5
`#I1=900 #I2=1000 #l3:#I1.-#I2:`	Zeile in L3, Spalte 900 bis 1000. Vor dem Bindestrich muss ein Punkt angegeben werden, damit der Bindestrich von dem Operator "Minuszeichen" (Subtraktion) unterschieden werden kann.

Hinweis:

Wenn in einer Zeichenfolge ein Hochkomma (') vorkommt, muss es verdoppelt werden.

Beispiele:

`'abc'*3`	'abcabcabc'
`X'50'*4`	x'50505050'
`#s1`	Inhalt der Zeichenfolgevariablen #S1
`5:10-20:`	Inhalt der Zeile 5, Spalte 10 bis 20
`#s3:#i4.-#i6:*#i9`	Inhalt der Zeichenfolgevariablen #S3, Spalte #I4 bis #I6 mit Wiederholungsfaktor #I9

Besonderheit für Suchbegriffe im ON-Kommando

Anstelle des einfachen Hochkommas (') kann auch ein Anführungszeichen (") verwendet werden. Wird eine Suchzeichenfolge links oder rechts von einem Anführungszeichen begrenzt, so bringt man dadurch zum Ausdruck, dass links bzw. rechts von dieser Suchzeichenfolge ein Textbegrenzerzeichen stehen muss. Wenn in einem Suchbegriff ein Anführungszeichen vorkommt, muss es verdoppelt werden.

Der Satz der Textbegrenzerzeichen ist durch den EDT vorbelegt mit den Zeichen **+.!*();-/,?:'**="**, dem Leerzeichen (X'20') und dem Tabulatorzeichen (X'09'). Diese Zeichen können mit dem Kommando DELIMIT verändert werden. Per Definition steht vor dem ersten und hinter dem letzten Zeichen immer ein Trennzeichen.

Beispiel:

Über das ON-Kommando soll in einer Zeile mit dem nachfolgenden Inhalt gesucht werden:

```
FLIEGEN+*FLIEGEN  ,FLIEGEN  +FLIEGEN*
1         10         20         30
```

Die Zahlen unter der Zeile zeigen die Spaltennummern an.

Suchbegriff Treffer

'FLIEGEN' auf Spalte 1, 10, 20 und 30
'FLIEGEN" auf Spalte 10 und 20
"FLIEGEN' auf Spalte 1 und 20
"FLIEGEN" auf Spalte 20

Klein- Großschreibung für Suchbegriffe im ON-Kommando

Ist die Option der Menüzeile Optionen / Einstellungen / Suchen /Klein-Großschreibung beachten (S. 110) aktiviert bzw. mit dem Kommando SEARCH-OPTION (S. 244) eingeschaltet, wird die Suchzeichenfolge ohne Beachtung der Klein-Großschreibung verglichen. Die Option gilt nicht für Hexa-Strings (X'string') und L'string'.

Setzt sich ein Suchstring aus mehreren Einzelstrings zusammen, von denen ein Teil ohne und ein Teil mit Berücksichtigung der Klein-Großschreibung verglichen werden soll, wird für den ganzen String die Klein-Großschreibung nicht beachtet.

Beispiel:

```
@search-option caseless=off
@on&find 'xxx'+V'yyy'

@search-option caseless=on
@on&find 'xxx'+x'61'
```

Substitution von EDT-System-Variablen und Umgebungsvariablen in Zeichenfolgen

In allen Zeichenfolgen können spezielle EDT-System-Variablen und Umgebungsvariable verwendet werden, die bei Ausführung des Kommandos durch den entsprechenden Wert ersetzt werden. Das Verändern und Anzeigen von Umgebungsvariablen ist mit dem Kommando SETENV (S. 245) möglich.

Die Substitution erfolgt nur, wenn sie mit der Option "Ersetzung in Zeichenfolgen" im Menü Optionen / Einstellungen / Verschiedenes (S. 127) oder dem Kommando PAR VARSUBST (S. 219) aktiviert wird. Das Zeichen, das den Variablennamen einleitet (Standard = "!"), kann definiert werden, siehe Menü Optionen / Einstellungen / Sonderzeichen2 (S. 121) und Kommando QUOTE (S. 220).

Soll nach der Variablen der String ohne Trennzeichen fortgesetzt werden, kann zur optischen Trennung ein Punkt eingefügt werden. Soll nach der Variablen ein Punkt folgen, müssen Sie 2 Punkte angeben.

Folgende Variablen sind vorgesehen:

!%*env*%	Inhalt der Umgebungsvariablen *env* Umgebungsvariable können in der DOS-Box mit dem Kommando SET gesetzt und angezeigt werden. Innerhalb des EDT können Umgebungsvariable mit dem EDT-Kommando SETENV (S. **245**) geändert und angezeigt werden.
!file	gesamter lokaler Dateiname
!drive	lokaler Laufwerksbuchstabe
!path	lokaler Pfadname
!name	lokaler Dateiname ohne Erweiterung
!lname	lokaler Dateiname mit Erweiterung
!ext	Extension des lokalen Dateinamens
!rfile	gesamter Ausdruck des entfernten READ
!rname	Dateiname oder Elementname eines Bibliothekselements
!rlib	Bibliotheksname der entfernten Datei bei Bibliothekselementen
!rtyp	nur Elementtyp der entfernten Datei bei Bibliothekselementen
!relem	nur Elementname der entfernten Datei bei Bibliothekselementen
!rvers	nur Version der entfernten Datei bei Bibliothekselementen
!rprofil	Name des Filetransfer-Profils
!rparam	entfernte Parameter, die beim READ an die Batch-Prozedur bzw. die Prozedur für die Folgeverarbeitung weitergegeben werden.
!pid	Interne Prozeß-Nummer
!ownip	IP-Adresse des eigenen PC's
!user	Name des Anwenders (nur Win-NT)
!pc	Rechnername
!date	Datum in der Form jjjjmmtt
!time	Uhrzeit in der Form hhmmss
!procnum	Nummer des Prozedur-Arbeitsbereichs
!datanum	Nummer des Daten-Arbeitsbereichs
!actline	Zeilennummer, in der sich der Cursor befindet
!actcol	Spalte, in der sich der Cursor befindet
!font	Aktueller Zeichensatz (Schrift->Fenster)
!fontsize	Aktuelle Fontgröße (Schrift->Fenster)

Beispiele:

eingelesene Datei	`c:\pfad1\test.dat`
Hostname	`host1`
PID	`123`
Datum	`31.12.1999`
Uhrzeit	`14:10:00`
Umgebungsvar. TMP	`c:\temp`

```
'!drive\verzneu\!name..!ext' → 'c:\verzneu\test.dat'
'!name!time..dat'            → 'test141000.dat'
'!name..!date..!time..dat'   → 'test.19991231.141000.dat
'!name!pid'                  → 'test123'
'!%TMP%\test                 → 'c:\temp\test'
```

Suchbegriff für das Kommando ON *rng* SEARCH und S

searchstr { *such* [*vk such .....*] | (*s-dat*) }

such [*col*] [*r*] *item*

col Spaltenbereich, in dem die gesuchte Zeichenfolge beginnen muss.

:col1-col2: Das erste Zeichen der gesuchten Zeichenfolge muss im Spaltenbereich zwischen *col1* und *col2* beginnen.

:col1: Die Zeichenfolge wird nur an der angegebenen Spalte *col1* gesucht und muss dort beginnen.

>:col1: | *<:col1:*
Die Zeichenfolge wird im Bereich ab Spalte *col1* bis Satzende (>:*col1*:) bzw. vom Satzanfang bis Spalte *col1* gesucht (<:*col1*:). Als Spalte kann auch eine Integervariable angegeben werden.

Standard: Suche in gesamten Spaltenbereich (von Spalte 1 bis Zeilenende).

r > | < | -

>	Suche nach einer Zeichenfolge > *item*
<	Suche nach einer Zeichenfolge < *item*
-	Suche nach einer Zeichenfolge ungleich item
Standard:	Suche nach einer Zeichenfolge = *item*.

item Suchzeichenfolge:
C'*string*' | A'*string*' oder V'*string*' | L'*string*' | *stringvar*
X'*hex*' | U'*1234*'

C'*string*' kann zu '*string*' abgekürzt werden.

'*string*' kann in den meisten Fällen auch ohne Hochkommas angegeben werden (on&s *string*). Die Hochkommas dürfen lediglich in den Fällen nicht weggelassen werden, in denen *string* Leerzeichen enthält bzw. wenn nach dem *string* ein weiterer Parameter folgt.

A'*string*' oder V'*string*'

Die Zeichenfolge wird unabhängig von der Klein-/ Großschreibung gesucht, auch wenn unter `Optionen/Einstellungen/Suchen` oder mittels des Kommandos `SEARCH-OPTION` (S. **244**) die Option "Klein / Großschreibung beachten" aktiviert ist.

L'*string*' Die Groß/Kleinschreibung wird beim Suchen beachtet, auch wenn unter `Optionen/Einstellungen/Suchen` oder mittels des Kommandos `SEARCH-OPTION` (S. **244**) die Option "Klein / Großschreibung beachten" deaktiviert ist.

X'*hex*' Hexadezimal-Zeichenfolge

`U'`*1234*`'` Hexadezimal-Zeichenfolge im Unicode-Format. Jedes Zeichen besteht aus 4 Halbbytes. Normale ANSI-Zeichen enthalten im 1. Byte X'00' und im 2. Byte den ANSI-Wert.

1	1. Halbbyte des UNICODE-Zeichens
2	2. Halbbyte des UNICODE-Zeichens
3	3. Halbbyte des UNICODE-Zeichens
4	4. Halbbyte des UNICODE-Zeichens

Beispiel:

`U'00300031'`	entspricht der Zeichenfolge `'01'`
`U'20AC'`	entspricht der Zeichenfolge `'€'`

Es ist zu beachten, dass die Bedeutung des Unicode-Strings `U'1234'` ab der Version 4.0 geändert wurde: Vor der Version 4.0 wurde ein Zeichen des U-Strings jeweils in ein UNICODE-Zeichen umgewandelt, z.B. `U'0'` wurde als `X'0030'` interpretiert. Siehe hierzu auch das Kommando `SHOWNIL` (S. **247**), das in der Version `3` zur Bearbeitung von UNICODE-Dateien zur Verfügung stand.

Enthält *string* Hochkommas (`'`), so müssen diese verdoppelt werden (`''`).

stringvar String-Variable #s1 - #s99, die eine Zeichenfolge enthalten. Es ist zu beachten, dass im Gegensatz zum Kommando `ON...FIND` eine Spaltenangabe nach der Stringvariablen nicht zulässig ist.

Eine Verkettung von Zeichenfolgen mit dem Operator "+", wie bei den sonstigen EDT-Kommandos ist nicht zulässig, weil das Zeichen "+" im Suchbegriff als Verknüpfungsoperator verwendet wird. Ebenfalls ist es nicht zulässig, eine Line-Variable oder eine Zeilennummer als Suchbegriff anzugeben.

vk , | + | *

Verknüpfungsoperator mit dem vorausgegangenen Suchargument *such*.

, Suche das vorausgegangene **oder** das nachfolgende Suchargument. Die Suchbedingung gilt als erfüllt, wenn zumindest eines der beiden Such-Items enthalten ist.

+ Suche das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente enthalten sind. Die Reihenfolge der Suchargumente in der Zeile ist ohne Bedeutung.

***** Suche das vorausgegangene **und** das nachfolgende Suchargument. Die Suchbedingung ist erfüllt, wenn beide Suchargumente enthalten sind. Die Suchargumente müssen in der gleichen Reihenfolge auftreten, wie im Suche-Kommando angegeben.

Es können beliebig viele Konstrukte der Art *vk such* aneinandergereiht werden.

Hinweise:

Die Reihe der angegebenen Suchargumente und Verknüpfungsoperatoren wird linear abgearbeitet. Falls mehrere mit "+" bzw. "*" verknüpfte Suchargumente angegeben wurden und eines von ihnen nicht in der Zeile enthalten ist, so wird der Suchvorgang beendet bzw. beim nächsten, mit oder "," verknüpften Such-Item fortgesetzt.

Für jedes einzelne Suchargument *such* kann ein Spaltenbereich (:col1-col2: / :col: / >:col: / <:col:), eine Negativ-Bedingung (-'*item*'), sowie ein Such-Item in Normaldarstellung ('*item*'), in hexadezimaler Darstellung (X'*item*') oder ein Such-Item für das Ignorieren der Groß-/Kleinschreibung (V'*item*') angegeben werden. Bei einem Such-Item in Normaldarstellung können die Hochkommas in der Regel sogar weggelassen werden. Durch das Kommando "on&sError*500" werden z.B. alle Datensätze gesucht, die die Zeichenfolge 'Error' und irgendwo danach die Zeichenfolge '500' enthalten.

(*s-dat*) Name einer Datei, in der die Suchbegriffe gespeichert sind.

Format der Datei *s-dat*:

Jeder Datensatz in *s-dat* beschreibt eine Suchbedingung, die mit der im nächsten Datensatz enthaltenen Suchbedingung verknüpft wird. Falls am Ende des Datensatzes keines der Verknüpfungszeichen **,/+/*** angegeben wurde, so wird standardmäßig die Oder-Bedingung als Verknüpfung mit dem Suchbegriff im nächsten Datensatz angenommen. Innerhalb eines Datensatzes können mehrere Suchitems mit dem Oder-, Und-, Wildcard-Zeichen verknüpft werden.

Beispiele:

```
on&s'xyz'
```
Suche im gesamten Arbeitsbereich die Zeichenfolge 'xyz'.

```
s,'xyz'
```
Suche ab der ersten angezeigten Zeile bis zum Ende des Arbeitsbereichs Arbeitsbereich die Zeichenfolge 'xyz'.

```
on&s'c'' '''
```
Suche nach der Zeichenfolge C' ' im gesamten Arbeitsbereich (Hochkommas in dem zu suchenden String müssen verdoppelt werden).

```
set #i2=73
set #s2=' '
on&s:#i2:-#s2
```
Suche in Spalte 73 jeder Zeile nach einem Zeichen ungleich Blank. Zeilen mit weniger als 73 Stellen werden hier nicht als Treffer erkannt.

```
on10-20s:100-200:x'47'+:1-2:-'x'
```
Suche in den Zeilen 10 bis 20 jeweils im Spaltenbereich 100 - 200 nach X'47', in den Spalten 1-2 darf kein Zeichen 'x' vorkommen.

```
on&s'='*'(','DC '*'('*')'
```
Es werden alle Zeilen gesucht, die eine der beiden Bedingungen erfüllen:

- Zeichen '=' und irgendwann danach Zeichen '('. z.B. '=A(...)', '=V(...)'
- Zeichenfolge 'DC ' und irgendwo danach die Zeichen '(' und ')'.
  z.B. 'DC A(...)', 'DC Y(...)'

```
on&s-'a'+-'b'+-'c'
```
Es werden alle Datensätze gesucht, die keinen der Kleinbuchstaben a, b oder c enthalten.

```
on&s,'a',>'a'+<'z','z'
```
Es werden alle Datensätze gesucht, die mindestens einen Kleinbuchstaben enthalten.

```
o&s(namen)
```
Es werden in der aktuellen Display-Datei alle Datensätze gesucht, die mindestens einen der in der Datei `namen` aufgeführten Suchbegriffe enthalten. Ein Suchbegriff wird durch einen Satz in der unten stehenden Datei festgelegt.

Die Datei `namen` habe folgenden Inhalt:

```
'ALBERT'
'ANDREAS'
'AMADEUS'+'THEODOR'+'ERNST'
........
'CARL'*'PHILIP'*'EMANUEL'
```

Und-Verknüpfung (+): Ein Satz mit dem String 'AMADEUS' wird nur dann als Treffer gewertet, wenn im gleichen Satz auch die Strings 'THEODOR' und 'ERNST' enthalten sind. Die Reihenfolge der einzelnen Ausdrücke ist bei der Suche mit dem Verknüpfungszeichen '+' ohne Bedeutung. Ein Satz mit 'ERNST THEODOR AMADEUS' würde z.B. die Bedingung erfüllen.

Wildcard-Verknüpfung (*): Ein Satz mit dem String 'CARL' wird nur dann als Treffer gewertet, wenn im gleichen Satz an späterer Stelle die Strings 'PHILIP' und 'EMANUEL' enthalten sind.

Es ist möglich, die Suche jedes einzelnen Strings auf einen bestimmten Spaltenbereich zu begrenzen.

Zusätze für Suchbegriffe in ON-Kommandos

Die Optionen **A**LL, **F**IRST, **R**, **Q**, **P**ATTERN und **N**OT sowie der Parameter *int* in den ON-Kommandos haben folgende Bedeutung:

ALL — Die gewünschte Aktion gilt für alle gefundenen Zeichenfolgen in der Zeile. Ohne diese Option wird die Aktion nur für die erste Zeichenfolge in der Zeile durchgeführt. Diese Option ist nur von Bedeutung für ON-Kommandos, die Daten ändern. Bei der Variante `ON` *range* `FIND` (S. **212**) ohne weitere Parameter (nur Treffer markieren) gilt im Dialogmodus automatisch diese Einstellung. Dieser Standard kann in der Dialogbox `Optionen / Suchen` (S. **110**) geändert werden.

ONCE — Die gewünschte Aktion gilt nur für die erste gefundene Zeichenfolge in der Zeile. Diese Option ist nur von Bedeutung für die Variante `ON` *range* `FIND` ohne weitere Parameter (nur Treffer markieren), falls als Standard die Option `ALL` eingestellt ist.

FIRST — Die gewünschte Aktion gilt nur für die erste gefundene Zeile eines Zeilenbereiches. Sind mehrere Zeilenbereiche angegeben, so wird die Aktion jeweils für die erste gefundene Zeile jedes Zeilenbereiches durchgeführt.

R — Reverse. Die Suche der Zeichenfolge *str* erfolgt jeweils vom Zeilenende Richtung Zeilenanfang. Diese Option ist nur von Bedeutung für ON-Kommandos, die Daten ändern.

NOT — Eine Zeile wird ausgewählt, wenn der Suchbegriff *str* nicht in der Zeile enthalten ist.

PATTERN Musterzeichen im Suchbegriff ersetzen. Es gibt zwei Musterzeichen:

*	ersetzt eine beliebig lange, auch leere Zeichenfolge;
/	ersetzt genau ein Zeichen.

Diese Zeichen (S. **119**) können auch umdefiniert werden.

Q Query. Vor jeder Aktion (CHANGE, PRINT, DELETE, COPY) wird in einer Dialogbox gefragt, ob die Aktion durchgeführt werden soll. Folgende Eingaben sind zulässig:

oder J: Aktion für diese Zeile ausführen;

oder N: Aktion für diese Zeile nicht ausführen;

oder A: Aktion für alle Zeilen ausführen. Bei den folgenden Zeilen erfolgt keine weitere Anfrage.

oder E: Kommando abbrechen.

int Ganze Zahl oder Integer-Variable. Die Suchbedingung ist erst erfüllt, falls der Suchbegriff *str* in einer Zeile so oft vorkommt, wie in *int* angegeben.

13 Kommandogedächtnis

EDT führt eine interne Tabelle für das Kommandogedächtnis, in der alle Eingaben der Kommandozeile aufgezeichnet werden. Auf dieses "Gedächtnis" kann auf zwei verschiedene Arten zugegriffen werden:

a) sequentiell:

Durch Betätigen der Taste `F5` bei leerer Kommandozeile wird die letzte, vorletzte, vorvorletzte usw. Eingabe angezeigt.

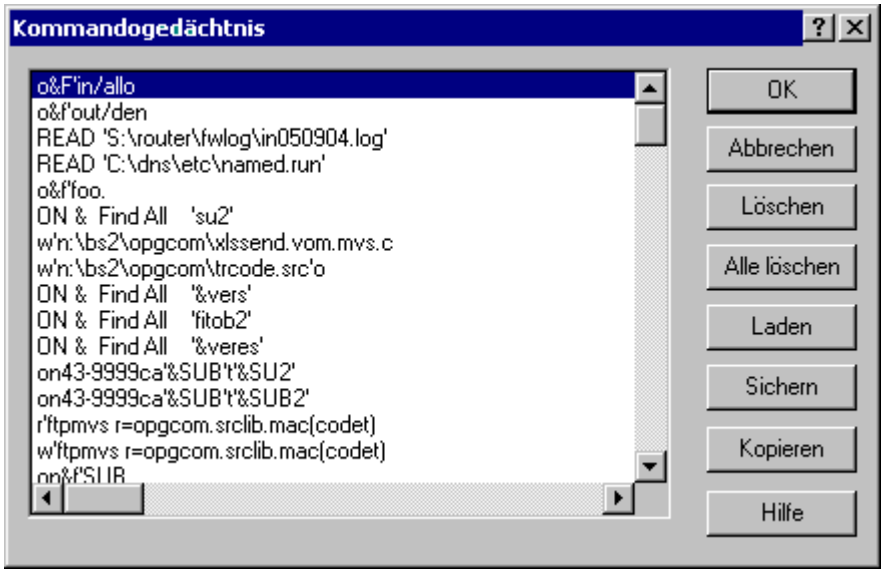
Durch Betätigen der Taste `F6` kann im Kommandogedächtnis wieder vorwärts geblättert werden, das heißt, es wird der zeitlich spätere Eintrag angezeigt.

b) assoziativ:

string F5 bzw.F6. Es wird die letzte Eingabe angezeigt, die mit dem angegebenen Suchmuster beginnt. Durch weiteres Betätigen der Taste `F5` bzw. `F6` wird die vorletzte bzw. nächste Eingabe angezeigt usw.

**string* F5 bzw. F6. Es wird die letzte Eingabe angezeigt, die an irgendeiner Stelle das Suchmuster '*string*' enthält.

c) Fullscreen:



In einem Fenster werden alle bzw. die dem Suchstring entsprechenden Eingaben angezeigt. Der Fullscreen-Modus wird aktiviert, indem in der ersten Stelle der Kommandozeile das Zeichen "-" `<F5>` angegeben oder im Menü `Extras` die Option `Kommandogedächtnis anzeigen` ausgewählt wird. Hier sind auch die Varianten "-*string*" und "-**string*" zulässig. Mit der Maus oder den Tasten `<Cursor_up>` bzw. `<Cursor_down>` kann eine Eingabe ausgewählt und mit der `ENTER`-Taste bzw. einem Doppelklick mit der Maus in das Kommandofeld übernommen werden. Durch Eingabe eines Buchstabens kann auf die Einträge positioniert werden, die mit diesem Buchstaben beginnen.

Beispiele:

del F5
zeigt das letzte delete-Kommando. z.B. del 243-298.65

`*item  F5`

zeigt das letzte Kommando, das an irgendeiner Stelle die Zeichenfolge '*item*' enthält. z.B. ON&C'*item*'T'*item-neu*'

`-o  F5`

zeigt alle Eingaben, die mit "o" beginnen in einem Fenster an. Mit den Cursorbewegungstasten und `<Enter>` wird das gewünschte Kommando aktiviert. Mit '*' wird das gerade ausgewählte Kommando gekennzeichnet.

```
*  on&f'test'
   on&&c'test't'test1'
   o500-600p'file'
   on&d'copy'
   on&find'test'insert suffix 'neu'
   on&find'test'(1)k;1
   on&ca'test't'test2'
```

`-*o  F5`

zeigt alle Eingaben, die irgendwo ein "o" enthalten in einem Fenster an. Mit den Cursorbewegungstasten und <Enter> wird das gewünschte Kommando aktiviert. Mit '*' wird das gerade ausgewählte Kommando angezeigt.

```
*  on&f'test'
   copy 10-50(2)
   code ansi
   comp 1
   on&find'test'insert suffix 'neu'
   on&find'test'(1)k;1
   on&ca'test't'test2'
```

14 Umgebungsvariable

Mit dem MS-DOS-Kommando `SET` (in der Regel in der Datei AUTOEXEC.BAT oder auch in einer eigenen BAT-Datei, die ihrerseits den EDT lädt) können folgende Umgebungsvariable für den EDT gesetzt werden:

APPDATA

Diese Variable wird in der Regel vom System automatisch mit einen für jeden Benutzer eindeutigen Pfad versorgt und wird wie folgt verwendet: Die Datei EDTW.INI wird in das Verzeichnis `%APPDATA%\OPG` geschrieben. Ist die Variable leer, wird das Windows-Systemverzeichnis `WINDOWS` oder `WINNT` benutzt.

Alle Variablen

In den Dialogboxen `Optionen / Sicherung` (S. **106**) und `Optionen / Code-Shell` (S. **116**) können in den Feldern für die Pfadnamen bzw. Dateinamen beliebige Umgebungsvariable angegeben werden, z.B. %TMP% für das Verzeichnis mit den temporären Dateien. Der Inhalt der Variablen kann beliebig mit Dateinamen - und Pfadnamen - Ergänzungen kombiniert werden, z.B. `%VAR1%\temp` oder `d:%VAR1%` oder `c:\tmp\%VAR1%\edtbackup`.

In allen Zeichenfolgen können Umgebungsvariable (S. **344**) verwendet werden, die bei Ausführung des Kommandos durch den entsprechenden Wert ersetzt werden, z.B. `@crea1:'!%TMP%'`. Mit dem Kommando `SETENV` kann eine Liste aller Umgebungsvariablen im Arbeitsbereich 32 erzeugt werden bzw. eine Umgebungsvariable verändert werden.

15 Programmierbare Tasten

EDT bietet die Möglichkeit, 27 Tasten mit einem beliebigen Inhalt von bis zu 256 Zeichen zu programmieren. Als Wert können alle Tasten verwendet werden, also normaler Text (Buchstaben von a - z, Sonderzeichen, Zahlen), Tastencodes wie (`ENTER`, `ESC` usw.) und Positioniertasten (`<TAB>` `<PGDN>` usw.). Beschreibung der Eingabemöglichkeiten siehe S. 133.

Es sind zwei verschiedene Kategorien von programmierbaren Tasten vorgesehen. Abruf des Wertes mit:

`<F7>` Der programmierte Wert kann mit einer Taste abgerufen werden. In dieser Kategorie gibt es nur eine Abrufmöglichkeit, die Taste `F7`.

`<F8> + A-Z` Für den Abruf des programmierten Wertes müssen zwei Tasten verwendet werden, nämlich die Taste `F8` als Einleitungstaste und eine der Tasten **A** bis **Z**. Insgesamt können also 26 Tasten mit je max. 256 Einzeltasten programmiert werden.

Weitere Informationen siehe:

Menüzeile `Extras / Programmtasten ...` (S. 133)

In Zeichenfolgen können auch spezielle EDT-System-Variablen verwendet werden, wie z.B. aktueller Dateiname, Datum, Zeit usw. Weitere Informationen siehe S. 344.

16 Verschlüsselung EDT-Prozeduren

Prozedurdateien können verschlüsselt werden, damit der Anwender den Inhalt der Prozedur nicht lesen kann. Wenn die Prozedur mit dem Kommando `INPUT` bzw. mit dem Schalter `-i` ausgeführt wird, erfolgt automatisch die Entschlüsselung.

Verschlüsselung:

Die Verschlüsselung einer Prozedurdatei erfolgt mit dem Programm `EDTCODE.EXE` durch den Aufruf

`EDTCODE {-o | -p` *passwd* `| -f` *file-passwd*`}` *input output*

-o Verschlüsselung ohne Passwort (kompatibel zum bisherigen Verfahren)

-p *passwd* Passwort in der Länge 4 - 256

-f *file-passwd* Datei, die das Passwort enthält. Als Passwort wird der Inhalt des ersten Satzes in der maximalen Länge von 256 Bytes verwendet.

input Eingabedatei, die zu verschlüsseln ist.

output Verschlüsselte Ausgabedatei.

Entschlüsselung:

Beim Einlesen einer Prozedurdatei mit dem Kommando `INPUT` bzw. über den Schalter `-i` wird eine verschlüsselte Datei automatisch entschlüsselt.

Eine verschlüsselte Prozedurdatei kann wieder in das Ursprungsformat konvertiert werden. Die Entschlüsselung erfolgt ebenfalls mit dem Programm `EDTCODE.EXE` durch den Aufruf

`EDTCODE {-o | -p` *passwd* `| -f` *file-passwd*`}` *input output*

-o Verschlüsselung ohne Passwort (kompatibel zum bisherigen Verfahren)

passwd Passwort in der Länge 4 - 256

file-passwd Datei, die das Passwort enthält. Als Passwort wird der Inhalt des ersten Satzes in der maximalen Länge von 256 Bytes verwendet.

input Verschlüsselte Datei.

output Entschlüsselte Ausgabedatei.

17 Code-Tabellen

Code-Varianten

Auf den verschiedenen Betriebssystemen werden unterschiedliche Codierungen verwendet. Damit mit dem EDT auch Dateien von anderen Plattformen verarbeitet werden können, ist die sowohl das Anzeigen von anders codierten Dateien als auch die Konvertierung von Daten möglich. Die Code-Beschreibungen sind in der Datei `codepage.txt` gespeichert. Diese Datei kann jederzeit um weitere Code-Varianten erweitert werden. Die Datei muss jedoch mindestens die vier Codes mit dem Namen ANSI, ASCIIDOS, ASCIIUNIX und EBCDIC enthalten. Es sind max. 16 Codes zulässig. Die max. Länge einer Anweisung beträgt 128 Zeichen.

Folgende Varianten sind in der ausgelieferten Datei enthalten:

UNICODE	für alle Plattformen	ISO 10646
ANSI	für MS-Windows	Codepage 1252
ISO8859-1	für alle Plattformen	ISO 8859
ASCIIDOS	für MS-DOS	Codepage 850
ASCIIUNIX	für Unix	UNIX-Dateien
EBCDIC	für BS2000	7-Bit: EDF03IRV
EBCDIC8	für BS2000	8-Bit: CCSN EDF041 / EDF04DR
CP500	für OS/390 (IBM)	8-Bit: Codepage 500

UNICODE

Unicode ist ein internationaler Standard, in dem langfristig für jedes sinntragende Zeichen bzw. Textelement aller bekannten Schriftkulturen und Zeichensysteme ein digitaler Code festgelegt wird. Ziel ist es, das Problem unterschiedlicher, inkompatibler Codierungen in unterschiedlichen Ländern oder Kulturkreisen zu beseitigen. Herkömmliche Computer-Zeichencodes umfassen einen Zeichenvorrat von entweder 128 (7 Bit) Codepositionen wie der sehr bekannte ASCII-Standard oder 256 (8 Bit) Positionen, wie z. B. ISO 8859-1 (auch als Latin-1 bekannt), wovon nach Abzug der Steuerzeichen 96 Elemente bei ASCII und 192–224 Elemente bei den 8-Bit ISO-Zeichensätzen als Schrift- und Sonderzeichen darstellbar sind. Diese Zeichencodierungen erlauben die gleichzeitige Darstellung nur weniger Sprachen im selben Text.

In Unicode finden Zeichen der wichtigsten ISO-Zeichensätze wie die ISO-Normen der Serie 8859 eine 1:1-Entsprechung (das bedeutet, dass bei einer Konvertierung von ISO zu Unicode und zurück das gleiche Ergebnis herauskommt, mit Ausnahme des EURO-Zeichens, ISO8859-1 = `X'80'`, UNICODE = `U'20AC'`).

ISO 10646 ist die von ISO verwendete praktisch bedeutungsgleiche Bezeichnung des Unicode-Zeichensatzes; er wird dort als Universal Character Set (UCS) bezeichnet.

ANSI (CP 1252)

Windows-1252 (S. 366) Westeuropäisch (Western European) ist eine 8-Bit-Zeichenkodierung des Microsoft-Betriebssystems Windows, die die meisten westeuropäischen Sprachen unterstützt. Sie baut auf ISO 8859-1 und ISO 8859-15 auf. Sie weicht von ISO-8859-1 im Bereich 80–9F ab, dessen 32 Positionen hier 27 darstellbare Zeichen beinhalten, u. a. die in ISO 8859-15 hinzugekommenen und einige für bessere Typographie notwendige Zeichen.

ISO 8859-1

Der **ISO 8859-1** (S. 367) ist eine Erweiterung des ASCII-Codes. Als ASCII-Code wird die US-Variante des 7-bit-Codes gemäß ISO646 bezeichnet. Neben dem internationalen ASCII-Code gibt es noch weitere nationale Varianten des 7-bit-Codes gemäß ISO646.

Die verschiedenen 8-bit-Codes sind in der internationalen Norm ISO 8859 definiert. Sie haben alle in der "linken" (niederwertigen) Hälfte der Codetabelle einen gemeinsamen Teil, analog zu ASCII, in der "rechten“ (höherwertigen) Hälfte unterscheiden sie sich. Einzelne Codes werden zu Gruppen kompatibler Codes zusammengefaßt, die über ihre ISO-Code-Variantennummer identifiziert werden. Folgende Codes sind zurzeit in ISO 8859 als Standard definiert:

– 8859-1 Latin-1 (West- und Nord-Europa)
– 8859-2 Latin-2 (Ost-Europa, ausgenommen Türkei und die Baltischen Staaten)
– 8859-3 Latin-3 (Mittelmeerraum und Süd-Afrika)
– 8859-4 Latin-4 (Skandinavien und die Baltischen Staaten)
– 8859-5 Kyrillisch
– 8859-6 Arabisch
– 8859-7 Griechisch
– 8859-8 Hebräisch
– 8859-9 Latin-5 (Türkei, West-Europa inklusive Skandinavien)
– 8859-10 Latin-6 (Nord-Europa und die Baltischen Staaten)

ASCIIDOS / ASCIIUNIX

Die ASCII-Codierung unterscheidet sich von der ANSI-Codierung hauptsächlich in den Umlauten und Sonderzeichen.

Für ASCII-Zeichensätze finden Sie in der ausgelieferten Datei `codepage.txt` die Varianten `ASCIIDOS` (S. 368), `ASCIIDOS2 und ASCIIUNIX`.

EBCDIC: 7-bit Code

Als Standard ist im BS2000 der 7-bit-Leitungscode ISO646-IRV und der dazugehörige EBCDIC-Code EBCDIC.DF.03-IRV (CCSN: **EDF03IRV** siehe S. 369) eingestellt. Wenn Sie im 7-bit-Mode arbeiten, verwenden Sie interne Tabellen. Das hat die Vorteile, dass die Arbeit unabhängig von XHCS (Verfügbarkeit, Änderung von Tabellen) ist und volle Kompatibilität mit älteren Versionen der Anwendung besteht.

Das Problem bei dieser Code-Variante besteht darin, dass bestimmte internationale Sonderzeichen und deutsche Umlaute nicht gleichzeitig dargestellt werden können. Dabei geht es um folgende Zeichen:

Deutsche Umlaute:	**ä**	**ö**	**ü**	**Ä**	**Ö**	**Ü**	**ß**
Hexa-Wert	FB	4F	FD	BB	BC	BD	FF
Internationale Zeichen:	{	|	}	[	\	]	~

Die Code-Variante `EBCDIC` verwendet die deutsche Variante.

EBCDIC8: 8-bit-Code

Auf der BS2000-Seite wird der Zeichensatz ISO 8859-1 durch den EBCDIC.DF.04-1 dargestellt. Der EBCDIC (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode), den BS2000 verwendet, muss so erweitert werden, dass jedes Zeichen ein Gegenstück im entsprechenden ISO 8859-x hat. Da EBCDIC-Codes nicht standardisiert sind, existieren unterschiedliche Zuordnungen zwischen EBCDI- und ISO-Codes.

Der **EBCDIC.DF.04-n (CCSN: EDF041** siehe S. 370**)** ist eine Erweiterung des EBCDIC.DF.03-IRV (CCSN: EDF03IRV **=I**nternationale **R**eferenz **V**ersion).

Für EBCDIC 8-Bit-Zeichensätze finden Sie in der ausgelieferten Datei `codepage.txt` die Varianten `EBCDIC8` (wie EDF041 für BS2000) und `CP500` (IBM-Standard).

Weitere Code-Varianten

Durch Erweiterung der Datei `codepage.txt` können jederzeit weitere Code-Varianten hinzugefügt werden, die dann von den Kommando `CODE` (S. 181) verarbeitet werden können.

Datei codepage.txt

Alle Code-Varianten sind in der Datei `codepage.txt` beschrieben. Es können vorerst max. 48 Codes definiert werden (Programmerweiterung bei Bedarf möglich). Die Reihenfolge der Codes bestimmt die interne Code-Nummer, die für die Speicherung von Einstellungen in der INI-Datei verwendet wird. Die Reihenfolge sollte daher nicht geändert werden, da sonst die gespeicherten Einstellungen zu Fehlinterpretationen führen.

Bei der Installation mit dem SETUP-Programm ist zu beachten, dass eine bereits bestehende `codepage.txt` nicht überschrieben wird, da die Datei auch vom Benutzer angepaßt werden kann. Soll die aktuelle `codepage.txt` übernommen werden, so muss die Datei `codepage.opg` nach codepage.txt kopiert werden.

Die Datei enthält für jede Code-Variante als Header die Anweisungen `CODE, NOTE und PARAM` sowie für jedes Zeichen von 00 bis FF eine Zeile mit der Definition des entsprechenden Unicode-Zeichens.

Neben den Beschreibungen der Code-Varianten muss noch die Tabelle `LOWCAPTAB` definiert werden, die alle Buchstaben enthält und für die Konvertierung von Klein- in Großbuchstaben und umgekehrt (siehe Kommandos `LOWER` und `UPPER`) verwendet wird.

`*`*bemerkung* Bemerkungszeile

`CODE=name,alias,alias,....`

Codename und Aliasnamen. Für das Kommando `CODE` können wahlweise der Codename oder einer der Aliasnamen benutzt werden. In den Dialogboxen wird immer nur der Codename angezeigt.

`NOTE=`*beschreibung*

Kurzbeschreibung des Codes für die Darstellung in den Comboboxen des Dialogs `Optionen / Einstellungen / Code` und `Funktionen / Code` bzw. des Menüs `Ansicht / Code`. In diesen Dialogelementen werden der Name und diese Beschreibung angezeigt. Die Beschreibung darf max. 50 Zeichen lang sein.

`PARAM=ss uu` Definition Leerzeichen und undefinierte Zeichen.

`ss` Dieses Zeichen wird während der Bearbeitung von Daten zum Auffüllen mit Leerzeichen verwendet. Dies kommt z.B. vor, wenn eine neue Zeile per Tastatureingabe erzeugt und das erste Zeichen in Spalte 10 eingegeben wird. Die Spalten 1-9 werden mit Leerzeichen aufgefüllt.

`uu` Sind in einer Code-Variante bestimmte Zeichen nicht belegt, wie z.B. beim 7-Bit EBCDIC-Code oder ist bei der Konvertierung ein Zeichen des "Sende-Codes" im "Ziel-Code" nicht vorhanden, wird dieses Zeichen erzeugt.

`BITS=x`

`x` kann sein: 7 oder 8, für einen 7-Bit-Code oder 8-Bit-Code. Der Wert 8 braucht nicht angegeben werden, dies ist der Standard.

`Korresp=xxxxxxx`

Name des korrespondierenden Codes. Bei einem ANSI-Code ist der Name des entsprechenden EBCDIC-Codes, bei einem EBCDIC-Code ist der Name des entsprechenden ANSI-Codes hier anzugeben.
Dieser Parameter ist optional.

`CC UUUU A` **#***bemerkung*

Beschreibung jedes Zeichens (pro Zeichen eine Zeile mit bis zu 4 Spalten). Für jede Code-Variante müssen alle 256 Zeichen beschrieben werden.

In dieser Beschreibung wird das entsprechende Unicode-Zeichen für die Code-Umwandlung definiert. Eine Code-Umwandlung erfolgt immer in zwei Schritten: 1. Umwandlung vom alten Code in Unicode, 2. Umwandlung von Unicode in den neuen Code. Die Tabelle Unicode → Code-Variante wird vom Programm automatisch generiert.

Die Beschreibung enthält außerdem Kennzeichnung für die automatische Code-Erkennung.

`CC` Zeichen 2-stellig hexadezimal

`UUUU` Entsprechendes Unicode Zeichen 4-stellig hexadezimal. Die ersten 256 Zeichen des Unicodes und die Zeichen des Codes ISO 8859 sind identisch. Die weiteren Zeichen des Unicodes enthalten spezielle themen- bzw. sprachabhängige Sonderzeichen, wie z.B. das EURO-Zeichen (`X'20AC'`). Die vollständige Beschreibung aller Unicode-Zeichen ist in der Datei `unicode.txt` enthalten. Weitere Informationen finden Sie im Internet unter **www.unicode.org**.

Ist ein Zeichen dieses Codes nicht belegt bzw. soll es bei der Konvertierung in ein Schmierzeichen übersetzt werden, ist als Unicode FFFF anzugeben.

`A` 1 = Zeichen für automatische Code-Erkennung (optional bis Version 3.21). Ab Version 4.0 wird diese Option ignoriert.

Im Menü `Optionen / Einstellungen / Code` kann die Option "Auto" eingestellt werden. Beim Einlesen einer Datei wird anhand von typischen Zeichen festgestellt, um welchen Code es sich handelt und dann die Anzeige auf diese Code-Variante eingestellt. Die maßgebenden Zeichen für die automatische Erkennung müssen mit "1" gekennzeichnet werden.

Ab Version 4.0 wird bei der Einstellung "Auto" der Code wie folgt ermittelt: Die ersten 50.000 Bytes einer Datei werden mit allen Codes, die in der Datei `codepage.txt` enthalten sind, nach UNICODE umgewandelt und nach Buchstaben, Umlauten und Zahlen untersucht. Der Code mit den meisten Treffern wird ausgewählt. Bei gleichen Treffern wird ANSI bevorzugt.

#*bemerkung* Bemerkung (optional)

Die gesamte Zeile darf max. 128 Zeichen lang sein.

Hinweise:

Die Spalten werden durch ein Leerzeichen getrennt. Die Spalten 3 - 4 sind optional, es können jedoch nur rechtsbündig Spalten entfallen, z.B. "CC UUUU", "CC UUUU A".

Falls in einem Code Zeichen nicht besetzt sind, wie z.B. beim 7-Bit EBCDIC-Code, muss für dieses Zeichen das Unicode-Zeichen FFFF eingetragen werden.

Da vom Programm automatisch eine zweite Tabelle UNICODE-→ Code-Variante aufgebaut wird, darf ein Unicode-Zeichen mit Ausnahme von FFFF nur einmal vorkommen.

Die Reihenfolgen der Anweisungen muss eingehalten werden:
1. CODE=
2. NOTE=
3. PARAM=
4. 256 Zeichenbeschreibungen.

`LOWCAPTAB` Beginn der Tabelle für die Konvertierung von Klein- in Großbuchstaben und umgekehrt. Die Definition der Buchstaben gilt für alle Code-Varianten. Danach folgt für jeden Buchstaben die folgende Anweisung. Die Reihenfolge der Eintragungen in diesem Abschnitt der `codepage.txt` ist unbedeutend.

`GGGG KKKK` #*bemerkung*

`GGGG` Unicode-Wert für den Großbuchstaben

`KKKK` Unicode-Wert für den Kleinbuchstaben

Beispiel codepage.txt:

```
* Beispiel Codepage.txt auszugsweise
* 1. Code-Variante
CODE=ASCIIDOS,ASCII,DOS,CP850
NOTE=ASCII MS-DOS (CP850)
PARAM=20 00
00 0000 #NULL
01 0001 #START OF HEADING
02 0002 #START OF TEXT
03 0003 #END OF TEXT
04 0004 #END OF TRANSMISSION
* Hier folgen weitere Anweisungen 05 bis 40
41 0041 #LATIN CAPITAL LETTER A
42 0042 #LATIN CAPITAL LETTER B
43 0043 #LATIN CAPITAL LETTER C
44 0044 #LATIN CAPITAL LETTER D
45 0045 #LATIN CAPITAL LETTER E
80 00C7 # Ç LATIN CAPITAL LETTER C WITH CEDILLA
81 00FC # ü LATIN SMALL LETTER U WITH DIAERESIS
82 00E9 # é LATIN SMALL LETTER E WITH ACUTE
83 00E2 # â LATIN SMALL LETTER A WITH CIRCUMFLEX
* Hier folgen weitere Anweisungen 84-FF

* 2. Code-Varianten
CODE=ASCII2
NOTE=ASCII Variante2
PARAM=20 00
00 0000 #NULL
01 0001 #START OF HEADING
* Hier folgen weitere Anweisungen 02-FF

* Tabelle mit den Zuweisungen der Klein/Gross-
* Schreibungen: erste Spalte Grossbuchstabe
* zweite Spalte Kleinbuchstabe
LOWCAPTAB
0041 0061 #LATIN LETTER A
0042 0062 #LATIN LETTER B
0043 0063 #LATIN LETTER C
0044 0064 #LATIN LETTER D
0045 0065 #LATIN LETTER E
0046 0066 #LATIN LETTER F
* Hier folgen weitere Anweisungen für die restlichen
* Buchstaben
```

Code-Funktionen

Editier-Code auswählen

Mit dem Kommando CODE Format 1 (S. 181) oder mit dem Menübefehl Ansicht / Code-Anzeige (S. 94) kann ein abweichender Editier-Code ausgewählt werden. Dies bewirkt, dass die Darstellung der abdruckbaren Zeichen umgestellt wird und dass alle Datenänderungen im ausgewählten Code durchgeführt werden. Im Hexa-Modus (Kommando HEX ON) wird z.B. im EBCDIC-Code das Zeichen "A" mit "C1" wie im BS2000 dargestellt. Die Codierung einer Datei wird in bestimmten Fällen beim Einlesen automatisch erkannt und eingestellt (siehe Menü Optionen / Code Standard, Option Auto als Standard-Code (S. 116)).

Die aktuelle Code-Einstellung wird in der Statuszeile angezeigt.

Die Darstellung der Zeichen erfolgt im Unicode-Modus, d.h. es können auch Zeichen, die lt. Definition in der Datei `codepage.txt` auf Unicode-Zeichen > 256 verweisen, angezeigt werden, soweit sie im aktuell eingestellten Windows-Zeichensatz verfügbar sind. Dies betrifft z.B. die Rahmenzeichen der Code-Variante ASCIIDOS (X'C0' - X'DF').

Codierung der Daten ändern

Die bestehenden Daten können mit dem Format 2 des Kommando `CODE` (S. **181**) bzw. über das Menü `Funktionen/Code-Umsetzung` (S. **83**) konvertiert werden.

Groß- Kleinschreibung ändern

Das Kommandos `UPPER` (S. **258**) und `LOWER` (S. **208**) sowie die Actioncodes U und L (S. **175**) benutzen die Angaben in den Codetabellen für die Konvertierung in Klein- bzw. Großbuchstaben.

UNICODE hexadezimal anzeigen

In UNICODE-Arbeitsbereichen kann mit dem Kommando `HEX4` (S. **206**) auf eine vierzeilige Hexa-Darstellung umgeschaltet werden. Es werden somit alle 16 Bit bzw. vier Halbbytes untereinander dargestellt.

Windows Codetabelle CP 1252

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F		!	"	v	$	%	v	'	(	)	*	+	,	-	.	/
30-3F	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40-4F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50-5F	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60-6F	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70-7F	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80-8F	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
90-9F		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
A0-AF		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B0-BF	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0-CF	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0-DF	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0-EF	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0-FF	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Windows-1252 Westeuropäisch (Western European) ist eine 8-Bit-Zeichenkodierung des Microsoft-Betriebssystems Windows, die die meisten westeuropäischen Sprachen unterstützt. Sie baut auf ISO 8859-1 und ISO 8859-15 auf. Sie weicht von ISO-8859-1 im Bereich 80–9F ab, dessen 32 Positionen hier 27 darstellbare Zeichen beinhalten, u. a. die in ISO 8859-15 hinzugekommenen und einige für bessere Typographie notwendige Zeichen.

ASCII Code-Tabelle ISO 8859-1 Latin Alphabet No. 1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F		!	"	v	$	%	v	'	(	)	*	+	,	-	.	/
30-3F	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40-4F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50-5F	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60-6F	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70-7F	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80-8F	€															
90-9F																
A0-AF		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B0-BF	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C0-CF	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D0-DF	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E0-EF	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F0-FF	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Der **ISO 8859-1** ist eine Erweiterung des ASCII-Codes. Als ASCII-Code wird die US-Variante des 7-bit-Codes gemäß ISO646 bezeichnet. Neben dem internationalen ASCII-Code gibt es noch weitere nationale Varianten des 7-bit-Codes gemäß ISO646.

ASCII Code-Tabelle CP850DOS Latin1

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F		!	"	v	$	%	v	'	(	)	*	+	,	-	.	/
30-3F	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40-4F	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50-5F	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	___
60-6F	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70-7F	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	⌂
80-8F	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
90-9F	È	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	ø	£	Ø	×	ƒ
A0-AF	á	í	ó	ú	ñ	Ñ	ª	º	¿	®	¬	½	¼	¡	«	»
B0-BF	▒	▓	▓	|	┤	Á	Â	À	©	╣	║	╗	╝	¢	¥	┐
C0-CF	└	┘	┴	├	─	┼	ã	Ã	╚	╔	╩	╦	╠	═	╬	¤
D0-DF	ð	Ð	Ê	Ë	È		Í	Î	Ï	┘	┌	▌	▄	¦	Ì	▀
E0-EF	Ó	ß	Ô	Ò	õ	Õ	µ	þ	Þ	Ú	Û	Ù	ý	Ý	¯	´
F0-FF	-	±	‗	¾	¶	§	÷	¸	°	¨	·	¹	³	²	■	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

EBCDIC Code-Tabelle EBCDIC.DF.03 (CCSN:EDF03IRV)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F																
30-3F																
40-4F	SP										`	.	<	(	+	|ö
50-5F	&										!	$	*	)	;	
60-6F	-	/									^	,	%	_	>	?
70-7F											:	#	@	'	=	"
80-8F		a	b	c	d	e	f	g	h	i						
90-9F		j	k	l	m	n	o	p	q	r						
A0-AF			s	t	u	v	w	x	y	z						
B0-BF												[Ä \ Ö] Ü				
C0-CF		A	B	C	D	E	F	G	H	I						
D0-DF		J	K	L	M	N	O	P	Q	R						
E0-EF			S	T	U	V	W	X	Y	Z						
F0-FF	0	1	2	3	4	5	6	7	8	9		{ ä		} ü		~ ß
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

EBCDIC Code-Tabelle EBCDIC.DF.04.1 (CCSN:EDF041)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F																
30-3F																
40-4F	SP	NbS	â	ä	à	á	ã	å	ç	ñ	`	.	<	(	+	|
50-5F	&	é	ê	ë	è	í	î	ï	ì	ß	!	$	*	)	;	
60-6F	-	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	^	,	%	_	>	?
70-7F	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	¨	:	#	@	'	=	"
80-8F	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
90-9F	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A0-AF	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B0-BF	¢	£	¥	·	©	§	¶	¼	½	¾	¬	[	\	]	´	×
C0-CF	ù	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D0-DF	¦	J	K	L	M	N	O	P	Q	R	¹	û	ü	Û	ú	ÿ
E0-EF	Ù	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F0-FF	0	1	2	3	4	5	6	7	8	9	³	{	Ü	}	Ú	~
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

EBCDIC Code-Tabelle EBCDIC.DF.04-DRV (CCSN:EDF04DR)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F																
30-3F																
40-4F	SP	NbS	â	|	à	á	ã	å	ç	ñ	`	.	<	(	+	ö
50-5F	&	é	ê	ë	è	í	î	ï	ì		!	$	*	)	;	¯
60-6F	-	/	Â	¦	À	Á	Ã	Å	Ç	Ñ	^	,	%	_	>	?
70-7F	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	¨	:	#	§	'	=	"
80-8F	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
90-9F	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A0-AF	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B0-BF	¢	£	¥	·	©	@	¶	¼	½	¾	¬	Ä	Ö	Ü	´	×
C0-CF	{	A	B	C	D	E	F	G	H	I	-	ô	[	ò	ó	õ
D0-DF	}	J	K	L	M	N	O	P	Q	R	¹	û	]	ù	ú	ÿ
E0-EF	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Û	Ò	Ó	Õ
F0-FF	0	1	2	3	4	5	6	7	8	9	³	ä	Ú	ü	Ù	ß
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

EBCDIC Code-Tabelle CP500

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00-0F																
10-1F																
20-2F																
30-3F																
40-4F	SP	NbS	â	ä	à	á	ã	å	ç	ñ	[	.	<	(	+	!
50-5F	&	é	ê	ë	è	í	î	ï	ì	ß	]	$	*	)	;	^
60-6F	-	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
70-7F	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	"
80-8F	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
90-9F	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A0-AF	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B0-BF	¢	£	¥	·	©	§	¶	¼	½	¾	¬	|	¯	¨	´	×
C0-CF	{	A	B	C	D	E	F	G	H	I		ô	ö	ò	ó	õ
D0-DF	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	Û	ú	ÿ
E0-EF	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F0-FF	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	Ÿ
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Translate-Tabellen für Filetransfer OpenFT (FTBS)

Für den Filetransfer mit OpenFT werden folgende Translate-Tabellen benutzt. Die Tabellen können mit Benutzer-Tabellen überschrieben werden, siehe auch Menü `Extras/Filetransfer/Filetrasfer-Profile` (S. 140).

EBCDIC → ASCII

	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**0A**	**0B**	**0C**	**0D**	**0E**	**0F**
00	00	01	02	03	85	09	86	7F	87	8D	8E	0B	0C	0D	0E	0F
10	10	11	12	13	8F	0A	08	97	18	19	9C	9D	1C	1D	1E	1F
20	80	81	82	83	84	92	17	1B	88	89	8A	8B	8C	05	06	07
30	90	91	16	93	94	95	96	04	98	99	9A	9B	14	15	9E	1A
40	20	A0	E2	E4	E0	E1	E3	E5	E7	F1	60	2E	3C	28	2B	7C
50	26	E9	EA	EB	E8	ED	EE	EF	EC	DF	21	24	2A	29	3B	9F
60	2D	2F	C2	C4	C0	C1	C3	C5	C7	D1	5E	2C	25	5F	3E	3F
70	F8	C9	CA	CB	C8	CD	CE	CF	CC	A8	3A	23	40	27	3D	22
80	D8	61	62	63	64	65	66	67	68	69	AB	BB	F0	FD	FE	B1
90	B0	6A	6B	6C	6D	6E	6F	70	71	72	AA	BA	E6	B8	C6	A4
A0	B5	AF	73	74	75	76	77	78	79	7A	A1	BF	D0	DD	DE	AE
B0	A2	A3	A5	B7	A9	A7	B6	BC	BD	BE	AC	5B	5C	5D	B4	D7
C0	F9	41	42	43	44	45	46	47	48	49	AD	F4	F6	F2	F3	F5
D0	A6	4A	4B	4C	4D	4E	4F	50	51	52	B9	FB	FC	DB	FA	FF
E0	D9	F7	53	54	55	56	57	58	59	5A	B2	D4	D6	D2	D3	D5
F0	30	31	32	33	34	35	36	37	38	39	B3	7B	DC	7D	DA	7E

ASCII → EBCDIC

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	00	01	02	03	37	2D	2E	2F	16	05	15	0B	0C	0D	0E	0F
10	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
20	40	5A	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
30	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
40	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
50	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	BB	BC	BD	6A	6D
60	4A	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
70	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	FB	4F	FD	FF	07
80	20	21	22	23	24	04	06	08	28	29	2A	2B	2C	09	0A	14
90	30	31	25	33	34	35	36	17	38	39	3A	3B	1A	1B	3E	5F
A0	41	AA	B0	B1	9F	B2	D0	B5	79	B4	9A	8A	BA	CA	AF	A1
B0	90	8F	EA	FA	BE	A0	B6	B3	9D	DA	9B	8B	B7	B8	B9	AB
C0	64	65	62	66	63	67	9E	68	74	71	72	73	78	75	76	77
D0	AC	69	ED	EE	EB	EF	EC	BF	80	E0	FE	DD	FC	AD	AE	59
E0	44	45	42	46	43	47	9C	48	54	51	52	53	58	55	56	57
F0	8C	49	CD	CE	CB	CF	CC	E1	70	C0	DE	DB	DC	8D	8E	DF

18 UNICODE-Unterstützung

Was ist UNICODE?

Unicode ist ein internationaler Standard, in dem langfristig für jedes sinntragende Zeichen bzw. Textelement aller bekannten Schriftkulturen und Zeichensysteme ein digitaler Code festgelegt wird. Ziel ist es, das Problem unterschiedlicher, inkompatibler Codierungen in unterschiedlichen Ländern oder Kulturkreisen zu beseitigen. Herkömmliche Computer-Zeichencodes umfassen einen Zeichenvorrat von entweder 128 (7 Bit) Codepositionen wie der sehr bekannte ASCII-Standard oder 256 (8 Bit) Positionen, wie z. B. ISO 8859-1 (auch als Latin-1 bekannt), wovon nach Abzug der Steuerzeichen 96 Elemente bei ASCII und 192–224 Elemente bei den 8-Bit ISO-Zeichensätzen als Schrift- und Sonderzeichen darstellbar sind. Diese Zeichencodierungen erlauben die gleichzeitige Darstellung nur weniger Sprachen im selben Text.

In Unicode finden Zeichen der wichtigsten ISO-Zeichensätze wie die ISO-Normen der Serie 8859 eine 1:1-Entsprechung (das bedeutet, dass bei einer Konvertierung von ISO zu Unicode und zurück das gleiche Ergebnis herauskommt, mit Ausnahme des EURO-Zeichens, ISO8859-1 = `X'80'`, UNICODE = `U'20AC'`).

ISO 10646 ist die von ISO verwendete praktisch bedeutungsgleiche Bezeichnung des Unicode-Zeichensatzes; er wird dort als Universal Character Set (UCS) bezeichnet.

Die Speicherung und Übertragung von Unicode erfolgt in unterschiedlichen Formaten ("Encodings“):

UTF-8 — UTF-8 (Abk. für 8-Bit Unicode Transformation Format) ist die am weitesten verbreitete Codierung für Unicode-Zeichen. Dabei wird jedem Unicode-Zeichen eine speziell codierte Bytekette von variabler Länge zugeordnet. UTF-8 unterstützt bis zu vier Byte, auf die sich wie bei allen UTF-Formaten alle 1.114.112 Unicode-Zeichen abbilden lassen. Das UTF-8-Header enthält X'EFBBBF'

UTF-16 — UTF-16 (Abk. für 16-bit Unicode Transformation Format) ist eine Codierung. bei der jedes Unicode-Zeichen mind. 16 Bit belegt. Unicode-Zeichen außerhalb der BMP (Basic multilingual plane, d.h. U+10000 bis U+10FFFF), werden durch zwei 16-Bit-Wörter (engl. code units) dargestellt, die wie folgt gebildet werden:

Von der Nummer des Zeichens wird 65536 (10000hex) abgezogen. So können mehr Zeichen kodiert werden. Danach wird das Ergebnis in zwei Blöcke zu je 10 Bit zerteilt. Dem ersten Block wird die Bitfolge 110110 und dem zweiten Block wird die Bitfolge 110111 vorangestellt. Das erste der so entstandenen 16-Bit-Wörter bezeichnet man als High-Surrogate, das zweite als Low-Surrogate. Diesen Namen entsprechend enthält das High-Surrogate die höherwertigen, das Low-Surrogate die niederwertigen Bits des Unicode-Zeichens. Der Codebereich von U+D800 bis U+DBFF (High-Surrogates) und der Bereich von U+DC00 bis U+DFFF (Low-Surrogates) ist für diese UTF-16-Ersatzzeichen reserviert und enthält keine eigenständigen Zeichen.

Von UTF-16 gibt es zwei Varianten:

a) UTF-16BE (Big-Endian), im Folgenden als UCB bezeichnet:

Das höherwertige Byte steht in den ersten 8 Bits, z.B. Die Zahl "0" (X'30') erhält den Wert X'0030'. Der UTF-16-Header enthält X'FEFF'.

b) UTF-16LE (Little-Endian), im Folgenden als UCL bezeichnet:

Das niederwertige Byte steht in den ersten 8 Bits, z.B. Die Zahl "0" (X'30') erhält den Wert X'3000'. Der UTF-16-Header enthält X'FFFE'

sonstige — SCSU (Standard Compression Scheme for Unicode, früher auch als RCSU – Reuters' Compression Scheme for Unicode – bezeichnet), UTF-EBCDIC, Punycode (Domainnamen, IDNA) BOCU (Binary-Ordered Compression for Unicode) CESU-8 und GB18030 werden nicht unterstützt.

Unterstützung von UNICODE-Dateien ab der Version 4.0

Die Daten der Arbeitsbereiche werden intern als Zwei-Byte-Zeichen gespeichert und zwar im Format Big-Endian (höherwertiges Byte zuerst).

UNICODE-Dateien im Format UTF-8 und UTF-16 Little-Endian (niederwertiges Byte zuerst) werden beim Lesen in das Format UTF-16 Unicode Big-Endian (höherwertiges Byte zuerst) umgewandelt.

Dateien, die in einem Ein-Byte-Code codiert sind, wie ISO8859-1 (ANSI), EBCDIC oder ASCII, werden beim Lesen nicht umcodiert, sondern linksbündig um X'00' ergänzt und im Ursprungscode gespeichert und editiert.

Daten in Stringvariablen und in der Zwischenablage werden immer in UNICODE codiert und bei Verwendung in Arbeitsbereichen in den Zielcode übersetzt. Beim Kopieren und Verschieben von Daten aus UNICODE-Arbeitsbereichen in andere Arbeitsbereiche und umgekehrt wird ggf. eine Umcodierung vorgenommen.

Erweiterungen für die Verarbeitung von UNICODE-Dateien

Folgende Kommandos und Menü-Funktionen wurden für die Verarbeitung von UNICODE-Dateien erweitert:

a) Neue Optionen zum Kommando `READ` (S. **220**) und `WRITE` (S. **261**):

`CHAR:` Datei im 1-Byte-Format (Standard, falls kein UNICODE-Header)
`UTF:` Datei im UTF-8-Format
`UCB:` UTF-16 Unicode Big-Endian (höherwertiges Byte zuerst)
`UCL:` UTF-16 Unicode Little-Endian (niederwertiges Byte zuerst)
`H:` Beim Schreiben Header voranstellen (Standard)
`NH:` Datei ohne Header schreiben

Falls einzulesende Dateien einen UNICODE-Header (Byte Order Mark) enthalten, wird automatisch die richtige Codierung verwendet. Das Gleiche gilt, falls in der Datei das UNICODE-Satzende-Kennzeichen `X'000D000A'` bzw. `X'0D000A00'` oder ein XML-Header mit UTF-Encoding vorkommt.

b) In allen Kommandos, bei denen Zeichenfolgen erlaubt sind, kann auch der Zeichenfolge-Typ `U'1234'` für Unicode-Strings (S. **341**) verwendet werden. Dabei werden für jedes UNICODE-Zeichen 4 Halbbytes angegeben, z.B. `U'20AC'` für das Zeichen '€'. Diese Art von Zeichenfolge darf nur in einem UNICODE-Arbeitsbereich verwendet werden.

c) Kommando `CODE` (S. **181**), Menü `Funktionen / Code-Umsetzung` (S. **83**), Menü `Ansicht / Anzeige-Code` (S. **94**) und Menü `Optionen / Code-Darstellung für neue Arbeitsbereiche` (S. **117**): Als Anzeige-Code bzw. Ziel- oder Sendecode kann auch UNICODE angegeben werden.

c) Menü `Optionen / Code/Shell / Dateien ohne UNICODE-Header` (S. **117**): Wahlweise kann eine automatische Erkennung für UTF8-Dateien eingeschaltet werden.

d) Das Kommando `HEX` (S. **206**) wurde um die Optionen "2" (2-zeilig) und "4" (4-zeilig) erweitert. Das Kommando `HEX 4` ist aber nur in einem UNICODE-Arbeitsbereich zulässig.

e) Kommando `COMPARE` (S. **184**): Arbeitsbereiche für das Compare-Protokoll und der Protokoll-Arbeitsbereich 32 werden automatisch in Unicode-Codierung angezeigt, falls Daten in Unicode-Codierung enthalten sind.

f) Kommando `VAR LOAD` (S. **258**) kann alte und neue (UNICODE) Dateien zum Laden von String-Variablen verarbeiten.

g) Mit dem Menü `Funktionen / Einfügen Sonderzeichen...` (S. **91**) können alle zulässigen Zeichen eines Zeichensatzes in das Datenfenster oder in die Kommandozeile eingefügt werden.

Stichwortverzeichnis

D

E

F

G

H

I

K

L

M

N

Ö

P

Q

R

S

T

Ü

V

W

Z

Dieses Benutzerhandbuch wurde mit Microsoft WORD erstellt.
Der Ausdruck erfolgte auf einer DocuTech 135 von Rank Xerox.

OPG Online-Programmierung GmbH